SOFTWARE SECURED

The Short Guide to Application Security For SaaS Companies

Table of Contents

About The Author

Sherif Koussa is the OWASP Ottawa Chapter Co-Leader, Software Developer, Hacker, and founder and CEO of Software Secured & Reshift Security. Sherif brings over 14 years of experience contributing to OWASP Ottawa, WebGoat, and OWASP Cheat Sheets. In addition, he supported the SANS and GIAC organizations in launching their GSSP-Java and GSSP-NET exams and contributed to several of their courses. After switching from software development to the field of security, Sherif took on the mission of supporting DevOps teams to build confidence in their application security.

Introduction

UNDERSTANDING PROACTIVE VS. REACTIVE SECURITY

For most, the hectic ride of a start-up or a scale-up does not provide the time to build systems and processes properly. The same can be considered for an application security program. As you juggle between building the product and supporting new clients, there is very little time to properly build a security program. Instead, you might find yourself running ad hoc and random security tasks just to support particular clients or certain compliance mandates. It's like '*Application Security Whack-A-Mole*.'

KEY SUBJECTS

Throughout this guide, we will cover the building blocks of typical application security programs and we'll identify at which stage of your growth they would be most useful. Planning ahead allows for a more strategic and thoughtful application security program with more favourable timeframes and budget requirements.

Here at Software Secured, we have collected the information in this guide through a decade of working with start-ups and scale-ups. We hope that the information provided will help to bring your company to the next level in application security to support your growth and maximize your exit. Please reach out if you have any questions.

Sherif Koussa

CEO

"Do what today others won’t, so tomorrow, you can do what others can’t."

Brian Rogers Loop

- CHAPTER 1 -

THE FOUNDATIONS

These are the essentials that **businesses of all sizes** should have in their cybersecurity strategy. They include security processes you can even include before you develop a single line of code.

- Training your developers on application security
- Choosing a security champion for your team

1.1 APPLICATION SECURITY TRAINING

Best for Seed companies.

Post-seed, team size grows beyond 6-10 developers. In this stage, it's important to make application security a clear priority for developers. Security can be integrated as they write code.

Your team can't fix security vulnerabilities if they don't know what to look for. Knowing the most prevalent bugs in application security programs is a top priority for any development team. Your team needs at least a well-rounded, base-level training in order to understand the top security bugs and how to find them. Training together is also helpful to ensure that everyone on the team has the same base level of knowledge so that they can work cohesively.

LEARN MORE

Types of Developer Application Security Training

1 General Security Awareness

This is a high level security training for employees. Usually, security awareness training focuses on general security hygiene such as understanding the basic concepts of social engineering and other phishing attacks. Many CTOs make the assumption that technical staff are better equipped to deal with phishing and social engineering than other non-technical staff. While there is some truth to that, the attack scene is changing so fast. Software developers are becoming the prime target for social engineering as they have a higher level of admin access to multiple systems.

2 Secure Coding Best Practices

While most developers are easily able to identify what good code looks like, they don't know how to identify secure code. Knowing the most prevalent bugs in application security programs should be a top priority for any development team. If your team doesn't already have this foundation, we recommend at least a well-rounded, base-level application security training for your developers in order to understand the top bugs and how to find them. A good place to start is through an OWASP Top 10 training, which teaches developers the most important techniques to write secure code in any language.

3 Security Operations Best Practices

Not all security breaches happen through a phishing email or a SQL Injection in the application. There are other scenarios that can lead to a security breach. For example, the lack of security operations best practices including password storage, internal system onboarding, off boarding, setting up cloud infrastructure in an insecure manner, building secure images, etc.

Taking a training course does not mean that employee behavior regarding security will change completely. Different people will react differently to the information they intake during the training. After providing employees with the necessary information via training, you need to start thinking about how to encourage behavior change following the course.

1.2 EMPOWER YOUR SECURITY CHAMPION(S)

In today's software development culture, there is an ever-increasing need for management to **drive empowerment within their teams.** You need to seek out, identify, and empower someone who can act as your team's security champion. Find at least one champion to start, and add more if they are available. As you grow, you may even consider assembling a **Security Champions team**.

Security champions should have some security background or knowledge of cyber security, as well as being willing, able, and motivated to learn much more. They should be interested in staying up-to-date about evolving developments in the field. Your champion can be a current team member or a qualified contractor/consultant. A thorough knowledge of the team's goals is necessary. A security champion needs to be a positive person that can offer diligent observations and constructive suggestions to the team.

Your security champion will promote the best practices in application security. They will work with your systems architecture and engineering teams, and with DevOps and DevSecOps project leaders. You may choose someone who has excelled in their Application Security Training courses, as an example. Their knowledge of security threats and remediation methods will be of great assistance in preventing and eliminating security problems earlier in the software development lifecycle.

