DevSecOps: Strategies to Achieve Security by Design

Deliver modern applications securely with minimal impact to the speed of the DevOps lifecycle.

Bottom Line Up Front

Frictionless Security • Fewer Bugs • Speed to Market

Everyone agrees that security is a top priority, yet few teams have integrated security into the development lifecycle. A common concern is that security creates roadblocks that impede the engineering process.

But what if teams could integrate security without slowing down development?

In this whitepaper we will explore ways for teams to achieve frictionless integration of security into the existing DevOps lifecycle. At each phase, we will identify security activities that teams should consider implementing to fuse security and the Software Development Lifecyle (SDLC).

The goal of each recommendation is to reinforce the technical, cultural, and business benefits of DevOps while ensuring adequate security is integrated without handicapping market release.

Contents

Background

Building a DevSecOps Toolchain

Why This Matters

The business problems modern technology organizations face are simple:

- Organizations must deliver products and features to the market quickly.
- Your customers demand solutions that are seamlessly scalable with minimal downtime.
- Security is a must-have to be trusted in the marketplace.

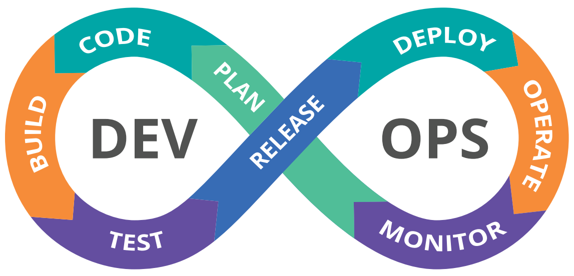
The DevOps model has emerged to meet the challenges of the first two bullet points, but security has not always been a given. For DevOps teams to effectively ensure the security of their platforms, they must integrate best practices into day-to-day process with minimal friction.

Let’s explore some options teams can take to improve their security posture while not hindering the Continuous Integration and Continuous Delivery or Deployment (CI/CD) pipeline.

Giving Context to the DevSecOps Lifecycle

The DevOps lifecycle typically consists of eight phases. These phases are intended to flow into one another and create a just-in-time style of agile software development. DevOps is focuses on removing friction, risk, and other constraints to enable faster, more successful delivery of software.

Our first step in integrating security into the DevOps lifecycle is to ensure we do not violate any of these core tenets.



The DevOps Lifecycle

When considering ways to integrate security into DevOps, we will take guidance from SANS and approach the DevOps lifecycle from five higher level phases they call “the DevSecOps lifecycle” to aid in identifying opportunities for security integration into the process.

Our goal at each phase is to identify different strategies available to teams to integrate security into their workflows.

The goal is not to prescribe a one-size-fits-all approach, but instead allow teams to make decisions based on risk tolerance, resources, and culture. This is the optimal process for achieving **security by design.**

