脆弱的NODE.JS

SECURITY RISK IN NODE WEB

姜天意

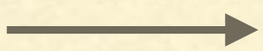
关于俺 / ABOUT ME

- F2E @ alibaba.com/1688.com
- GitHub.com/jtyjty99999
- Data visualization & mobile & Node.js

TOPIC

- 为什么我们关注Node.js安全
- Node.js安全风险介绍
- egg-security & 安全开发

~~8~~900+ Application build in Node.js



https://www.quora.com/What-companies-are-using-Node-js-in-production-in-Texas

Home > Software Development > Node.js

INFOWORLD TECH WATCH

By Paul Krill, Editor at Large, InfoWorld | JUN 20, 2014

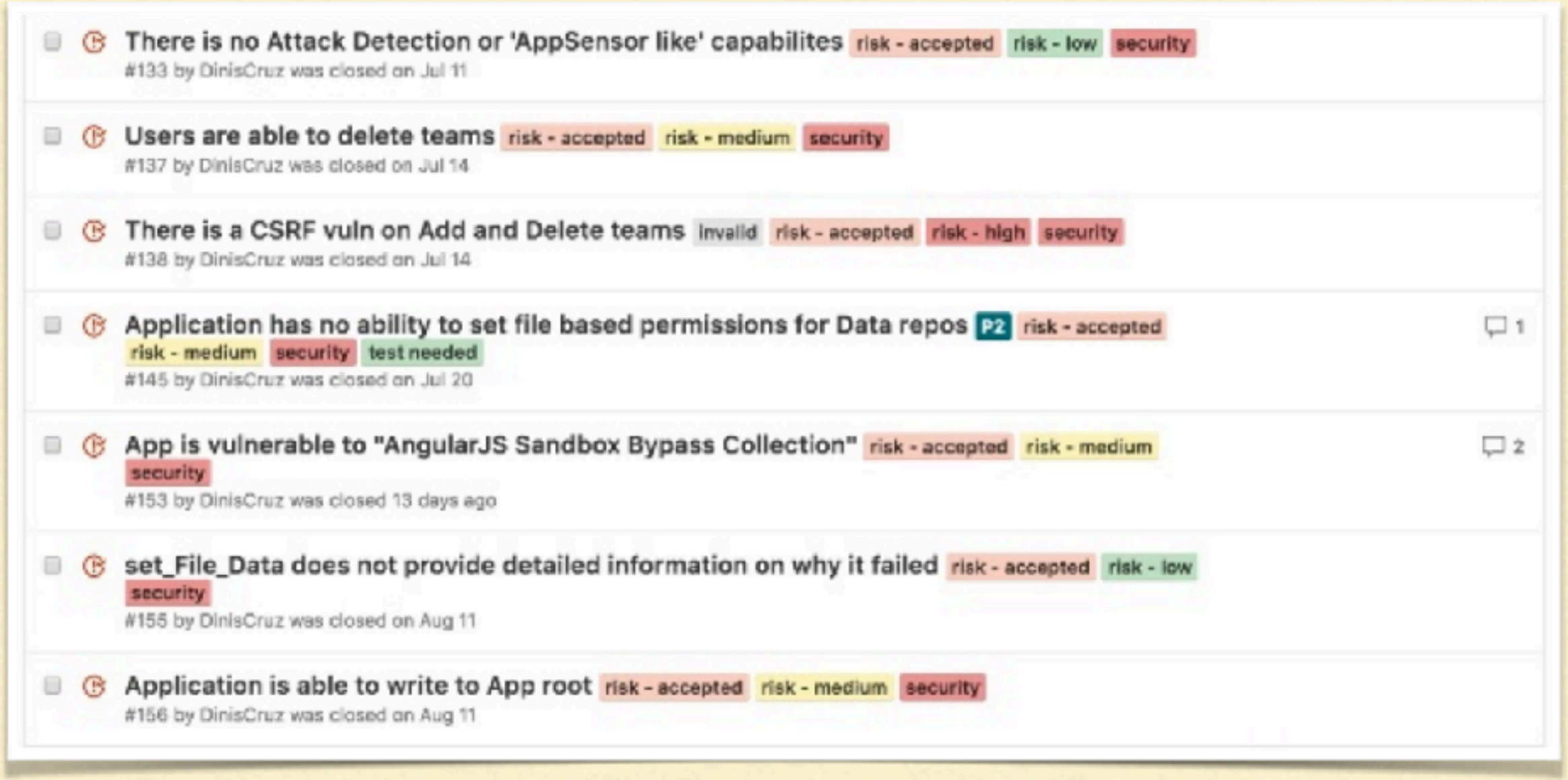
About
Informed news analysis every weekday

Node.js is the latest security risk for developers

Node.js isn't especially risky, but its popularity means sloppy coding can cause harm in a new venue

With Node.js having become a critical cog at places such as PayPal and Wal-Mart, developers need to be mindful of securing their Node.js applications, technologists are advising.