Your security champion(s) will act as the reminder to be conscious about security in team meetings and design sessions.

MORE BENEFITS OF SECURITY CHAMPIONS

- CHAPTER 2 -

THE CASM FRAMEWORK & INTEGRATED ANALYSES

Application security isn't one-size-fits-all. You grow into it as your business scales, your revenue increases, or your development team gets larger. Along the way, you can look into adopting new types of integrated analyses like:

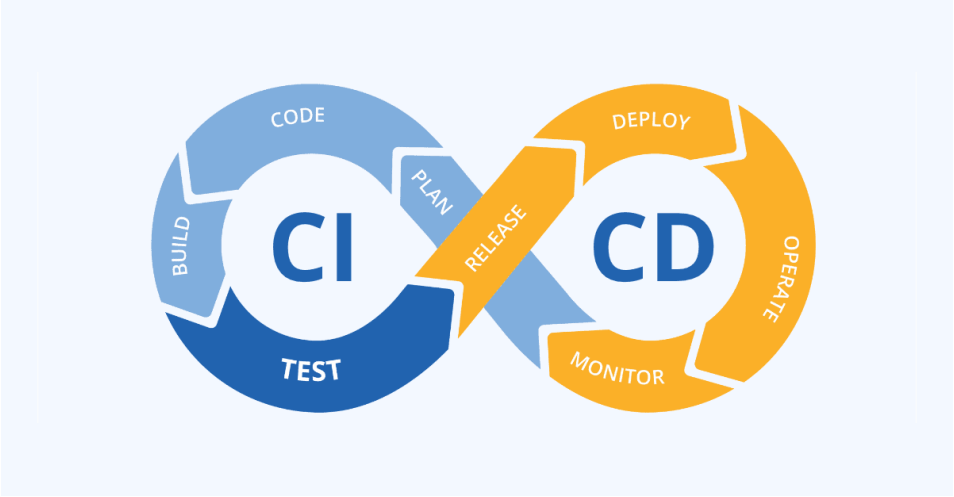
- Threat modeling & secure design decisions
- Automated SAST & DAST tools
- Dependency checking
- Penetration testing
- Web application firewalls

Careful: they may not all be necessary for your business, but they aren't all interchangeable either. Choose the ones that are right for your business function, size, and goals.

2.1 THE CASM FRAMEWORK

Some describe security as boring, a progress-inhibitor or necessary-evil. We see it as a competitive advantage. Take DevOps for example. Netflix was able to weather the storm of AWS blackout due to their superior DevOps practices. While other AWS users were offline, Netflix remained operational and pretty close to full capacity. Application security functions exactly the same. Improving security practices helps you compete better on many fronts and avoid any disruption due to cyber attacks.

- **Continuity.** A repeatable process has a distinct start and end point.
- **Automation.** Automatic processes eliminate the barrier to execution.
- **Scheduling.** This process cannot be forgotten under the distraction of tight deadlines
- **Measurability.** Developers must be able to draw insights and make decisions from the results of the analyses.



Source: TechAhead

Using the Continuous Integration approach as your CASM framework is a great choice as it provides you with each of the continuity, automation, scheduling and measurability factors.

If the CI/CD approach cannot be used, then a security code review process is a possible alternative. However, the code review process will likely provide limited results in comparison. Regardless, all reported or found bugs should be recorded and prioritized so that they can be dealt with during the next scheduled sprint backlog event.

The types of integrated analyses processes included in this section, when applied together, will form the bedrock of your application security.

2.2 INTEGRATING THREAT MODELING

Best for Pre-Seed companies.

With limited budgets, access to additional resources and commercial tools is likely unavailable. Threat modeling is the most economic activity to bake security into the SDLC. Additionally, many design decisions are made at the pre-seed stage. These decisions can be difficult to change in later stages, so it's important that they are considered for security through threat modeling.

Threat modeling is the ultimate shift left approach. It can be used to identify and eliminate potential vulnerabilities before a single line of code is written. Employing threat modeling methodologies should be your first step toward building networks, systems, and applications that will be secure by design.

3 CHALLENGES INTEGRATING THREAT MODELING

One reason that threat modeling is performed as a first step is to obtain an objective viewpoint of the big picture for the project and define the locations of potential security vulnerabilities. **This can be done once the design has been defined conceptually.**

Obtaining this viewpoint will be helpful in choosing the threat modeling methodology that best fits the project. This step is a mission-critical decision, so it will require that some investigative research be completed to ensure that the right choice is made.

