A guide for software engineers and team leaders

APPLICATION SECURITY PROGRAM Handbook

Derek Fisher

MEAP

MANNING

MEAP Edition
Manning Early Access Program

Application Security Program Handbook

A guide for software engineers and team leaders

Version 2

For more information on this and other Manning titles go to
manning.com

welcome

Thank you for your interest in this book *Application Security Program Handbook*. This book is intended for developers, architects, and leaders that are looking to understand the purpose and benefit of building security into the development lifecycle. Although you don't need to be a developer, having an understanding of how code is developed and delivered to production will make the content easier to understand.

I have been working in application security for nearly a decade and have seen the successes and failures in organizations that are fighting the good fight of delivering secure software. Although the fundamentals of application security haven't changed much in that time, the tools, processes, and technology have. The threats have changes, the attacker's techniques are getting more robust, and the risks to organizations have grown. Organizations need to change with them and develop the framework that enables them to build security in from the start.

My desire for this book is that it will show how application security is applied in organizations and where it can provide the most amount of assistance in delivering secure software. The book is made up of three sections that take the reader through an effective application security program. The first section is dedicated to what are common application security practices and technology that you find in an organization. The second section dives more into how application security is evolving to meet the current demands of rapid development. This includes how application security should be developed as a service and not as the gated exercises. The last section will focus on delivering application security in an organization and measuring success.

I appreciate you picking up this book and my hope is that it brings you an appreciation for what it takes to deliver secure software in today's complex systems. Please be sure to post any questions, comments, or suggestions you have about the book in the liveBook discussion forum. Your feedback is essential in developing the best book possible.

-Derek Fisher

brief content

1

Why do we need application security?

This chapter covers:

- **Current state of application security**
- **Going right or going left**
- **Breaches as a result of insecure applications**
- **The cost of inaction**

Every company uses software to function. Whether they are a Fortune 500 technology company or a sole proprietor landscaping company, software is integral to businesses large and small. Software provides a means to track employees, customers, inventory, and scheduling. Data moves from a myriad of systems, networks, and software providing insights to businesses looking to stay competitive. Some of that software used is built within the organization or it is purchased and integrated. What this means is that every organization, regardless of size and industry, has a software need. It enables organizations to move quickly and stay ahead of their competition. In the United States, the growth of software in various industries including finance, sales, human resource, and supply chain has seen a steady increase with that trend continuing. Over the next decade these industries will see a steady increase of market size. For instance, in 2020 the market size of nearly 390 billion USD in the global business software and services. This is expected to have an annual growth of 11% from 2021-2028. This expansion is based on the increasing need for automation and processing solutions in nearly every sector on the economy.

The shift to software driven organizations can be seen in several prominent examples. Movies were rented at a physical store before Netflix began streaming them in 2007 directly to your television. Books were purchased in a shop before Amazon upended not just the book buying experience but the overall retail market for any item you can think of. It used to

take a phone call to order food, now a mobile app allows you to not only order your food but track it from the restaurant to your front door. Moreover, software is used in all parts of the supply chain for every organization. Leaders in these organizations can get real-time updates on product as it moves around the world and to their customers hands. This isn't a nice to have any more for organizations but is critical to any organization's ability to compete.

> *"In short, software is eating the world" Marc Andreessen.*

Each new piece of software brings new capabilities but also new challenges, especially when that software is available over a network where it is open to a wide audience. More capabilities mean more software. More software means more data. This data is intended to assist an organization and its customers to gain insight into the services the organization offers. However, the additional software and data means more opportunity for bad actors to leverage weaknesses that might exist. Data is a lucrative target for attackers and difficult for organizations to protect especially when the organization collects a large amount of it. By the beginning of 2020, the amount of data created worldwide had reached 44 zettabytes and this is expected to reach 463 exabytes by 2025.

The increasing software landscape and the repository of data growing every second offers a target rich environment for malicious actors looking for a way in and access to the data. There is constant background noise of malicious activity that pervades the internet. The moment that any software becomes available online, it is immediately probed and prodded. Some of that activity is from automated tools that will then alert when it finds software that can be compromised. Some of it is from bad actors waiting for the latest vulnerability to become public so they can hunt the internet for applications that are vulnerable to it and create exploitation software. Other activity comes from well financed bad actors that are simply looking for any weaknesses so that they can gain a foothold in an organization in order to come back later for exploitation. If ever there was a more fitting metaphor for software on the internet, it would be the Wild West of the early United States where lawlessness was rampant, and sheriffs were often overwhelmed and outgunned by the range of activity.

Software, data, attackers, oh my! When organizations don't take software security seriously, they run the risk of not only putting their client's data at risk, but the organizations future as well. Most organizations that are not in the software building industry will often say that they are not a prime target, or that their data is not interesting to an attacker. They would be wrong. Almost all data that is collected and processed by an organization can be used and misused by an attacker. Nothing is too unimportant. And although many large organizations can weather a data breach, smaller companies often cannot while remaining in business.