The battle for Node.js security has only begun



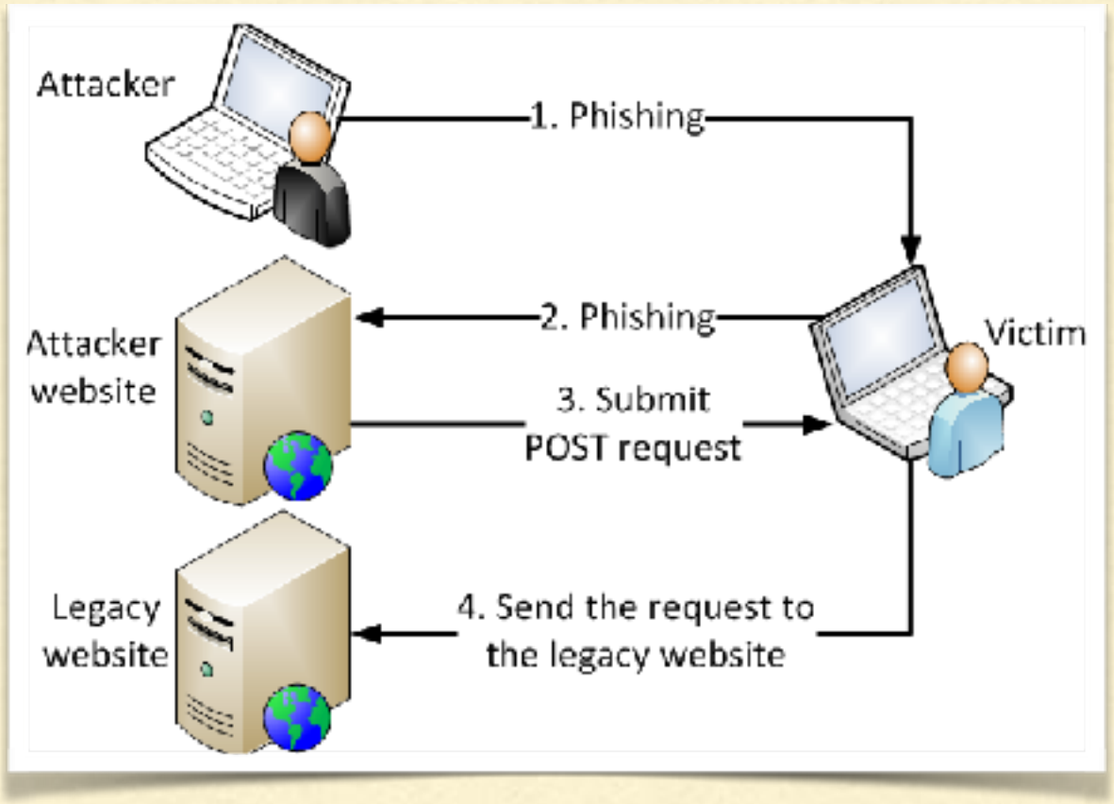
Growing security risk in OWASP

LET'S BOOM

CONCEPT OF CSRF

- CSRF（Cross-site request forgery跨站请求伪造，也被称为 One Click Attack 或者 Session Riding

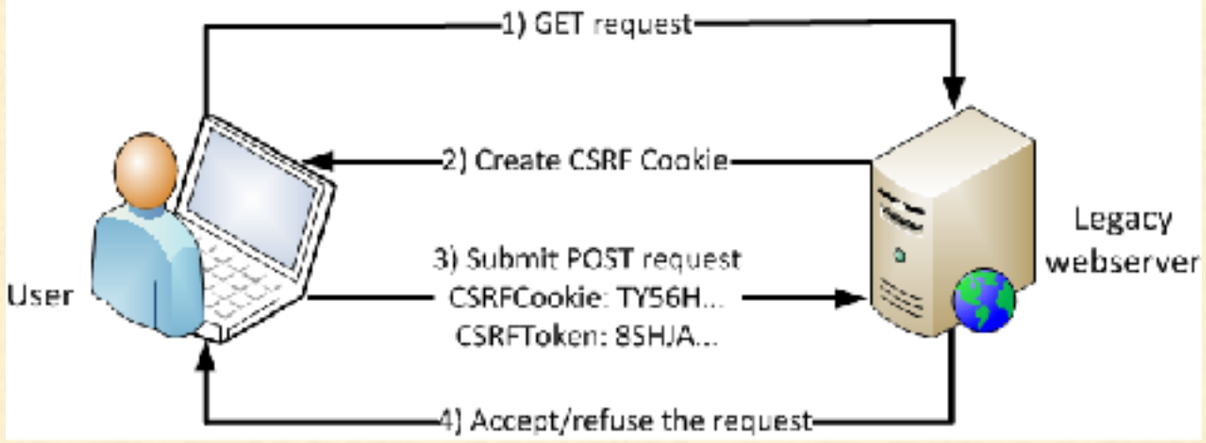
- 用户不知情的情况下执行了攻击者伪造的请求



RISK OF CSRF

- 强刷购物车、强刷收藏夹、强刷关注、强刷团购
- 发送垃圾购买信息、回复帖子
- 修改密码、找回密码
- 转账、付款

HOW TO PREVENT CSRF

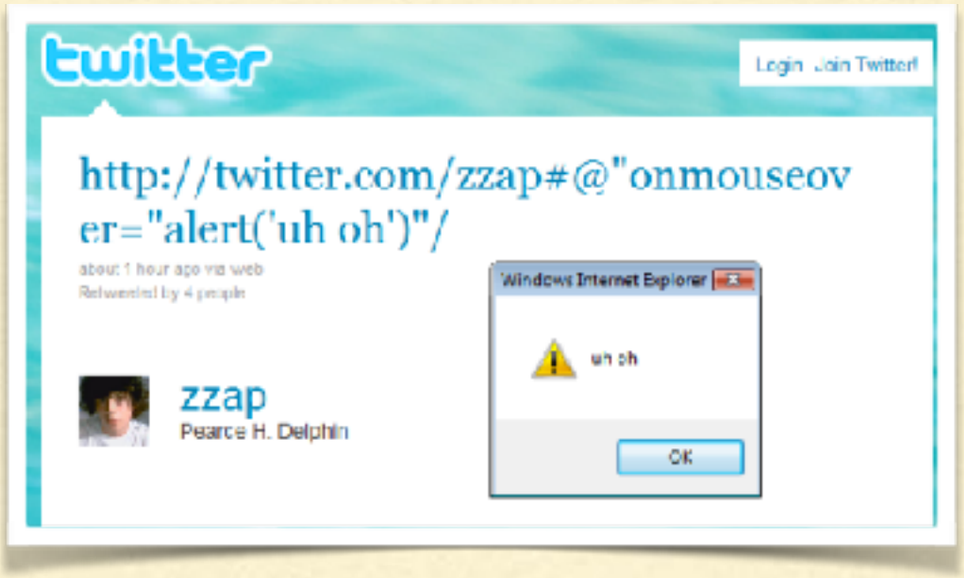


- 基本思路：Token验证
- 方式一：服务端强token验证
- 方式二：Double cookie submit
- 方式三：Custom Header
- 其他：验证码

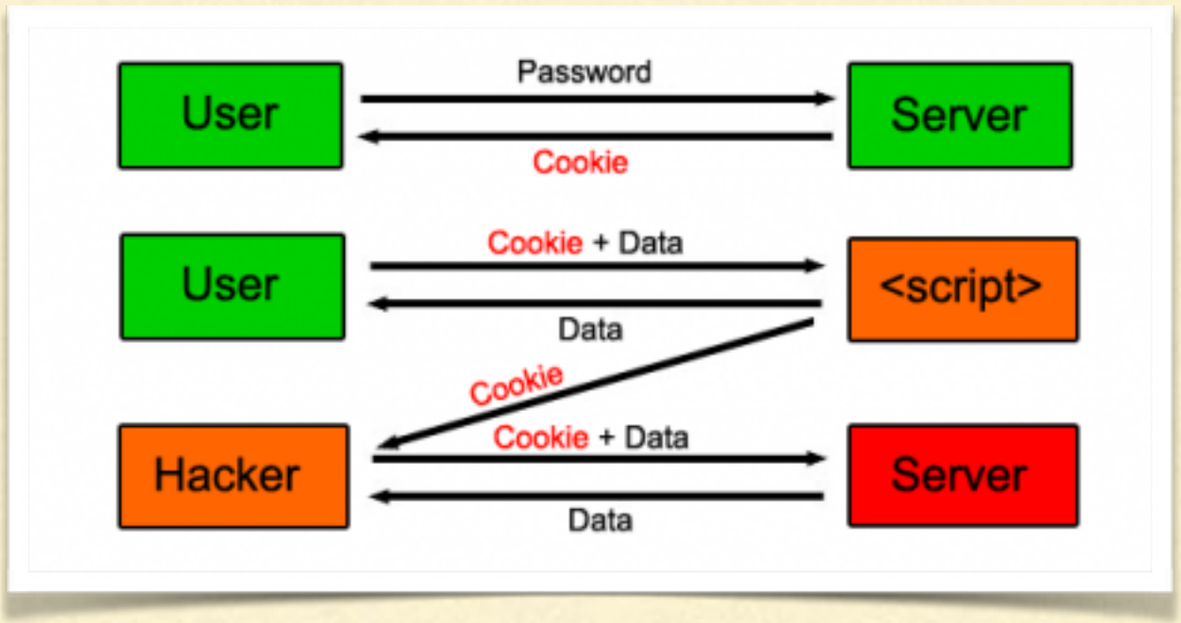
https://www.owasp.org/index.php/Cross-Site_Request_Forgery_%28CSRF%29_Prevention_Cheat_Sheet#CSRF_Specific_Defense

CONCEPT OF XSS

- 反射型XSS,由于服务端接收到客户端的不安全输入，在客户端触发执行从而发起 Web 攻击。



- 存储型XSS,通过提交带有恶意脚本的内容存储在服务器上，当其他人看到这些内容时发起 Web 攻击



AVOID XSS- HTML BODY

Encode for HTML Body

```
<a href=# ><img src=# onerror=alert(1) /></a>
<div> Untrusted Data </div>
```

&	→ &	"	→ "
<	→ <	'	→ '
>	→ >	/	→ /

AVOID XSS- HTML ATTRIBUTES

Encode for HTML Attributes

"><script>alert(/xss/)</script><"

```
<input type="text" name="firstname" value="Untrusted Data">
```

Non-alphanumeric characters → &#xHH; format

Enclose attribute value in quotes

AVOID XSS- CSS

Encode for CSS

```
body {background-image:url("JavaScript:alert('XSS')");}
```

```
<div style="width= Untrusted Data ;">contents</div>
```

Untrusted data → CSS Hex Encoding (\HH or \HHHHHH)

AVOID XSS- URL

Encode for URL

href="**javascript**:alert(1);"

```
<a href=" Untrusted Data ">Show Details</a>
```

Untrusted data → encodeURI()

AVOID XSS- SERVER&CLIENT

DOM Based XSS: Encode on both server and client

```
<a href="/reviews#[Untrusted Data]">Movie Reviews</a>
```

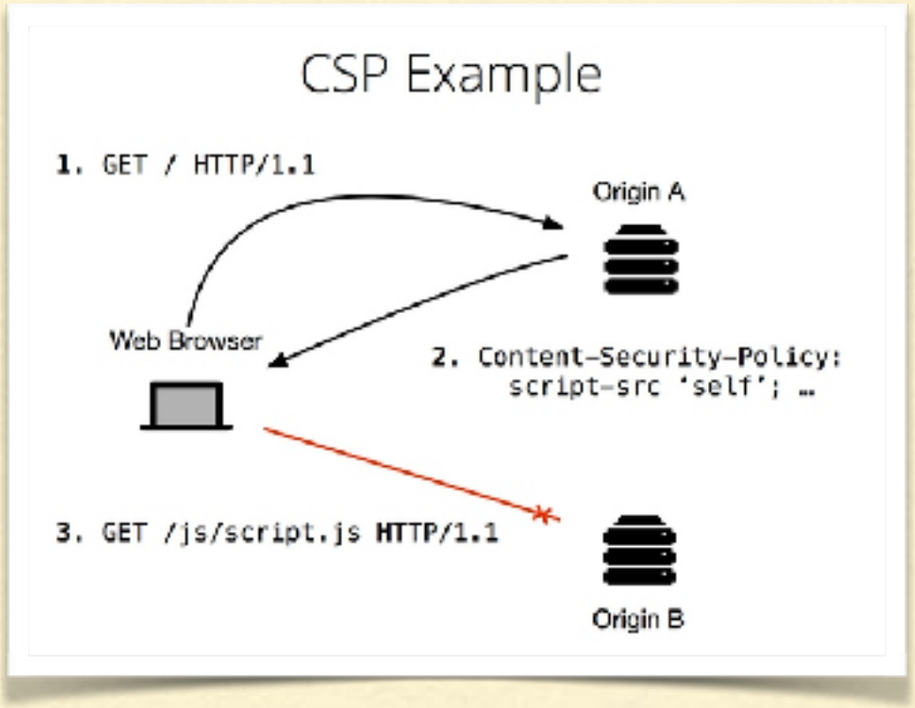
```
<script>
 document.write("<h1>"+ document.location.hash +"</h1>");
</script>
```