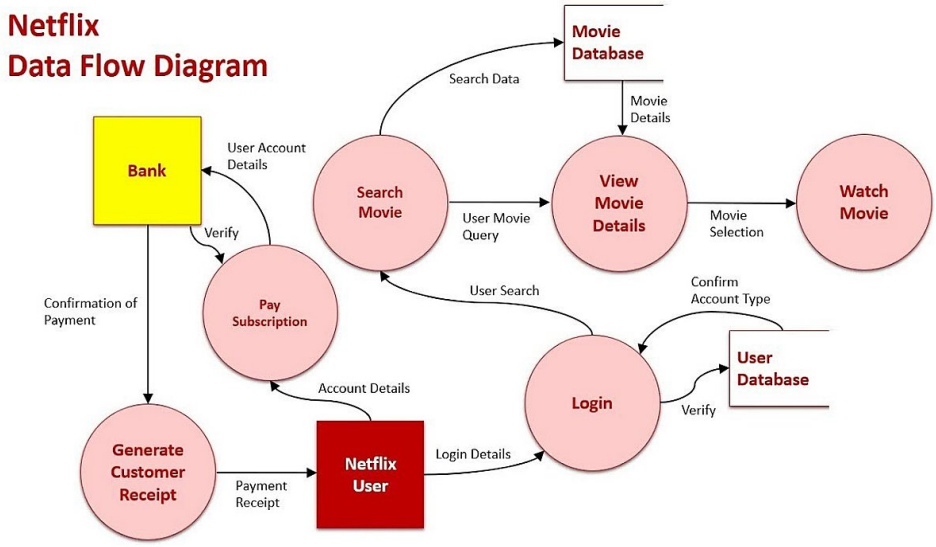
Today's threat modeling is evolved from practices that were developed nearly thirty years ago, such as drafting process (or data) flow and **attack tree diagrams**. Those practices led to the development of the well-known methodology **STRIDE**, discussed later in this section.

Though STRIDE is a highly popular and effective methodology, several others are also available. Some are more appropriate for different IT disciplines or have different focuses, such as applications instead of networks, for example.

No threat modeling technique is perfectly tailored to a specific use. **You should choose the one that most closely aligns with your goals.** However, your DevOps team should be encouraged to adapt or customize threat modeling techniques to better fit their specific use case.

Creating a **process or data flow diagram** is one of the best methods you can use to gain a top-level view of your system or application and its interaction points. Frequently, flow diagrams can be adapted from existing network architecture diagrams. Diagrams can be created with a pen and paper, or by using an application like PowerPoint or LucidCharts. Flow diagrams are useful tools which are more helpful and easier to interpret than trying to review the process or data flow in a textual design document alone.

Example of a Data Flow Diagram



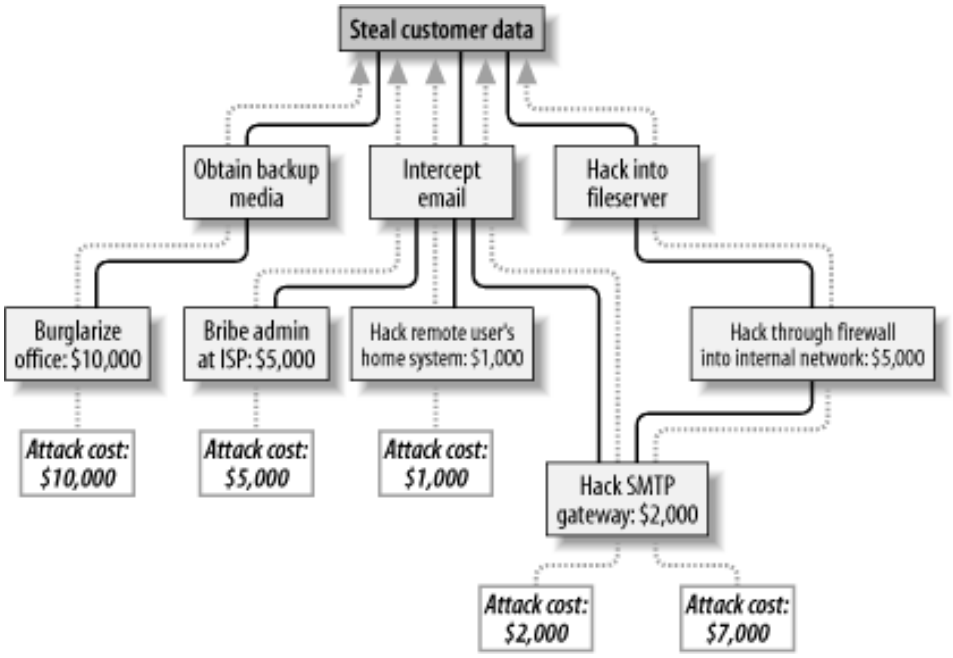
Source: Christopher Kalodikis

Start by assessing your system or application, and note where and how each entry point interacts with all of its associated external entities. One of the best formats for quickly and dynamically creating a threat model is the use of a whiteboard in a team environment. Determine which assets could be infiltrated by intrusion at each noted entry point. Rank those assets by the target's value. The higher the value, the more likely that asset will be of interest to a hacker. Then, identify the known possible threats associated with each identified vulnerable asset.