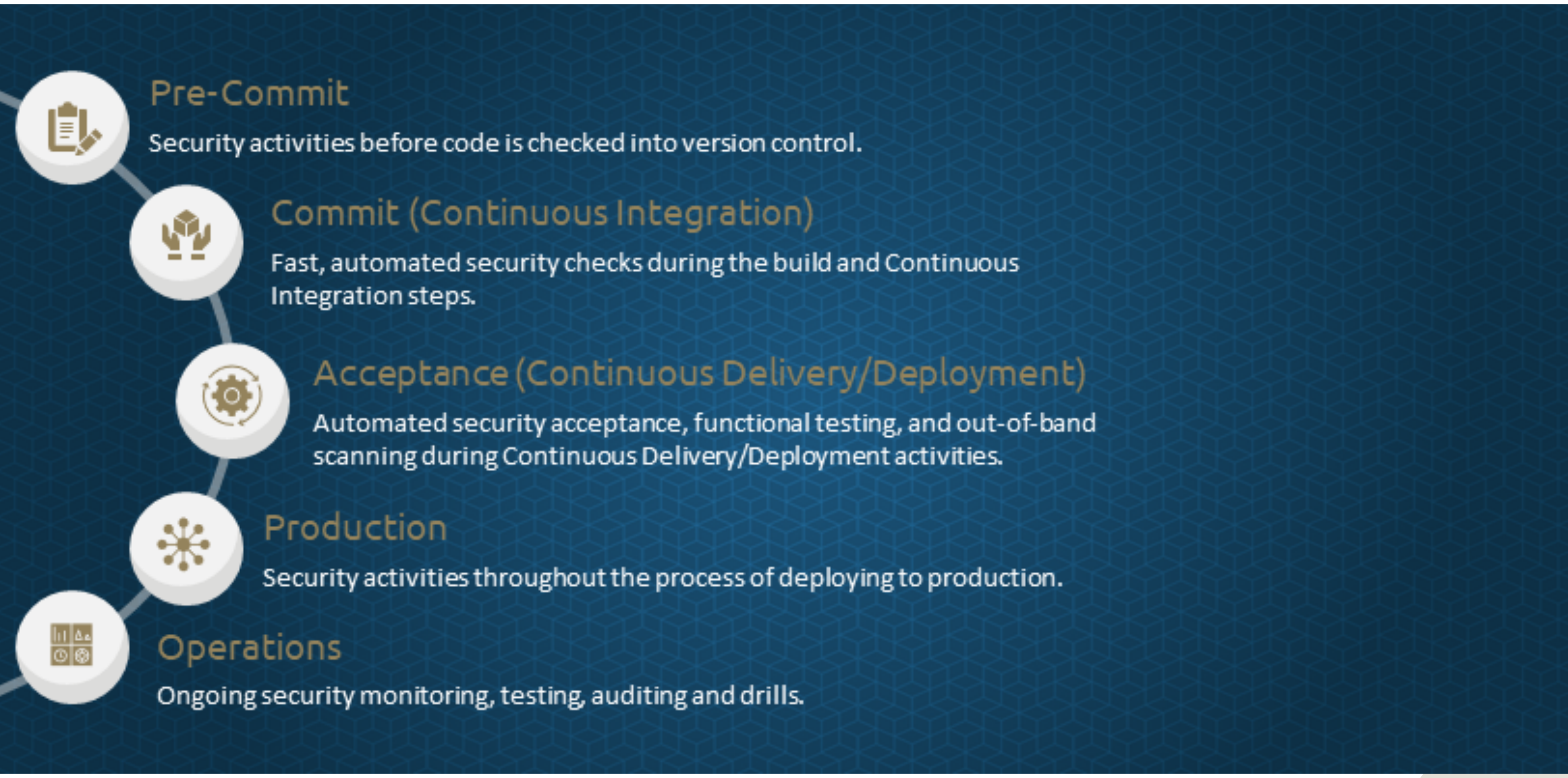
The DevSecOps phases are as follows:

- Pre-Commit
- Commit (Continuous Integration)
- Acceptance (Continuous Delivery/Deployment)
- Production
- Operations

How This Article is Structured

This article is structured to explore each DevSecOps phase and present a series of security integration options for that phase. In cases where solutions are referenced, open-source options will be highlighted over commercial products.

At the end of the article there will be an appendix with suggested tools and resources for teams to consider.



Pre-Commit

Security activities before code is checked into version control.

Commit (Continuous Integration)

Fast, automated security checks during the build and Continuous Integration steps.

Acceptance (Continuous Delivery/Deployment)

Automated security acceptance, functional testing, and out-of-band scanning during Continuous Delivery/Deployment activities.

Production

Security activities throughout the process of deploying to production.

Operations

Ongoing security monitoring, testing, auditing and drills.

Pre-Commit Phase

Security Activities Before Code is Ever Committed to Source Control

The Pre-Commit phase is focused on identifying security activities and controls that can be applied during project planning, coding, developer training and developer onboarding. Before code authorship even begins, security should be integrated.

Secure Coding Training and Standards

Secure code authorship is the foundation of application security. To groom a team that develops secure code, leadership needs to be unified on the training and standards it holds its developers to.

Three Steps to Enforcing Secure Coding Practices:

1. *Implement Secure Coding Training*: Implement reputable training that your development team enjoys and requires interaction (no surface-level OWASP slide decks).

2. *Develop Secure Coding Standards*: Establish rules and guidelines all developers must follow to prevent security vulnerabilities.