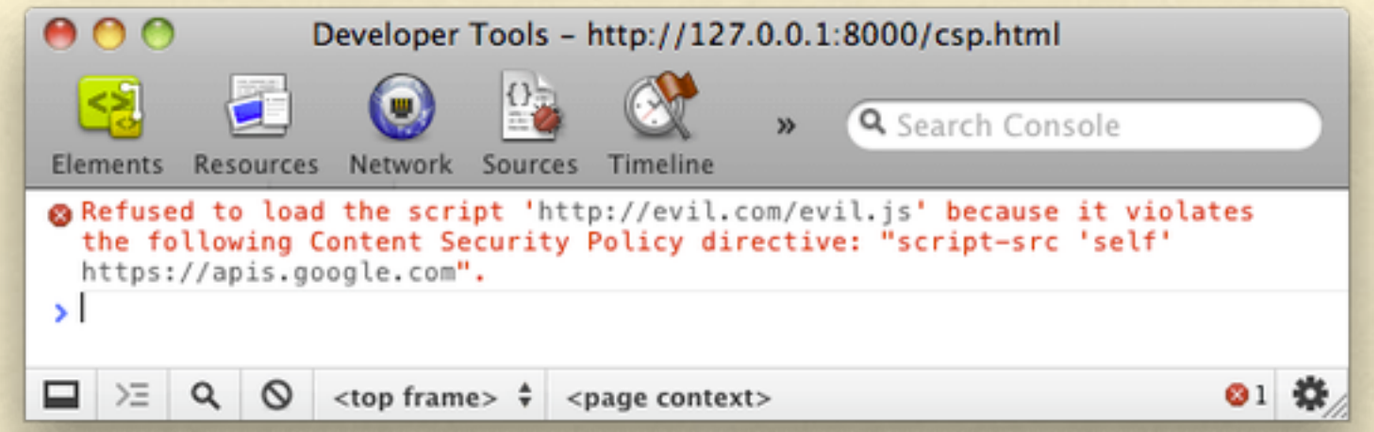
OVERVIEW PREVENTION OF XSS

- 不要相信用户任何的输入，过滤之
- 使用http only的cookie
- 使用web 安全头
- 新技术要慎重使用

CONCEPT OF CSP

- csp1.1
- 资源加载的限制
- url限制
- http与https
- report-uri
- unsafe-inline





https://report-uri.io/home/generate

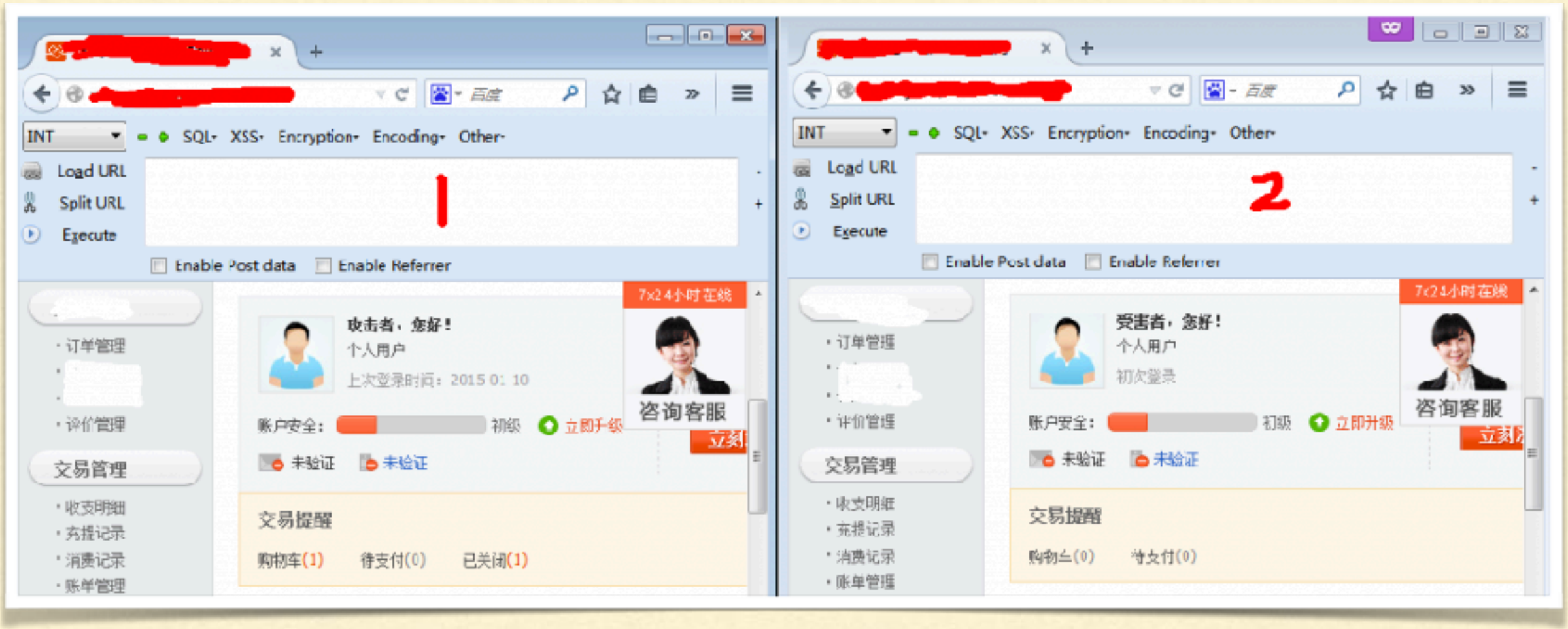
CONCEPT OF CSP 2.0 NONCE

Content-security-policy: ‘nonce-aaaa';

```
<!DOCTYPE html>
<html>
<title>Hello Strapdown</title>


<script>document.write("11111111<br>");</script>

<script nonce="aaaa">document.write("22222222<br>");</script>

<script nonce="bbbb">document.write("33333333<br>");</script>


<img src="http://aaa.aliimg.com" onerror="alert(1)" nonce="aaaa">

</html>
```

```
1  <!DOCTYPE html>
2  <html>
3  <  Step out of current function (⇧F11).
4
5
6  <script>document.write("11111111<br>");</script>
     ⊗ Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self'
     http://*.aliimg.com 'unsafe-inline' 'nonce-aaaa'". Note that 'unsafe-inline' is ignored if either a hash or nonce value is
     present in the source list.
7
8  <script nonce="aaaa">document.write("22222222<br>");</script>
9
10 <script nonce="bbbb">document.write("33333333<br>");</script>
     ⊗ Refused to execute inline script because it violates the following Content Security Policy directive: "script-src 'self'
     http://*.aliimg.com 'unsafe-inline' 'nonce-aaaa'". Note that 'unsafe-inline' is ignored if either a hash or nonce value is
     present in the source list.
11
12
13 <img src="http://aaa.aliimg.com" onerror="alert(1)" nonce="aaaa">
     ⊗ Failed to load resource: net::ERR_NAME_NOT_RESOLVED
     ⊗ Refused to execute inline event handler because it violates the following Content Security Policy directive: "script-src
     'self' http://*.aliimg.com 'unsafe-inline' 'nonce-aaaa'". Note that 'unsafe-inline' is ignored if either a hash or nonce
     value is present in the source list.
14
15 </html>
16
```

BROKEN ACCESS CONTROL



- 水平越权与垂直越权

ALL ABOUT ID：RISK OF ID

- 水平越权重灾区：各种管理系统 学校/图书馆/电商/论坛

- http://xxx/bad?user_id=3 && http://xxx/good?token=36423c633d685645232ac3ee7c0d77bc