The next useful exercise in threat modeling is the creation of an attack tree diagram that integrates all vulnerability information that has been accumulated. This is a great aid in determining all potential threats against either infrastructure or application(s). Using it, you can identify specific threats, their possible entry points, and the potential exploitation of each threat. A thoroughly defined attack tree facilitates the ranking of threats, too.

THE NEW OWASP TOP 10 & WHAT IT MEANS FOR YOU

Example of an Attack Tree Diagram



Source: eTutorials.org

After fully defining the flow and attack tree diagrams, examine how the system or application will permit legitimate access to resources. Identify the external entities that will need to have such access. Establish the process by which that access will be granted. Determine the authentication and encryption methods to employ. The strength of those measures should be relative to the value of the asset(s), to ensure that the required trust level will be maintained.

It is beyond the scope of this guide to explain how to build your diagrams, but there are many excellent resources with examples on the internet.

There are several threat modeling methodologies used within the industry, as well as some tools that can aid you in performing threat modeling. Note that a tool that can effectively and automatically perform a complete threat model does not exist. On the next page is a very basic explanation of the threat modeling process.

Steps to Building a Threat Model

1. Gather threat intelligence

Identify and prioritize the current threats to your architecture, networks, systems, and applications.

2. Identify and inventory your assets

System assets, actors, connections, and your intended goals should all be inventoried. This includes both components and data. Be sure to identify their respective locations. Use this inventory to have the security team identify assets for which vulnerabilities are known.

3. Identify bad actors for DREAD risk management

Perform research to identify and assess bad actors. This is a complex process, because the identity of bad actors varies depending on the organization and its desirable targets. Locate reports of specific documented cases for attacks that have targeted the same type of organizations. Focus on the identities and capabilities of the bad actors in those documented cases. Use that information to develop a relevant security plan.

4. Identify and inventory mitigation assets

This should include technology, expertise, and available processes. Use this inventory to identify any areas that require the procurement or development of additional mitigation assets.

Steps to Building a Threat Model

5. Identify and inventory mitigation assets

Compare your threat intelligence findings with your asset inventories to determine your current risk assessment status. Abuse cases and potential attack types for the system should be assessed using models like STRIDE as a guideline. STRIDE is a modelling tool initially developed by Microsoft© to assist security engineers in identifying and classifying the possible threats on a server. STRIDE stands for the categories of threats: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Escalation of privileges.

- Risk levels should be assessed and weighted using a model like **DREAD** as a guideline. DREAD is a methodology developed by Microsoft©, used as a predictive tool to assess and rate the following potential risks: **D**amage, **R**eproducibility, **E**xploitability, **A**ffected users, and **D**iscoverability.
- Use the results of that exercise to prepare your plan for elimination of identified vulnerabilities.
- In the design phase, known or accepted risk factors should be inventoried and included in testing, per Chapter 2, subsections 2.3 through 2.7.

6. Perform threat mapping

Analyze your network(s), system(s), and application(s) to identify how potential threats might navigate through your assets in an attack. Use this mapping as a tool to identify how to employ your mitigation assets for the most effective defense possible.

STRIDE THREAT MODELING

This worksheet breaks down the six main categories of vulnerabilities in the STRIDE threat model. You may use it to help guide your own threat model. Use the space provided on the left to help record and track in your early stages of building the threat model.

CATEGORY	BREAKDOWN	EXAMPLES	YOUR NOTES
SPOOFING	Illegal access and use of another user's information	Email spoofing, DNS spoofing, IP spoofing, DDoS spoofing, ARP spoofing	
TAMPERING	Malicious modification or unauthorized changes	Wire transfer fraud, credit card micropayments, changing links on public websites	
REPUDIATION	Denial of malicious action	Illegitimately claiming a transaction did not complete	
INFORMATION DISCLOSURE	Exposure of information to unauthorized individuals	Providing access to source code files via temporary backups, revealing hidden directories	
DENIAL OF SERVICE	Service denial to valid users	ICMP flood, smurf attacks, ping of death attacks	
ELEVATION OF PRIVILEGE	Unprivileged user gains privileged access	Process injection, horizontal privilege escalation, vertical privilege escalation	

Other popular threat modelling methodologies that you may wish to explore are PASTA, VAST, Trike, OCTAVE, and NIST.

NOTE: Going forward, remember that your threat model is a living document and needs to be constantly reviewed and updated. After a system wide threat model has been performed it can be valuable to perform mini threat models as a secure engineering design requirement.

Regarding frequency:

- Full initial threat modelling should be performed and repeated annually.
- Delta threat modelling should be performed on any newly proposed feature or significant change at the design phase.

2.3 INTEGRATING STATIC CODE ANALYSIS

Best for Seed companies.