It is often said that data is the new oil. Mining, processing, and selling data is a lucrative business both for the organizations that do so openly and the malicious actors who are looking for ways to profit off an organization's missteps. Building security into the software from the start is the first, and primary, step to ensuring that organizations have the means to protect their software, their data, and their livelihood.

1.1 The role of an application security program

Application security is the implementation of security through practices, tools, technology, people, and processes in the development lifecycle. The rest of this book will cover how application security is used throughout the development lifecycle to ensure that an organization has reduced the risk posed to its software and has done so in a way that does not impede an organization's ability to deliver that software in a timely manner. It is important to know that there is never a silver bullet to solving security issues. No one tool, process, or person can claim that as attacks are constantly evolving and technology keeps moving. However, ensuring that you have a robust approach to application security that includes not just the fundamentals of an application security program but also the ability to adapt and evolve your approach on an ongoing basis is what sets apart a good program from a bad one. This is what we will discover throughout this book.

Products start as an idea in a client or someone at an organization's head. Transferring that idea from concept, to paper, to minimum viable product, to a production application takes a team of people from multiple disciplines to bring it to life. Most will think first about all the developers, testers, and product people that go into making a successful product, but what about those that make the product secure? The goal is to not just make a product that does what the customer wants, but also is free from defects and security issues that would otherwise devalue not just the product but the organization as a whole.

1.1.1. Security from concept to production

The Software Development Life Cycle is defined differently in every organization, but in general, every organization has a similar path from concept to production. Figure 1.1 depicts a standard software development lifecycle, where:

- The client or product owner envisions a new feature or enhancement.
- Use cases and requirements are developed. The team determines where to slot the work for an upcoming release.
- The development team then makes decisions on how the requirements will be implemented through technology choices. Development begins and the code is integrated into a feature branch.
- The feature is then moved to a lower environment like a test or staging environment where tests can be performed. The product owner will accept the results of the test and agree to have it released to production for the clients to use.
- The feature is released to clients and the organization manages the feature as a set of a larger application through client and technical support. Eventually, the feature is decommissioned in favor of a new release.

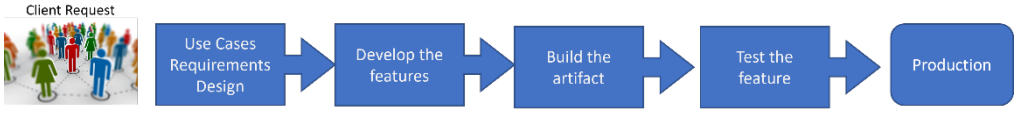


Figure 1.1 Phases of an SDLC

In the first step, the product may already exist in which case the client would request an enhancement or a new feature. For instance, perhaps a new method of accessing reports within the application, or a new dataset to be available in the user interface. Otherwise, the product owner may be collecting information from several clients and put in an enhancement request for a new feature. In either of these cases an organization will gather the client needs in the form of use cases. A simple use case could be written as:

> ***"As an administrator I want to be able to create weekly reports that show the application usage among users in my organization."***

This is a simple use case for an application that is SaaS (software as a service) based and is used by many organizations to provide some service. The details are not important in this case as every organization is different regardless of the industry they are in. In most cases, a feature would include multiple use cases.

Once the client or product owner has the use case defined this is then reviewed with the development leaders of the product. The architects, lead developers, and operational members of the development team review the use cases and determine the feasibility of the request.

This brings the use case to the next step in the process where the functional and non-functional requirements are defined. Like having multiple use cases, each use case can have numerous requirements. Following is a functional and non-functional example of requirements for the previous use case:

> ***Functional: "Application shall provide the ability to create a report of application usage by user."***
> ***Non-functional: "Report creation shall be available to only administrators."***

This is not a book on writing requirements, but a simple explanation is that *functional requirements* describe what the feature should do, and *non-functional requirements* are how the feature should do it. Most information on requirements describe security as a non-functional requirement. However, it is not uncommon to find security features being described as functional requirements, especially as it comes to things like encryption, authentication, and access.

1.1.2. Where does application security program

The product owner will take the defined requirements and decide with the development team what priorities might need to shift and when delivery dates are for the new feature. There is something missing so far. Where are the people, tools, and processes that bring security into a product? As mentioned, security is typically a non-functional requirement. However, those non-functional requirements should not be left up to the product owner and the development team to determine on their own. A more mature organization would bring in the application security team to not just review the use case and requirements, but also define security requirements that should be in place as part of the feature development. If the product team defines their requirements, and begins coding before engaging the application security team, they run the risk of creating security issues that are more difficult to resolve once the feature is nearing completion or ready for release. For instance, a development team may know that they need to provide authorization and may decide to develop an internal solution to manage the authorization without looking at a more fitting solution that is used across the organization and is more industry aligned like OAUTH.