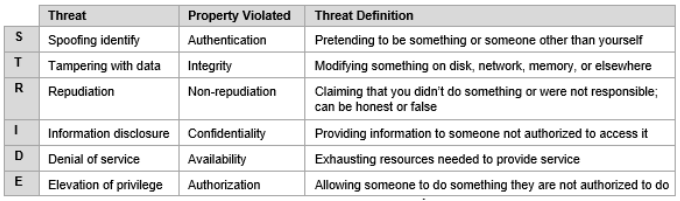
3. *Develop KPIs to Track Secure Coding Performance*: Adopt a set of KPIs to hold developers accountable. The Magic Numbers Web App Security Key Performance Indicators (KPIs) are a great place to start.[1]

[1] https://www.slideshare.net/RafalLos/magic-numbers-5-kpis-for-measuring-ssa-program-success-v132

Threat Modeling

Threat modeling consists of a detailed risk assessment of an application's architecture and inner workings to identify structural vulnerabilities, required safeguards, and controls.

Threat modeling should consider both the technical and business context of the organization (i.e. attacker stories, and the classification of data being processed). It is smart to follow an established threat modelling method such as STRIDE to guide exercises. Many threat modeling exercises may result in application diagrams with data flow.



	Threat	Property Violated	Threat Definition
S	Spoofing identify	Authentication	Pretending to be something or someone other than yourself
T	Tampering with data	Integrity	Modifying something on disk, network, memory, or elsewhere
R	Repudiation	Non-repudiation	Claiming that you didn't do something or were not responsible; can be honest or false
I	Information disclosure	Confidentiality	Providing information to someone not authorized to access it
D	Denial of service	Availability	Exhausting resources needed to provide service
E	Elevation of privilege	Authorization	Allowing someone to do something they are not authorized to do

STRIDE Threat Model

Threat modeling exercises should evolve with projects (e.g. when new features or changes in architecture are introduced). Threat modeling exercises should result in a list of risks and mitigation strategies.

IDE Security Plugins and Linters

Integrated Development Environment (IDE) plugins, also called Linters, can help reinforce secure coding practices and identify well-documented coding errors before the developer can check their code into source control.

IDE security plugins and Linters help teams retain their agility by saving the developer from having the switch back and forth between the coding environment (i.e. the IDE) and standalone dynamic and static code analysis tools.

IDE security plugins and Linters should also be considered for operations teams automating infrastructure deployments using Infrastructure as Code (IaC).

Pre-Commit Security Hooks

Pre-commit hooks are scripts that are automatically executed by the source control utility (e.g. Git) prior to allowing code to be committed to the source code repository.

Security task-specific pre-commit hooks are useful in preventing sensitive information such as passwords, cryptographic keys, and other secrets from being pushed to source control.

Peer-Reviewing of Code

Peer-reviewing of code involves a manual and independent review of source code before being merged into a master source code repository or branch.

Though these activities are manual, they are an important mechanism to maintain segregation of duties in the continuous deployment process.

The peer-reviewing process should have documented guidelines and objectives that include both security and quality control measures. Teams may also choose to document developer key performance indicators (KPIs).

Development KPIs include:

- Defect Rate: Average number of bugs found per review.
- Defect Density: Average number of bugs found per lines of code.

Commit Phase

Security Checks and Controls as Part of Continuous Integration

In the Commit Phase, the goal of the development team is to merge their changes back to the main code branch as seamlessly as possible. The objective at this phase is to implement fast, automated security checks that are completed during the building and testing phases of the DevOps lifecycle.

All security mechanisms introduced in the Commit Phase should emphasize testing automation of code before it is integrated into the main branch.

Static Code Analysis

Static Code Analysis is an automated process of scanning source code for bugs and syntax issues prior to running the program.

There are dozens of static code analysis tools available on the market. When considering a tool, make sure it fits the needs and style of your team to increase likelihood of adoption.

Key Attributes to Consider in a Static Code Analysis Tool:

- Tailored rulesets to turn off rules of little or no value to software developers.
- Efficient management of false positives (e.g. in-code suppression).
- Fast and easy to deploy. A good tool should need less than five minutes to run a scan.

Do not rely too much on static code analysis tools. Many issues related to dynamic interactions with the application, like authentication and access control, will probably not be found during static code analysis.

Security Unit Tests

Security unit testing is meant to expand upon typical unit testing and integrate security checks into the process.

In unit testing, individual modules of an application are tested in isolation to confirm the code is processing as expected and perform checks for application issues before they are moved along in the code promotion process.

Security unit tests expand unit testing to perform checks for syntax and conditions associated with common attacks and security flaws.

Testing languages such as JUnit, xUnit, and Mocha allow for the automation of thousands of tests to ensure software functionality. They can be expanded to include security checks for input validation, encoding of outputs, and ensuring access to sensitive resources is restricted to appropriate user contexts.