Static code analysis is a great way to catch low-hanging fruit in code paths that are not exercised often. It also acts as a way to standardize security across the program. Many enterprises request that their service provider have a SAST tool in place.

Static code analysis is a methodology used to locate vulnerabilities in application source code. Typically, it is an automated process that is performed prior to compilation, without code execution, to ensure that the code meets industry standards, leading to establishing a sustainable, secure program.

TOP SAST TOOLS FOR DEVELOPERS

The testing is done with full knowledge of the program's internal structure and the expected logic paths. Knowledge of the implementation is required. Therefore, the application's finalized design document is a prerequisite to test preparation and execution. Due to the transparency in the static code analysis testing process, it is referred to as a **White Box Test**.

The testing itself is a time-consuming process. Fortunately, there are automated testing tools available. Static code testing should be conducted by an expert with considerable experience in this discipline.

When implementing automatic SAST into your development, the following steps are recommended:

- ▢ Expand peer code reviews and evaluations to include security code review.
- ▢ Implement SAST on new code whenever a build or push is performed.
- ▢ Include a process for peer approval and validation of any exceptions or reclassification of security issues. If a vulnerability is found that has a medium severity or higher, escalate the issue to utilize your security or champion(s).
- ▢ Implement a full codebase scan every time a major version releases, or quarterly if there hasn't been a major version update recently.
- ▢ Consider implementing unit testing for security issues which have been a historical, common, or high-risk area.
- ▢ Perform quarterly scanning of code repositories' configurations, history, exposure, and management of secrets. All passwords, keys, or secrets should be kept in secure cryptographic storage, and never in the code repository. Any previously exposed keys need to be updated.
- ▢ Confirmed critical severity findings should be addressed before release. See Chapter 4 for how to identify these.
- ▢ Perform a minimum bi-annual auditing of code access controls.

2.4 INTEGRATING DYNAMIC CODE ANALYSIS

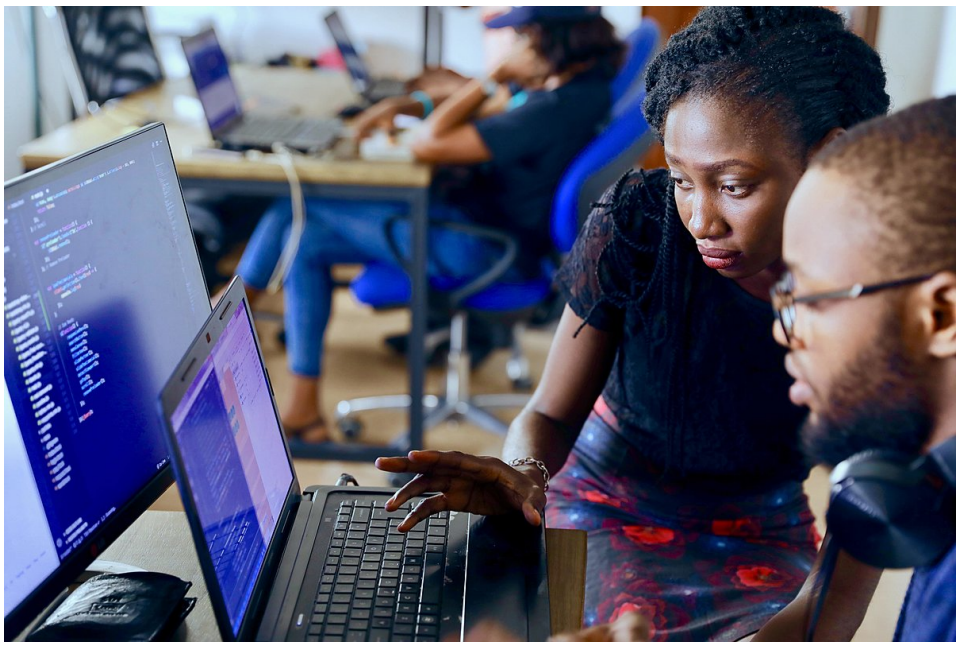
Best for Seed companies.

Integrating dynamic analysis is a great way to capture low-hanging fruit in production. It is an easy win on a vendor security questionnaire.

Dynamic code analysis is the process of analyzing a software program or application after development is finalized and the application is running. Usually, it is an automated process with the objective of verifying the quality, reliability, and security of the live/running application.

A primary advantage of integrating dynamic analysis into your project process is the ability to detect vulnerabilities that are either too obscure or too complex to be discovered by static analysis.

One of the benefits of dynamic testing is that it finds real-time, exploitable faults. In comparison, static analysis often finds theoretical vulnerabilities, which are not always exploitable faults.



2.5 INTEGRATING DEPENDENCY CHECKING

Best for Pre-Seed companies.