- 分布式 + 防爬：唯一、尽量短、无规则、不可遍历 https://www.zhihu.com/question/20180484

- session/token 一定要过期

HOW TO PREVENT

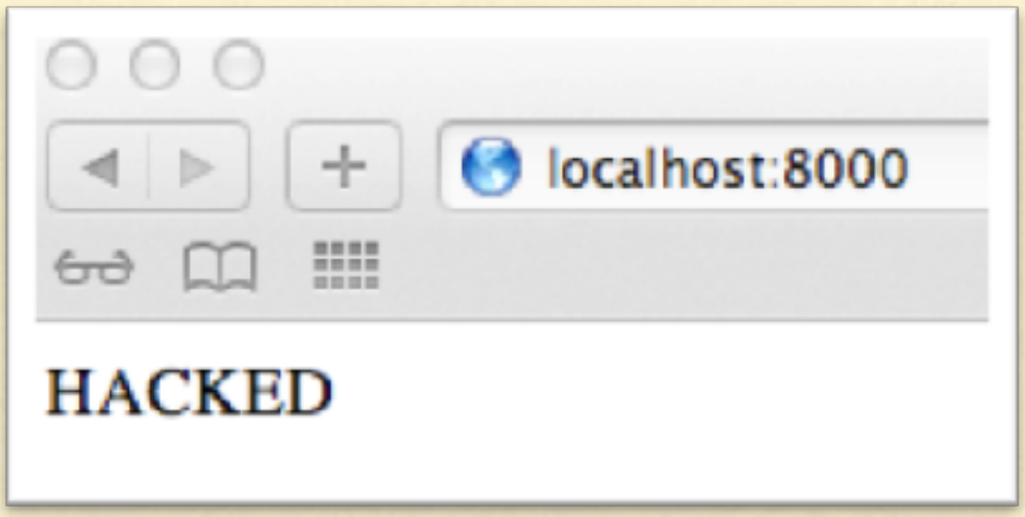
- 每个页面都要做好访问控制
- id 要取的艺术：唯一、尽量短、无规则、不可遍历
- 不要用一些危险的url： /admin. /manage
- 权限要存在生命周期，记得销毁
- 不要把debug版本带上线，比如 _debug，_logs
- 干掉所有的超级管理员

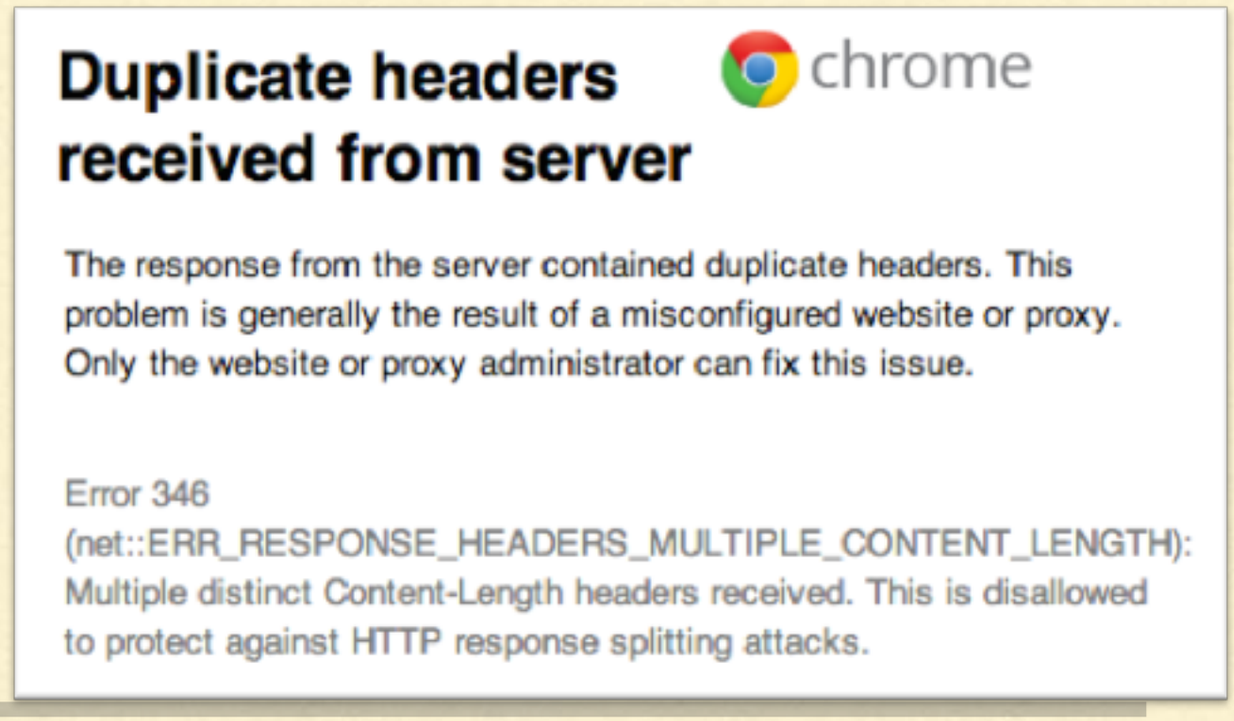
CONCEPT OF HTTP RESPONSE SPLITTING

```
var http = require('http');

http.createServer(function (req, res) {
  res.writeHead(200, { 'Content-Length': '0\r\n\r\nHTTP/1.1
  200 OK\r\nContent-Type: text/html\r\nContent-Length:
  19\r\n\r\n<html>HACKED</html>' });
  res.end();
}).listen(8000, '127.0.0.1');
```

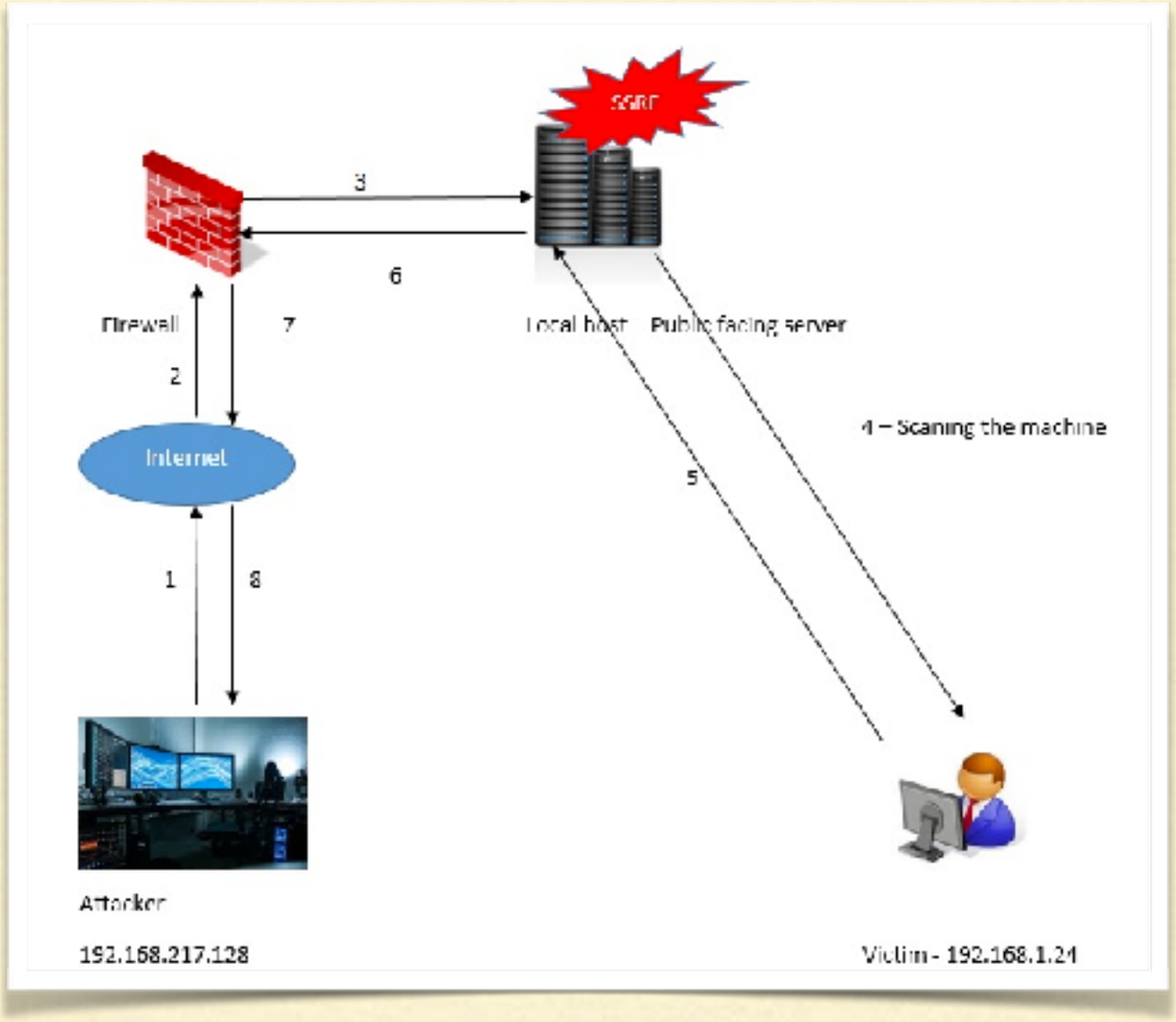
CONCEPT OF HTTP RESPONSE SPLITTING

- CRLF可以允许多段http response
- 某些浏览器有防范，但是可以绕过
- 造成XSS和钓鱼存储型XSS,通过提交带有恶意脚本的内容存储在服务器上，当其他人看到这些内容时发起Web 攻击
- 攻击场景：接收用户的参数并重定向，写cookie，操作header