Container Hardening and Security

Containers are lightweight, standalone packages of software that can run consistently across any infrastructure capable of running the Container Host (e.g. Docker). This evolution in infrastructure management creates a new attack surface for engineering teams to protect. Container security should be considered prior to production deployment.

While many engineering teams tout the security benefits of containerized infrastructure such as process isolation or more efficient configuration management, other security concerns exist.

Container Security Concerns to Consider:

- Distribution of poisoned or outdated images.
- Challenges managing container secrets (i.e. avoid building them into images or passing them as runtime variables).
- Protection of the Docker daemon and management APIs.
- Ensuring the host OS is sufficiently hardened and protected.

Two important considerations for the DevOps team include ensuring containers images are properly hardened and implementing utilities to monitor them while in operation.

Container Hardening and Monitoring

Perhaps the most effective approach to hardening containers is through generation of AppArmor profiles. AppArmor is a Linux kernel security module that allows for the whitelisting of commands strictly necessary for operation and the disabling of writing and reading to directories not normally used by the applications deployed on a container.

An AppArmor profile generator is a tool that will automate the creation of these profiles.

AppArmor is a mandatory access control (MAC) like security system for Linux. It is designed to be easy to use and deploy and allow an admin to restrict access for specific processes

Teams should also consult the CIS Benchmarks and implement suggested features into their configuration files. Commercial solutions like GRSecurity also provide managed patching that simplify container hardening.

Once containers are operational, it is vital to monitor them for vulnerabilities and suspicious behavior just as you would traditional infrastructure.

Container security utilities typically perform analysis based on of the following:

- Auditing containers for Common Vulnerabilities and Exposures (CVEs) and other potential vulnerabilities (e.g. Clair, kube-hunter).
- Auditing containers against the CIS Benchmarks or organization developed security polices (e.g. Docker Bench, Grafaes).
- Behavior monitoring for both network and internal system calls (e.g. Sysdig Falco).
- File integrity monitoring to identify unauthorized changes to sensitive files.

Infrastructure as Code – Code Analysis

For teams that have adopted an Infrastructure as Code (IaC) approach to infrastructure deployment, integration of IaC Analysis tools into the Change Management lifecycle is a must.

Common configuration issues IaC security utilities look for:

- Use of overly permissive Identity Access rules (i.e. use of wildcards)
- Use of overly permissive security group rules (i.e. use of wildcards)
- Identification of access logs that are not enabled.
- Identification of encryption not being enabled.
- Identification of hardcoded secrets, keys, and passwords.

Dependency and Open-Source Software (OSS) Analysis

Virtually all modern applications rely on third-party libraries. Mitigating the risk of vulnerabilities existing in third-party libraries (i.e. dependencies) in use by your team is an important and easy step.

Teams can mitigate risk of using vulnerable third-party libraries through the use of dependency checkers. Gartner also refers to these tools as Security Composition Analysis (SCA) tools.

Most dependency checker solutions work by running the list of dependencies in your project against NVD public vulnerability databases and producing a report of associated issues or updates to be applied.

Some projects, like OWASP's Dependency-Check and GitHub's Security Alerts, allow for seamless integration into your source control and deployment engine utilities for automation within the Dev/Ops lifecycle.

“

Enterprises should use SCA tools on a regular basis to audit repositories that contain software assets (such as version control and configuration management systems) to ensure that the software developed and/or used by the enterprise meets security and legal standards, rules and regulations.

— Mark Horvath, Hype Cycle For Application Security 2018, Gartner

Gartner

Acceptance Phase

Security Within Continuous Delivery and Deployment

Within the Acceptance phase, security integration steps should focus on functional testing and out-of-band scanning. At this stage of the process, we see teams diverge between Continuous Delivery and Continuous Deployment styles of delivery.

Continuous Delivery is focused on making sure the DevOps team is ready to release changes quickly, sustainably, and generally on a set cadence.

Continuous Deployment is focused on achieving a just-in-time (JIT) system of delivering changes either directly to production, or a staging/QA environment that turns over to production very frequently (sometimes multiple times a day).

Regardless of the frequency or style of delivery, many of the same security considerations should apply.

Infrastructure as Code – Security Configuration Concerns

Infrastructure as Code (IaC) allows DevOps teams to provision and manage infrastructure through source code.

Many teams start with templates when working with IaC. While templates provide a great starting point for DevOps teams, they present a great risk to organizations if not properly configured.