Since the number of dependencies used by applications on average continues to grow as the business grows, it is important that businesses integrate dependency checking early. This is also beneficial to ensure your business is not stuck with inherently vulnerable components, which become very expensive to replace later on. Dependency checking should be a primary tool for B2B companies, as visibility for supply chain software is very high.

The integration of dependency checking is important to your cyber security. There are scanning tools available to perform dependency checking. Modern applications often leverage several common reusable components developed by third parties. They look for known vulnerabilities that have been publicly disclosed. An application is only as secure as its weakest component.

OWASP has created and made available a free tool for dependency checking. **Dependency-Check** is a very useful, free tool which supports Java, .NET, and has experimental analyzers for Python, Ruby, PHP and Node.js.

It is helpful to maintain a reference inventory report of third party dependencies as they are added to your system. Doing so will enable you to determine those which may become publicly identified as having vulnerabilities.

Hosting a company's entire virtual environment on cloud-based servers does not negate vulnerability or the possibility of a security breach. Many clients that utilize this approach are complacent about security, largely because they are relying on the hosting service to assure the security of the cloud infrastructure.

However, servers could be misconfigured, which could then provide an entry point for breaches. If an administrator were to replicate a misconfigured server's image to create additional servers, they would be multiplying the same vulnerability. There is always a requirement for testing, risk identification, and mitigation in cloud-based enterprise architecture.

Cloud Native Application Security (CNAS) requires the same security disciplines that we have previously defined (i.e., static and dynamic code reviews and analysis) (refer to Chapter 2, subsections 2.3 and 2.4), and identification of open source dependencies and vulnerabilities (Chapter 2, subsection 2.5)).

Using automated tools that are specific to containers and Infrastructure-as-Code (IaC) puts security and compliance oversight in the CI/CD pipeline. This dramatic shift allows security gurus to conduct evaluation of IaC templates in the design phase.

In cloud-based security, the speed at which fixes are applied to discovered vulnerabilities and security breaches is critical. It is important to note that DevSecOps is not DevOps with a small security supplement. Rather, security is prioritized and implemented directly into the DevOps process. Companies who integrate DevSecOps and plan ahead will save development time and ease budget constraints in the long run.

DevOps teams are continuously pushed to deliver their wares on-time and within budgets. As a result, DevOps teams often lack adequate time to focus on the risks posed by intrusive security vulnerabilities. Opting for a DevSecOps team where security is prioritized and implemented right into the development process saves time, energy and risk.

Consider the benefits of a DevSecOps team for your project. The earlier that you can integrate security into your existing DevOps processes, the sooner you can begin leveraging your newfound competitive advantage.

CHECKLIST FOR BLOCKCHAIN SECURITY

2.6 INTEGRATING PENETRATION TESTING AS A SERVICE (PTaaS)

Best for Seed companies.Penetration testing as a service (PTaaS) provides more than just a one-time annual pentest. PTaaS grants constant access to a team of security experts, meaning code can be tested as it ships out. Security experts are also available for consultation on any other area of your application security, including design, threat modeling, results analysis and more. It is important to integrate security testing early in your business to set up efficient workflows and gain an understanding of security concerns while the team is smaller.

Automated tooling will always have a limited capability, as it usually cannot observe and construct the same level of sophisticated, elaborate attack scenarios as a human hacker. On the other hand, PTaaS is performed by humans, which means that it is one of the most realistic and comprehensive forms of security testing available.

Penetration testing (also known as pentesting) often uncovers weaknesses that would otherwise go undetected during the procedures applied in Chapter 2, subsections 2.2 through 2.6. As the tests are human-driven, the pentester is able to apply their own experience and closely simulate a real cybersecurity attack. An additional benefit of penetration testing is that it takes more time to understand an application's unique business logic. As such, the penetration tester will design specific application test cases for each application.

LEARN MORE

Penetration tests are necessary to locate and eliminate risks of significant, disruptive security breaches that cannot be found with the usual, automated scanning tools. The level of customization in this type of testing will result in the lowest rate of false positives versus automated tools which have a very high rate of false positives.

If you are getting one-off pentests, the automated testing described in Chapter 2, subsections 2.2 through 2.6, is still completely necessary as an ongoing part of your process in order to reduce the risk in between pentests. It will help evolve your overall security posture and culture.

If you choose to use PTaaS, the service includes unlimited re-testing between quarterly deep assessments. Unlimited re-testing means our team will check in on your application to ensure that no vulnerabilities are present after a fix has been applied.