CONCEPT OF SERVER SIDE REQUEST FORGERY (SSRF)



http://niiconsulting.com/checkmate/2015/04/server-side-request-forgery-ssrf/

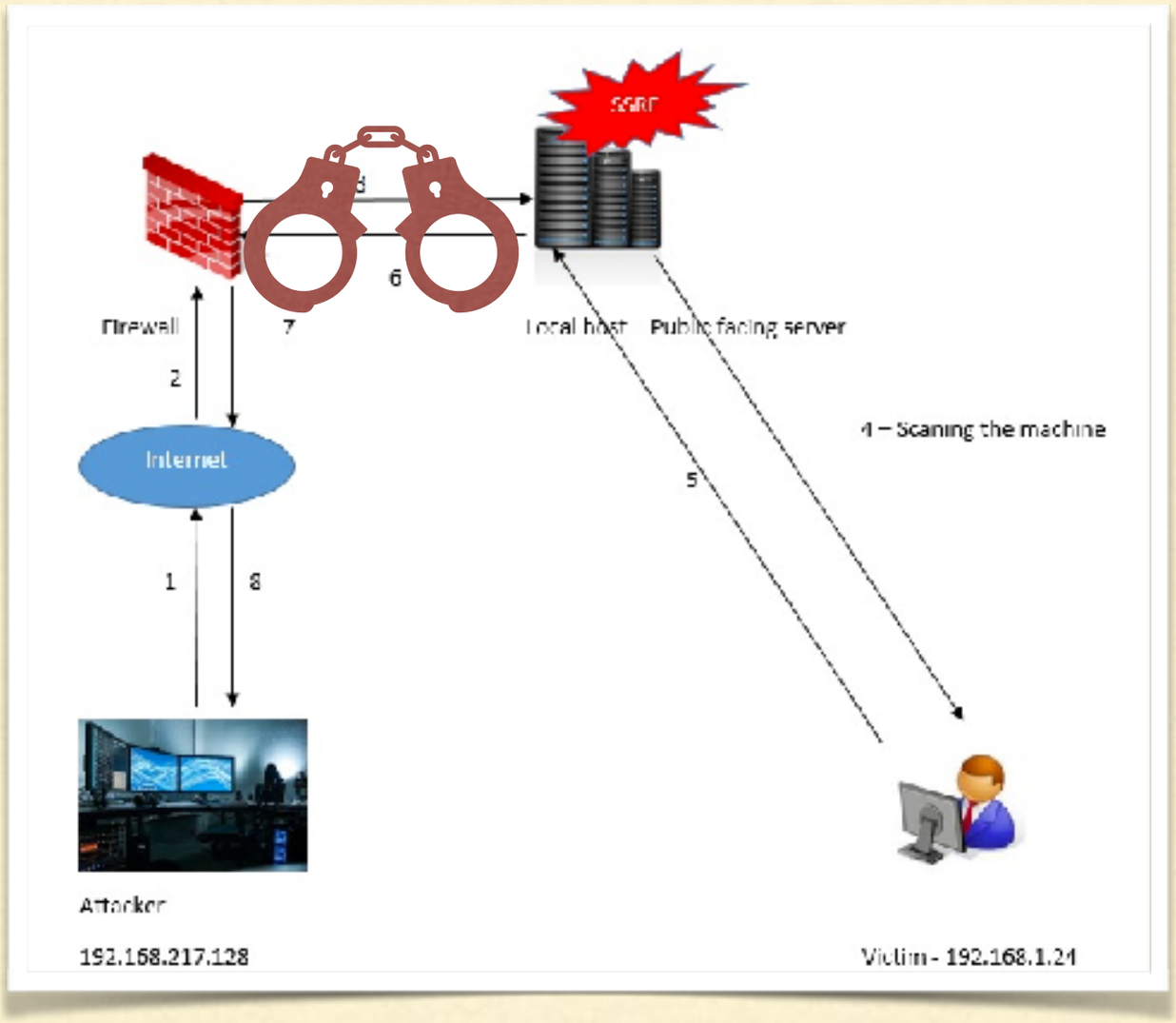
WHEN WILL CAUSE SSRF?

- 从用户指定的url获取图片，保存后展示给用户
- 获取用户制定url的数据（文件或者html），使用socket跟服务器建立tcp连接
- 根据用户提供的URL，抓取web站点，并且自动生成XX站
- 测速功能，根据用户提供的URL访问目标获取访问速度

EXAMPLE OF SSRF

```
'use strict';
const request = require('koa-request');
const midway = require('midway');
const logger = midway.getLogger();
// 透传第三方页面
exports.page = function* (){
  let pageUrl = this.query.url;
  if(!pageUrl){
    this.body = 'page url required';
    return;
  }
  try{
    let response = yield request(decodeURIComponent(pageUrl));
    this.body = response.body;
  }catch(e){
    this.render('error');
    logger.error(e.message);
  }
};
```

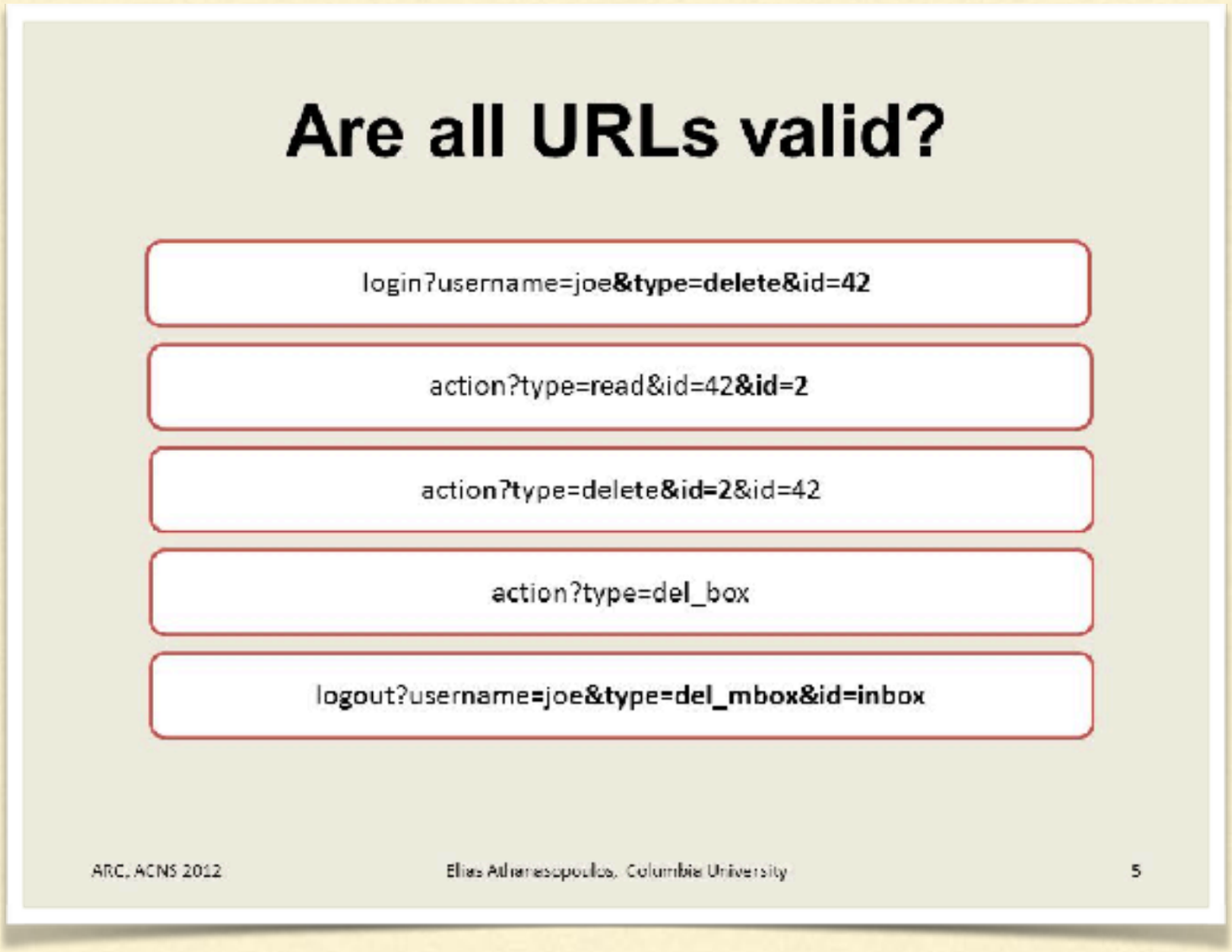
HOW TO PREVENT SSRF



- 基本思路：识别危险跳转
- 危险跳转： 内网域名
- 危险跳转： ip
- 危险跳转： xip.io
- 直接获取url对应的ip，域名则返回对应ip，然后再对ip进行判断是否为内网ip，即可防御SSRF内网探测

https://www.secpulse.com/archives/30876.html

CONCEPT OF HTTP PARAMETER POLLUTION(HPP)