Palo Alto Networks published research detailing the most common vulnerabilities found in IaC templates.[2] These findings can be used to create a list of items for teams to consider when starting to customize their IaC templates.

The Most Common Infrastructure-as-Code Template Security Issues:

- Encryption is not enabled on all cloud resources (e.g. S3 and RDS in AWS).
- Logging is not enabled and configured.
- Sensitive resources are unintentionally exposed publicly (e.g. databases, storage, non-web facing app servers, cloud resource snapshots/backups).
- Sensitive ports are unintentionally exposed publicly.
- Ingress and egress traffic for virtual networks has not been configured based on business needs (i.e. left wide open).
- Security groups/local firewall rules on cloud compute instances are configured to be overly permissive.
- Containers (in a containerized environment) are configured to run with root or privileged access by default.

“Automation of the infrastructure deployment process increases the importance of security and compliance testing, as with using IaC, with the push of a button, you are able to make highly impactful changes to your cloud environment. . . In the public cloud, where simple configuration changes can leave sensitive data and private servers exposed to the world, the security implications of automation are profound.

— Roy Feintuch, Co-founder and CTO, Dome9 Security

[2] https://unit42.paloaltonetworks.com/cloud-threat-report-intro/

Out-of-Band Security Scanning and Compliance Checks

The most non-invasive way for security teams to ensure the secure configuration of infrastructure is to perform independent, out-of-band scanning.

Scanning takes a variety of forms. Teams may opt to run vulnerability, port, cloud configuration, and/or web application security scanners to verify IaC templates have been tailored to the organization's needs.

Other specialized tools like Serverspec can be used to perform security audits against a server or container's actual state. Tools like Scout2, Hubble, and InSpec are also useful for performing configuration audits against standards like CIS.

Immutable Infrastructure

Teams are also quickly adopting the concept of Immutable Infrastructure, which can have security benefits.

In this approach, infrastructure is implemented as read-only. Configuration changes and updates are not allowed in production. Instead, all upgrades, patches, and changes happen in source code, base images, and dependencies, which are then redeployed.

For teams that have properly externalized data (e.g. are not storing production data or static configurations on the local drive of the compute instance), this approach is worth considering as it forces changes through the various security checks and balances of the Dev/Ops SDLC.

Production Phase

Security Checks on Code in Production

In the Production Phase we are concerned with security integrations and checks to code that have already been deployed to production.

Secrets Management Utilities

Failure to properly manage secrets within applications impacts both the scalability of your application and its security.

Many teams are guilty of hardcoding secrets (e.g. API keys, cryptographic keys) directly into web configurations, IaC configurations, source code, and so on.

Through use of a secrets manager, you can provide an application with a secure way to call secrets as they need them while also putting access controls around secrets.

This also allows for the development of more scalable applications thanks to secrets being centrally located. Teams also benefit from simplified key rotation and resetting.

We have decided to address secrets management at the end of the SDLC, but realistically, it could also be considered in the Pre-Commit phase.

Serverless Infrastructure Protection

Serverless infrastructure adoption is still growing. One benefit of using serverless infrastructure is that a lot of the security burden is passed on to the cloud platform provider.

Still, some security concerns persist, including unintended outbound internet connectivity from the serverless infrastructure, accidental exposure of temporary directories where sensitive information might be cached, and source code leakage.

Tools like FunctionShield and PrismaCloud provide prebuilt functions that help secure serverless solutions like AWS Lambda, Azure Functions and GCP Serverless solutions.

Configuration Safety Checks/Monitoring

In the Acceptance Phase, we recommended scanning tools to perform configuration spot checks prior to full deployment of source code. Once an application has gone live, certain tools can provide continuous monitoring and reporting on configurations.

Many services, including AWS Config and Trusted Advisor or Azure Advisor, all have a cost associated with them. But teams may also opt to implement open source and free solutions such as OSQuery and Netflix's Security Monkey to perform similar monitoring.

Host Intrusion Detection Systems (HIDS)

The Host Intrusion Detection Systems (HIDS) space is crowded with solutions and options. This is no surprise as HIDS are high-impact solutions that satisfy requirements mandated by many popular compliance frameworks.

HIDS can be a valuable solution when paired with an effective incident response process. Multiple open-source, free options exist including OSSEC, Wazuh, and Security Onion.