Types of Penetration Testing

PENTESTING EXAMPLE: EXPLOITING LESS.JS

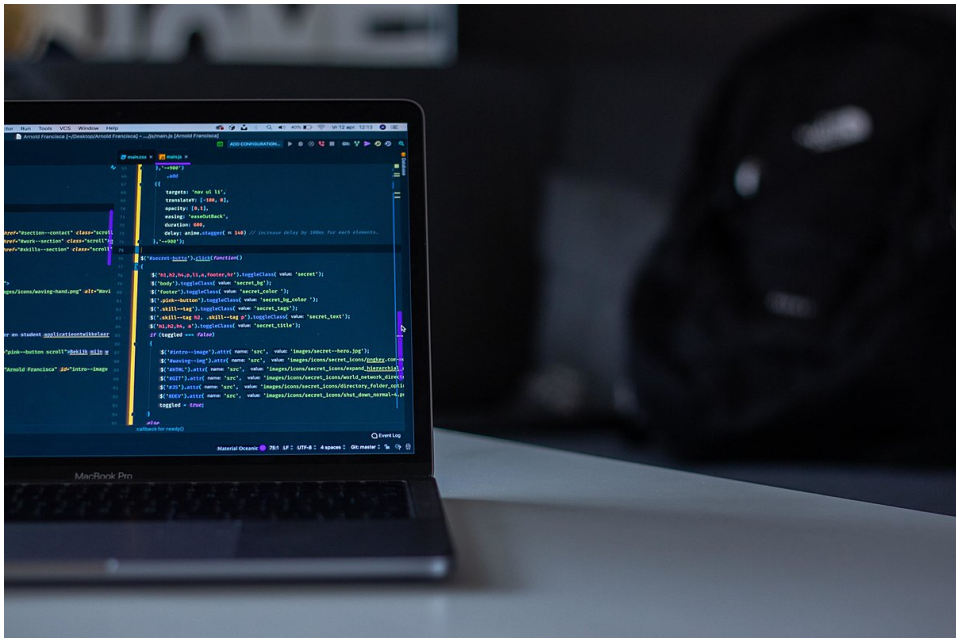
2.7 INTEGRATING A WAF

Best for Round B companies.

As your business continues to grow and service offerings expand, the attack surface increases. At this point, it might be more difficult to stay on guard with each application. A WAF acts as an another line of defense against attacks. It works in addition to any other controls.

A **Web Application Firewall (WAF)** should never be relied upon as a primary line of defense against application vulnerability. However, it does provide a good supplemental layer of protection to augment secure coding practices.

If a WAF is ever updated or swapped for a different vendor's product, you could suddenly expose undetected application weakness if Chapter 2, subsections 2.2 through 2.7, were not implemented. Frequently, WAFs can be bypassed or circumvented by attackers – it just takes a little longer. Think of comprehensive security as layers of an onion. If there are too many layers for an attacker to break through or overcome, it is often not cost-effective to attempt an attack at all.



- CHAPTER 3 -

LOGGING & AUDITING

Best for companies in Round A or higher.

In Round A, your company begins playing in the big league, meaning you're now a worthy target for hackers. Logging and auditing signals to your customers that you have a mature SDLC that can withstand attempted attacks. This security measure can help close bigger enterprise clients.

3.1 Logging & Auditing

Effective logging and auditing can be a bit of a balancing act. You need to log enough to be able to perform a forensic analysis, should you ever have a breach or other cause to validate the activity performed on a system. However, it is not desirable to log so much that you expose anything sensitive about the system or its users. Refer to the OWASP Top 10 2017–A10: Insufficient Logging & Monitoring.

Logging Too Much

- ☐ Is Personal Identifiable Information (PII) or other user information, when viewed in the context of General Data Protection Regulation (GDPR), being retained in logging? Is PII logged or obfuscated by the Universally Unique Identifier (UUID), also known as Globally Unique Identifier (GUID)? How long is logging with PII retained?
- ☐ Are there applications or system errors that might reveal a user system password, or key in a stack trace (which is now being stored in an unsecured storage medium)?
- ☐ Is there logging that an external customer is able to access, which may reveal too much internal system detail around a hosted resource?
- ☐ Is activity logging aggregated and tied to external alerting or detection tools which would bring immediate attention to a potential attack or breach?

Insufficient Logging

- ☐ If a breach occurred, could you determine the originating user, IP, tactics/payloads, and modifications to the application or system?
- ☐ If a legal situation arose where you may be called upon to prove non-repudiation of data on a system, what would you do?
- ☐ If a scenario arose where the target system had been breached and local logs were altered or deleted, what would you do?
- ☐ If the system were actively under attack, would an increase in system requests or other attack types be proactively alerted and/or detected by operations personnel?

- CHAPTER 4 -

INTERNAL & EXTERNAL SERVICE LEVEL AGREEMENTS

Best for companies in Round A or beyond.

SLAs are typically mandated by larger clients. As one of the goals of a Round A company is to target and close new large-size vendors, this step should be started in the Round A stage.

For any security program to work, it will need to have consistency, calibration, and accountability. The following specific information must be defined in your organization's **Service Level Agreement (SLA):**

- What needs to be done in any security relevant situation;
- How often, by whom, by when; and
- An effective way to track your organization's compliance to its own security policies.