RISK OF HPP

```
// POST firstname=John&firstname=John

req.body.firstname

//=> ["John", "John"]
```

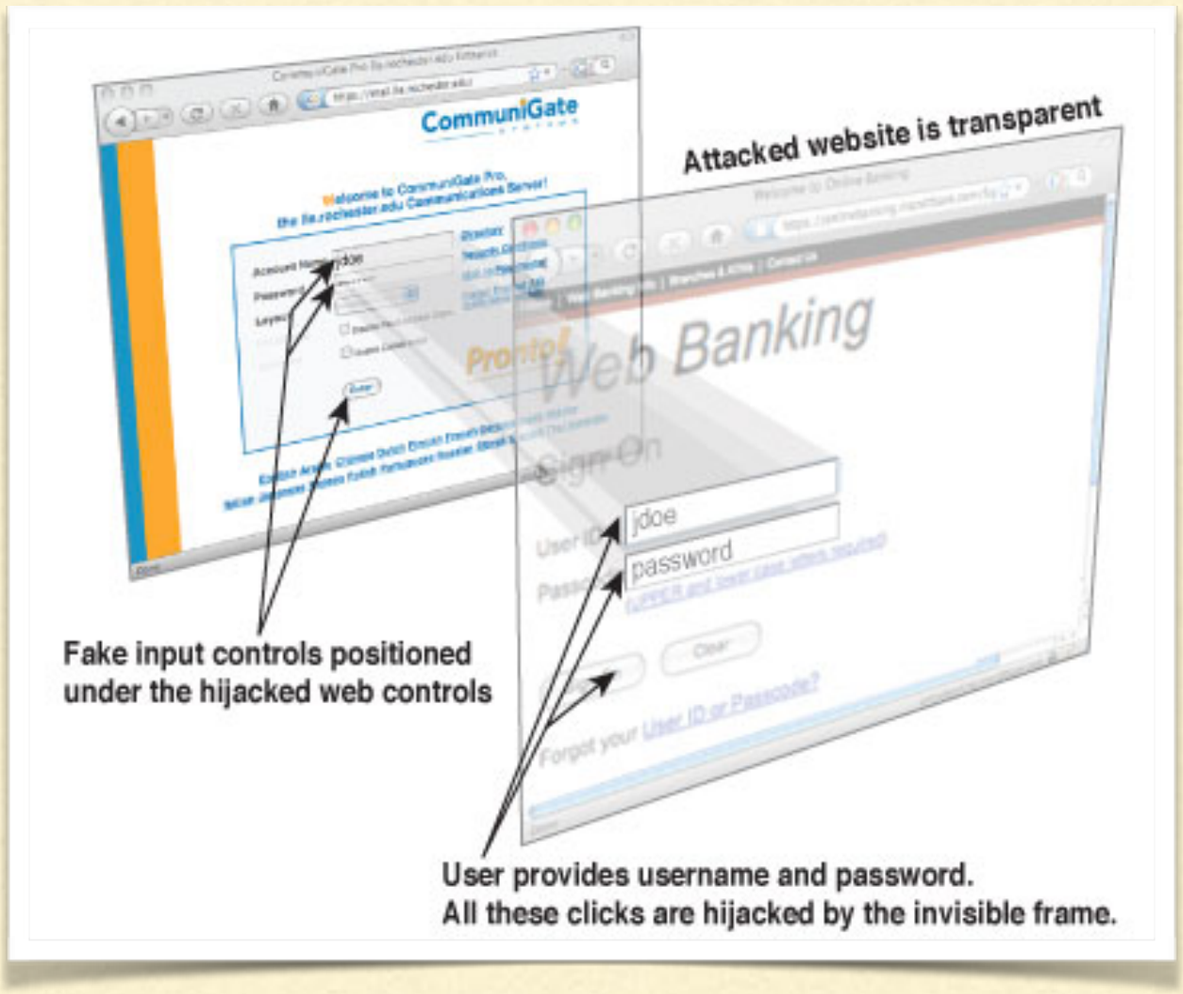
```
Object John,John has no method 'trim'
TypeError: Object John,John has no method 'trim'
```

```
> ["John", "John"] + " Doe"
"John,John Doe"
```

```
> db.users.find({userName:"p"}).pretty()
{
        "_id" : ObjectId("53092b495ad132dfd56dbf70")
        "address" : "",
        "dob" : "",
        "firstName" : [
                "John",
                "John"
        ],
        "lastName" : "Doe",
        "password" : "$2a$10$iZBDC45NulBco7s2GhCkJ.j
        "ssn" : "1234",
        "userId" : 39,
        "userName" : "p"
}
```

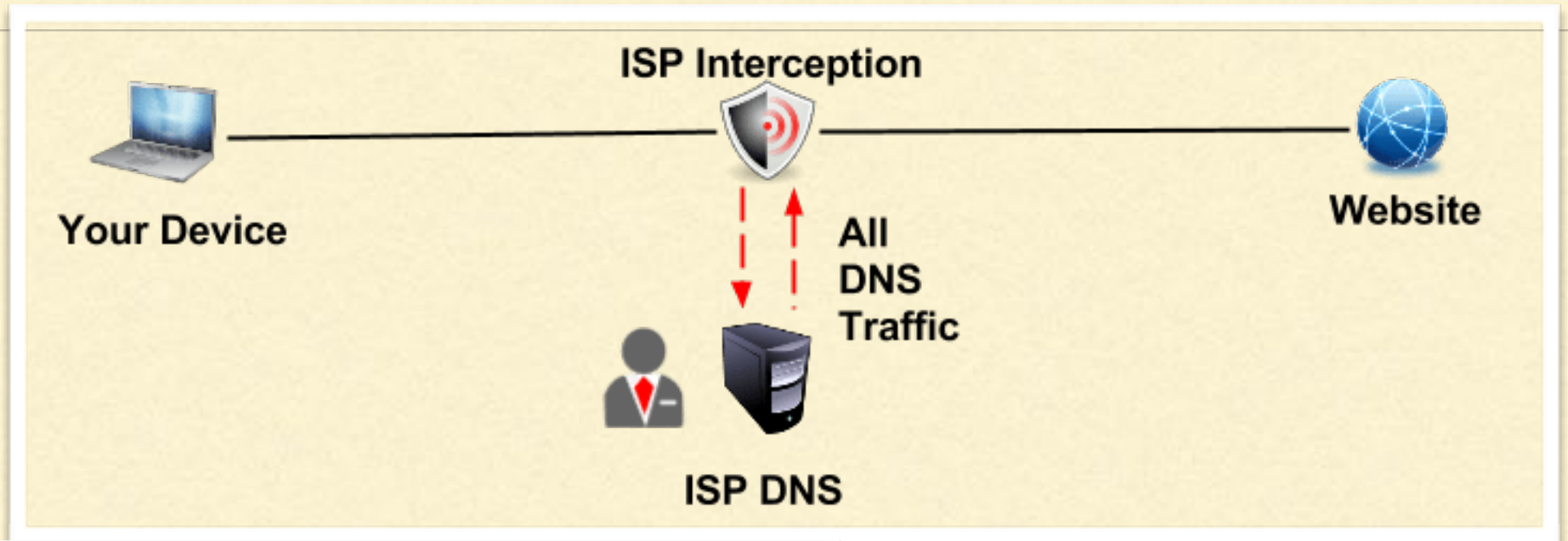
CONCEPT OF UNSAFE REDIRECT

如何钓鱼?