Many of these solutions can also be expanded to include threat hunting capabilities. Beware that the learning curve can be steep for effective deployment and management. For organizations seeking out low-touch deployments, commercial solutions might be a better option.

Operations Phase

Continuous Security Monitoring for Deployed Applications

In this final phase, we focus on security monitoring and testing of the production or near production environment. Most of the prior security integration ideas involved direct integration of security tools and processes into the Development and Quality Assurance processes.

This final phase is heavily focused on processes owned by security and Site Reliability Engineering teams.

Configuration & Vulnerability Scanning and Compliance Checks

Virtually all teams are running vulnerability scans. There are many populat solutions available for this, but configuration scans are much less prevalent, despite being just as important.

Configuration scanning utilities like Scout2, Prowler, or Aqua's Cloudsploit do not necessarily reveal vulnerabilities out of the box but do offer something just as valuable: a look into potential security gaps that often go unseen by vulnerability scanners.

Key Benefits of infrastructure Configuration Scanners:

- Enabling management to verify that application configurations meet security design requirements.
- Assisting with threat modeling and finding your weaknesses before your adversaries.
- Enforcing accountability of DevOps teams.
- Simplifying compliance demonstration with security framework and control requirements.
- Naturally passive, non-invasive, and do not reply on DevOps teams to pull results once set up.

Continuous Monitoring

Logs are of limited usefulness if monitoring and alerting is not in place. Each major cloud platform has a flagship monitoring solution (e.g. Azure Security Center, AWS CloudWatch). This means implementation of this control is a no-brainer.

Some teams opt to go even further and create highly tailored monitoring and alerting solutions using tools like ElastAlert, SOF-ELK, and Grafana.

Best-in-class teams will identify specific configurations that should never be changed and monitor for unexpected changes. Some teams may even opt to deploy canary files and tokens. These approaches pair nicely with immutable infrastructure.

Fault Injection

Fault injection is the process of introducing unexpected behavior to a system and studying how it reacts to stress.

For teams wishing to test infrastructure resiliency and their response handling processes, Netflix's Chaos Monkey and Chaos Kong both test AWS infrastructure resiliency. And for teams wishing to simulate active cyber-attacks, Infection Monkey can get the job done.

Cyber Event Drills and Incident Postmortems

The final suggested element of the DevSecOps toolchain is one that is of utmost importance but is also often neglected.

Gameday Exercises

Gameday exercises are mock drills (usually associated with disaster recovery) that can be used to train personnel in incident response and handling.

Effective exercises will be scenario-based and critique team response and coercion just as much as performance of technical controls.

Tabletop Exercises

Tabletop exercises differ from gameday exercises in that they do not involve active interaction with systems. These exercises are generally more focused on planning and role-playing. Effective exercises involve a cross-functional team from across the business.

Postmortems

Postmortems are meetings and/or reports on the conclusion of both planned exercises and unplanned events. Postmortems are important elements of the enterprise information risk management program. All findings in a postmortem should drive an enterprise risk register and be tracked through remediation. These exercises should never be considered optional for any team.

Let’s Get Started

Programs That Leave No Doubt

Risk3sixty is a nationally recognized security, privacy, and compliance advisory firm serving firms across the United States and globally.

We strive to be “craftsmen” in our space and as a result, we offer our clients an uncommon level of service demonstrably unchallenged in our industry.

By the Numbers:

- ✓ 100% of our Clients are References
- ✓ 100% Three-Year Client Retention
- ✓ Clients in across the United States and 17 countries
- ✓ Certified Security Experts such as CISSP, CISA, CISM, GPEN, CEH, CRISC, PCI QSA, ISO 27001 Lead Auditors, and many more
- ✓ Certified Privacy Experts such as CIPP/US, CIPM, IAPP Privacy Fellows, ISO 27701 Lead Auditors, and more
- ✓ Contributions to the security community such as monthly “Capture the Flag” events, open-source security toolsets, and industry research

Our Promise of Quality:

We pride ourselves in our ability to provide outstanding service, meeting our clients’ deadlines, and exceeding expectations. This is why we are proud of our 100% three-year client retention and that 100% of our clients are references. The bottom line is that if you aren’t satisfied with the quality of our services, we’ll make it right. Period.

Speak With a Professional

Shane Peden
Director, Cyber Risk & CISO Advisory

CISA | CISSP | GPEN | PCI ASV
HITRUST CSSFT | MCSE

https://www.linkedin.com/in/speden/
404.519.8877