4.1 INTERNAL SERVICE LEVEL AGREEMENTS

The internal service level agreement (SLA) defines a contract between the security and software development teams. While security teams are responsible for identifying risk, developers are the ones in charge of fixing found issues.

Where possible, automate and integrate all components of security issue analysis. Anything that cannot be automatically performed or integrated should be added to some form of a release checklist.

Specific internal SLAs for vulnerabilities discovered during scanning or pen-testing should be assigned a severity level appropriate to the specific weakness. These could vary based on the risk potential of the specific target or context. The chart below is an example of an SLA.

Known open security issues and/or defects should be tracked in a common database or issue tracking system. This database should be regularly maintained and audited by a security focal leader. Any collection of issues approaching or expiring the SLA date should be discussed among management, security focals, and development to ensure that nothing is forgotten or overlooked. Regularly scheduled meetings with stakeholders could be an ideal way to discuss issues together.

CRITICAL	STOP-SHIP. Fix before release or within 2 weeks.
HIGH	Fix within 14-30 days.
MEDIUM	Fix within 90-180 days.
LOW	Fix within 180-270 days.
INFORMATIONAL	Fix at developer's discretion or by customer requirement only.

AVOID SECURITY THEATER: HOW TO IDENTIFY SEVERITY

4.2 EXTERNAL SERVICE LEVEL AGREEMENTS

Your organization should have a publicly visible process by which to receive responsible vulnerability disclosures by third parties (e.g., a portal in which to log them or a secure email alias, published somewhere on your website, under security policies). Third party reporters could include prospects, customers, customer pen-testers, security researchers, bug bounty hunters, etc.

It would be advisable to impose a set of SLAs for external reports, similar to internal SLAs but containing more aggressive deadlines.

4.3 COMMUNICATION WITH THIRD PARTY REPORTERS

Communication with third parties about potential vulnerabilities should be carefully managed. How you handle these reports and the level of security culture demonstrated to these third parties is critical. It can often go a long way toward either improving or diminishing your organization's reputation with regards to security.

A note regarding bug bounty hunters: if your application or system has undergone several rounds of pentesting and you are more confident in your security posture, consider establishing a bug bounty program with clear rules and guidelines. Pay third party reporters for their contributions. However, it is usually not advisable to do this in place of pentesting as it can become very costly if your security posture is not yet fully matured.

For example, many security researchers will disclose the vulnerability details publicly, in a blog or security conference, to help advance their own reputation. They are well within their rights to do so and were kind enough to tell you about the issue first. Therefore, debating the issue with the third party reporter or threatening legal action would not be recommended in this situation.

A common external responsible disclosure period is about **90 days** from the initial report. It is usually best to employ the following steps when dealing with a third party reporter:

- Thank the reporter for reporting the issue to you.
- Follow up with them when you have confirmed the issue is legitimate (or request more details from them, if needed).
- Ensure there is a fix implemented (and communicated to the reporter, in follow-up) before the disclosure period expires. This will serve to avoid potential embarrassment or impact to your organization's reputation.

External third party security issue reports can be valuable assets when handled correctly and addressed within the time-frame that you have specified.

4.4 - RISK ACCEPTANCE & THE EXCEPTION PROCESS

No security policy should be completely rigid. There are going to be situations where a team cannot practically resolve a particularly complex security issue within the defined SLAs. Also, there will be situations where it might make more business sense to accept some minor risk versus expending major resources or costs to address it.

A good security program will include some form of exception process. Document when and how a development team or other stakeholders can present their case to senior decision-makers to request approval to override SLAs or accept risk. There are cases when this might be preferable instead of fixing some security issues. Be sure to clearly document these decisions.

Risk should be clearly measured. The DREAD system could be used to quantify the risk, similarly to the way in which it was employed in threat modelling (refer to Section 2, subsection 2.1). This approach will help assess any potential asset damage, exploitability, impact to users, and discoverability, versus the business case and costs to fix the weakness. This process should be the exception to the norm and only used selectively and infrequently, after all other feasible options are considered.

SOFTWARE SECURED

CONCLUSION

It is our hope that this guide will serve as an inspiration and reference for the current best practices available to VPs of Engineering, CTOs, Security Champions, and Developers in their quest to develop reliable, sustainable, and secure networks, systems, and applications.

We wish you every success in your business and hope you achieve the growth necessary to exceed your goals.

If there is anything we can do to assist you with security in your development projects, we would be pleased to discuss your requirements with you. Our services include:

- Penetration Testing as a Service (PTaaS)
- Application Security Training
- Secure Code Review
- One-time WebSec, AppSec and IoT Penetration Testing options

BOOK A CONSULTATION

VISIT US ONLINE

info@softwaresecured.com @SoftwareSecured 1-800-611-5741