- this.redirect(url)
-
- iframe 钓鱼(click jacking)
- 使用 x-frame-options

ISP HIJACKING



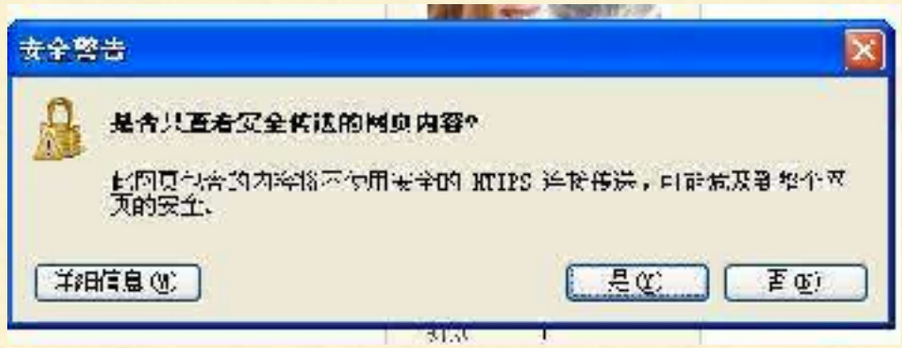
HTTPS


淘宝网 - 淘！我喜欢
https://www.taobao.com
www.taobao.com
您与此网站建立的是私密连接。
权限
连接
消息 1
手机逛淘宝
“镇店之宝


HTTP
HTTP
TCP
IP
网络接口
应用层
传输层
网络层
数据链路层
HTTPS
HTTP
TSL or SSL
TCP
IP
网络接口
应用层
安全层
传输层
网络层
数据链路层

CONCEPT OF MIXED CONTENT

- 去schema = 》http://xx/xx.jpg = > //xx/xx.jpg
- Mixed Content： IE / Modern W3C : Optionally-blockable & Blockable
- optionally: img, video, prefetched



- Content-Security-Policy: block-all-mixed-content

SMALL TIPS ABOUT HTTPS

- SRI- Subresource Integrity 鸡肋
- '<!--for injection--><!--</html>--><!--for injection-->'
- Strict-Transport-Security: max-age=36000000

PACKAGE RISK

Vulnerability DB

Subscribe to RSS feed
Report a new vulnerability

Detailed information and remediation guidance for known vulnerabilities.

SHOW: ○ All ○ Maven ● npm ○ pip ○ RubyGems

Vulnerability	Affects	Type	Published
H Denial of Service (DoS)	js-quantities <1.6.4	npm	02 Aug, 2017
H Out of Memory Crash	js-quantities *	npm	02 Aug, 2017
M Cross-site Scripting (XSS)	foundation-sites <6.0.0	npm	02 Aug, 2017
H Directory Traversal	xxf11 *	npm	02 Aug, 2017
H Malicious Package	tkinter <= 1.0.2	npm	02 Aug, 2017
H Directory Traversal	srverqq *	npm	02 Aug, 2017
H Malicious Package	sqlserver <= 1.0.2	npm	02 Aug, 2017

HOW TO GET SECURITY PACKAGE

NSP

- https://github.com/nodesecurity/nsp
- npm install -g nsp
- **Analyze package.json**
- nsp check --output summary

Node Security Project

Snyk continuously finds and fixes vulnerabilities in your dependencies.

Protect your JavaScript, Ruby, Java, Scala and Python apps

All checks have passed — Hide all checks
2 successful checks

✓ **Node Security** — No known vulnerabilities found — Details

✓ **continuous-integration/drone** — the build was successful — Details

This branch has no conflicts with the base branch
Merging can be performed automatically.

Merge pull request — You can also open this in GitHub Desktop or view command line instructions.

PROJECT DEPENDENCIES

54

NPM DEPENDENCIES

⚠ 1 module has a security vulnerability.

⚠ 22 modules are outdated.

OTHER SECURITY RISKS

CONCEPT AND PREVENTION OF XST

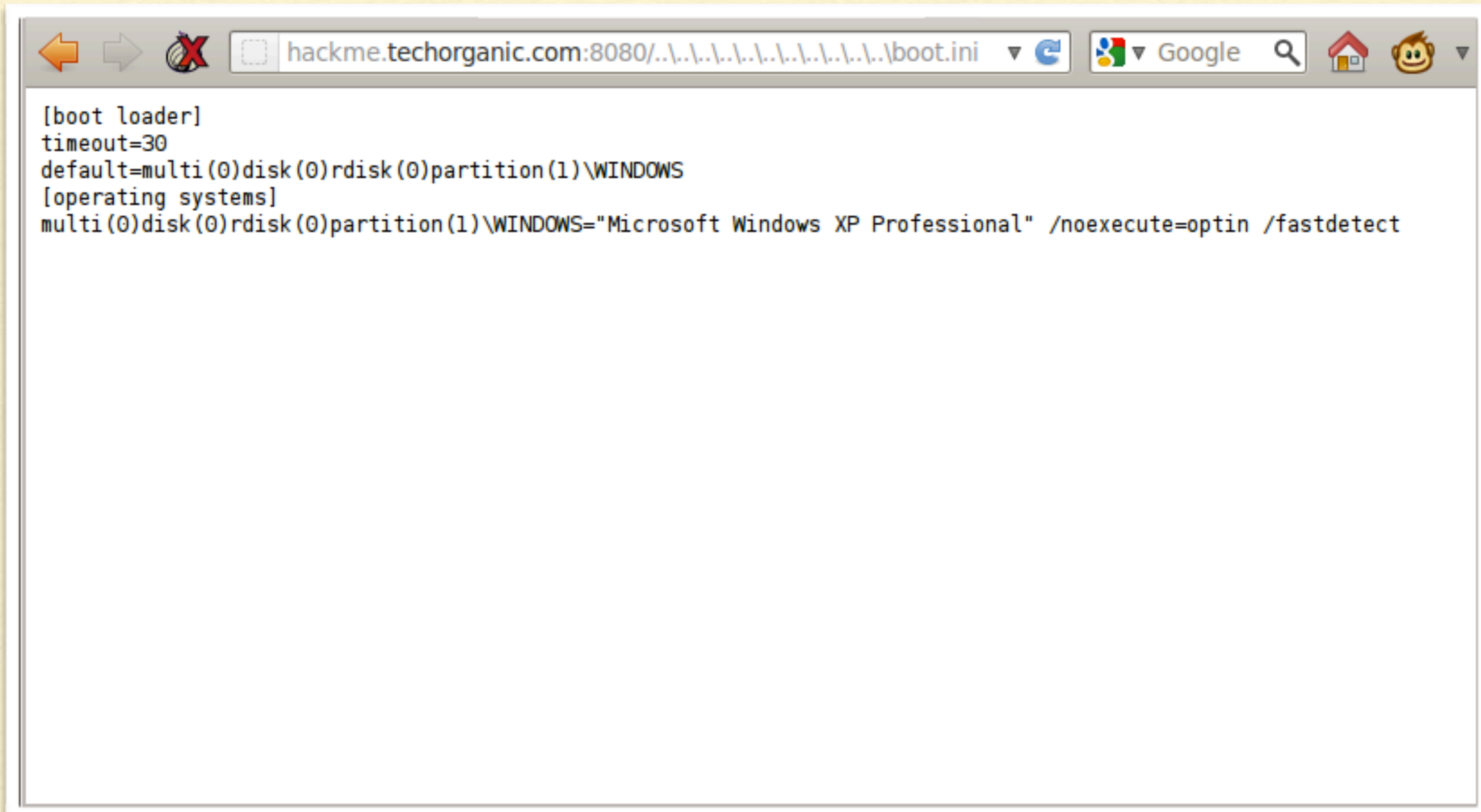
- Cross-Site Tracing，客户端发TRACE请求至服务器，如果服务器按照标准实现了TRACE响应，则在response body里会返回此次请求的完整头信息。通过这种方式，客户端可以获取某些敏感的头字段，例如httpOnly的cookie。

curl -X TRACE -b a=1 -i http://127.0.0.1:7001

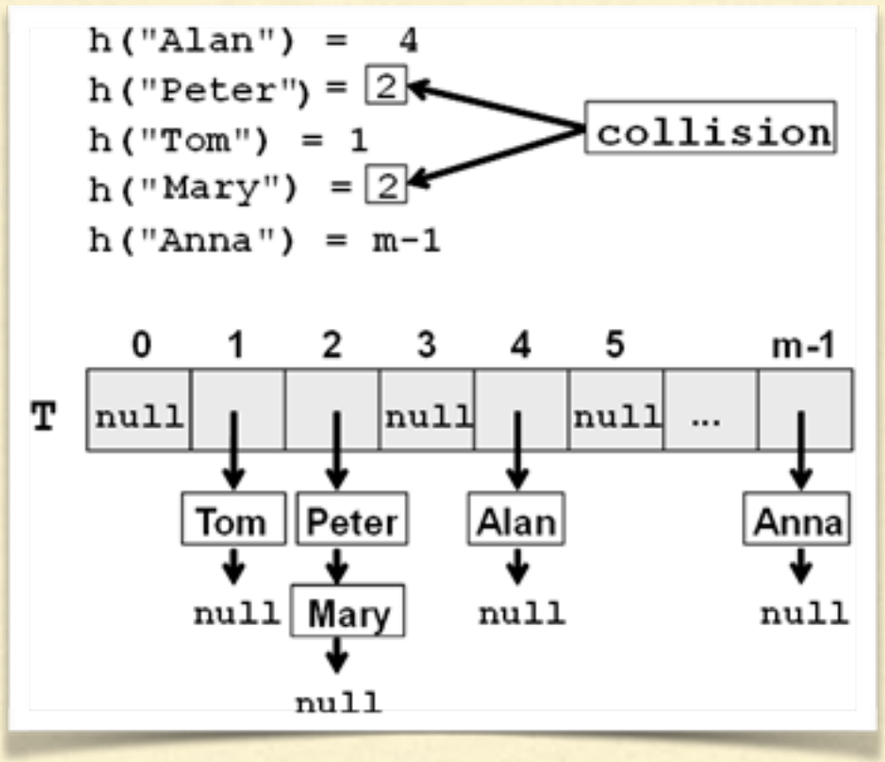
```
HTTP/1.1 200 OK
X-Powered-By: koa
Set-Cookie: a=1; path=/; httponly
Content-Type: text/plain; charset=utf-8
Content-Length: 73
Date: Thu, 06 Nov 2014 05:07:47 GMT
Connection: keep-alive
user-agent: curl/7.37.1
host: 127.0.0.1:7001
accept: */*
cookie: a=1
```

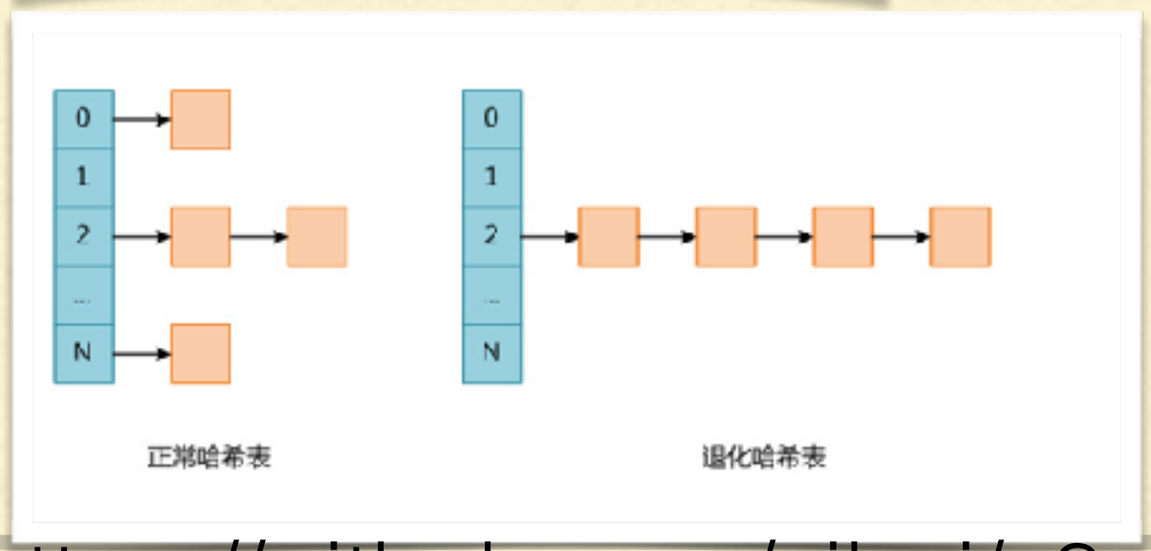
- 禁止trace track options 三种危险类型请求

DIRECTORY TRAVERSAL ATTACK



HASHTABLE COLLISIONS





- Insert:

```
i = h(x)
HT[i] = x
```

- Search:

```
i = h(x)
if (HT[i] == x) return true;
else return false;
```

https://github.com/xibei/v8-patch-for-hash-collision-leak

TIMING ATTACK

```
module.exports = function scmp(a, b) {
  a = String(a);
  b = String(b);
  if (a.length !== b.length) {
    return false;
  }
  var result = 0;
  for (var i = 0; i < a.length; ++i) {
    result |= a.charCodeAt(i) ^ b.charCodeAt(i);
  }
  return result === 0;
};
```

- https://codahale.com/a-lesson-in-timing-attacks/

B

- Binary planting
- Blind SQL Injection
- Blind XPath Injection
- Brute force attack
- Buffer overflow attack

C

- Cache Poisoning
- Cash Overflow
- Code Injection
- Command Injection
- Comment Injection Attack
- Content Security Policy
- Content Spoofing
- Cornucopia – Ecommerce Website Edition – Wiki Deck
- CORS OriginHeaderScrutiny
- CORS RequestPreflighScrutiny
- Credential stuffing
- Cross Frame Scripting
- Cross Site History Manipulation (XSHM)
- Cross Site Tracing
- Cross-Site Request Forgery (CSRF)
- Cross-site Scripting (XSS)
- Cross-User Defacement
- Cryptanalysis
- Custom Special Character Injection

D

- Denial of Service
- Direct Dynamic Code Evaluation ('Eval Injection')

E

- Execution After Redirect (EAR)

F

- Forced browsing
- Format string attack
- Full Path Disclosure
- Function Injection

H

- HTTP Response Splitting

I

- Inyección de Código
- Inyección SQL
- Inyección SQL Ciega
- Inyección XPath
- Inyección XPath Ciega

L

- Log Injection

M

- Man-in-the-browser attack
- Man-in-the-middle attack
- Mobile code: invoking untrusted mobile code
- Mobile code: non-final public field
- Mobile code: object hijack

O

- One-Click Attack
- OWASP Cornucopia

P

- Parameter Delimiter
- Path Traversal

R

- Reflected DOM Injection
- Regular expression Denial of Service – ReDoS
- Repudiation Attack
- Resource Injection

S

- Server-Side Includes (SSI) Injection
- Session fixation
- Session hijacking attack
- Session Prediction
- Setting Manipulation
- Special Element Injection
- Spyware
- SQL Injection

T

- Traffic flood
- Trojan Horse

U

- Unicode Encoding

W

- Web Parameter Tampering
- Windows ::DATA alternate data stream

X

- XPATH Injection
- XSRF


我真是日了狗了

安全无小事

EGG-SECURITY & IN ALIBABA

SECURITY HEADER SUPPORT

- X-XSS-Protection 默认开启
- -Content-Type-Options:nosniff 禁止嗅探内容并按照html渲染，默认开启
- X-Download-Options:noopen 默认开启
- Strict-Transport-Security (maxAge 一年 includeSubdomains 默认 false)
- X-Frame-Options 默认sameorigin
- content-security-policy
- csp2 nonce，支持模板自动种入script标签的nonce值，其他的script需要pe设置X-CSP-Nonce

FILTERS

- surl url转义与过滤，防范钓鱼攻击与XSS
- sjs javaScript转义，防范XSS
- sjson json转义，防范XSS
- shtml html转义，防范XSS
- escape 通用转义函数，防范XSS
- spath 防范目录遍历攻击
- clifilter 防范命令行注入

SERVER PREVENTION

- 完善的CSRF实现
- 禁止不安全的trace track options
- request.query参数类型固定，防止HPP攻击
- 提供安全的jsonp
- 覆盖框架的redirect方法，提供带有白名单的跳转

安全编码原则

- 不要关闭任何安全配置
- 不要相信任何用户输入
- 跟进最新的社区版本node包和nsp https://nodesecurity.io/advisories/，推荐你使用egg + egg-security

低成本安全防范的几个思路

XSS 拦截与上报

- 勾住危险代码 eval/setAttribute/document.cookie
- 覆盖危险方法 html()/redirect
- CSP report uri

```
{
  "csp-report": {
    "document-uri": "http://example.com/signup.html",
    "referrer": "",
    "blocked-uri": "http://example.com/css/style.css",
    "violated-directive": "style-src cdn.example.com",
    "original-policy": "default-src 'none'; style-src cdn.example.com; report-uri /_/csp-reports"
  }
}
```

- 危险动作上报： MutationEvent

线下与CI代码扫描

- 扫描安全配置是否关闭和开启
- 扫描模板是否进行转义 {{ }} => {{ escape()}}
- 生成抽象语法树，扫描特殊场景
- 对项目依赖进行nsp扫描
- …

REFERENCE

- https://eggjs.org/zh-cn/core/security.html
- https://www.owasp.org/index.php/Category:Attack
- WebGoat
- egg-security

THANK YOU!