Fitting security in during the initial phases is done by taking industry and organizations standards and best practices and building them in to the process of creating requirements. For instance, using industry guidance on encryption would lead to setting requirements on key management and encryption strength. Or the organization may have standards that require the development to adhere to certain architecture or regulatory requirements like using a specific analysis tool. **Additionally, when building these requirements, the team will take inputs from items like threat models, and risk assessments to further develop requirements that align to the business requirements so that security can be built into the process as early as possible. We will discuss threat models and risk assessments more in depth in future chapters.**

1.2 The current state of application security

As mentioned, application security is the implementation of security into a development organization. The reality is that application security teams, if they exist in an organization at all, are often external to the engineering teams they work with. They can be found under an enterprise-wide function like the enterprise architecture organization, or under a broader security organization. The last one is the most typical. These application security teams will bring in tools, processes, and people to identify software vulnerabilities that are then backlogged for remediation. These vulnerabilities are found through several opportunities, including the following:

- Through the tools that are provided to engineering by the application security team.
- Through external or internal penetration tests completed by the application security team or an external vendor, respectively.
- Through identified issues from clients, or other external sources like a bug bounty program, or vulnerability disclosure policy. More on these in later chapters.

If the application security team and the organization are considered mature, these vulnerabilities, when found, will block builds. You can see how this can cause issues between

the engineering team that is working toward a release of code and is suddenly stopped by tools and processes that have been put in place by an external team. Blocking a build is by far the most preferred method for an application security team that wants to ensure a secure product, but this can become an instant point of contention. In less mature organizations a process is in place to allow the engineering team to continue with their build and deployment as by having the found vulnerability backlogged for future resolution.

The application security team is in regular competition with feature release. Every new release brings new features. New features bring new defects. New defects bring new security vulnerabilities. As mentioned previously, in some cases these vulnerabilities can break a build and block the release of a feature. However, most organizations will prioritize their feature release over a nebulous vulnerability. I will discuss this more later, but many scanning tools are noisy and produce results that are not easily consumed by developers. Furthermore, not all application security teams are great at translating results from tools and other tests into something that is understandable by engineering without having a meeting or work session to understand. This obviously doesn't scale well in large organizations.

The security issues are backlogged with an I-owe-you to address the issue in a future. But every new release brings more vulnerabilities to the growing backlog and the cycle continues. Most organizations will take a systematic approach to reducing vulnerabilities such as focusing on only the high and critical ones, or ones of a certain type like SQL injection or cross-site scripting. Other organizations may focus on the riskiest vulnerabilities based on product capability and exposure like a financial organization processing sensitive account information. Some organizations may even take the approach to have a "security release" where their focus is on resolving a large number of security issues in a single release. These different methods help reduce vulnerabilities in burst but don't address the overall issue.

Like Jacob Marely from the Charles Dicken's 'A Christmas Carol' story, these vulnerabilities become chains that weigh down the development team and will eventually haunt an organization. The continued accumulation of vulnerabilities adds to what is called security debt where an organization continues to add new vulnerabilities to the old ones that were already existing. This security debt increases the risk level of the product being developed and the organization that is selling it. Eventually one, or several, of those vulnerabilities will lead to an exploitation of the application by a bad actor. This is similar to the concept of technical debt that builds when an organization takes a quick and easy path to getting features to their customers instead of choosing more sustainable design choices.

Further complicating the job of the application security team is the fact that they are often brought in too late in the development process. Usually, it is once the feature or product has gone through several design and architectural decisions. The code development may have already been well underway, or worse, nearly complete by the time the application security team has been involved. This means that many "one-way" decisions may have already been made and it is up to the application security team to provide some blessing of the design and code, or identify mitigations for discovered threats. This is not the case everywhere, but in a sufficiently large organization, this will happen. If the application security team has had the opportunity to provide guidance, requirements, and security tooling early in the process, potential vulnerabilities can be reduced. Unfortunately, the cases where the application security team is involved early are few and far between, despite it

being effective. This leaves the application security team hampered with the decision to be "that person" by blocking a release in order to impose security requirements before production or face consequences from the broader organization for allowing code to be released with known weaknesses.

Such is the current picture of application security where there exists a constant struggle between enabling the business and reducing the organizations overall risk.

1.3 Why building security in is challenging

> **"There comes a point where we need to stop just pulling people out of the river. We need to go upstream and find out why they're falling in."**
>
> **— Desmond Tutu**

The application security team has at its disposal the most state-of-the-art tools that include technologies like machine learning, artificial intelligence, natural language processing, and automation. However, some of these tools detect issues once the code is written, and most likely checked in and on its way to a production environment near you. I will talk more about the various tools in future chapters, but the as mentioned there are several tools that are commonly found in a modern development pipeline related to security. Static application security testing (SAST) can scan written software looking for commonly found security issues like hardcoded passwords and SQL injection. Dynamic application security testing (DAST) will attempt to perform real time security testing on a web application while it is running in an environment. Software composition analysis (SCA) tools will look for known security vulnerabilities and license concerns with third party and open-source software that is used to build the overall application. Additionally, there are cloud architecture, container, infrastructure template and mobile security tools that can produce scan results that will identify vulnerabilities or other weaknesses in the code or deployment of the software.

When a vulnerability is detected, the tools can open a ticket to application security team and the engineering team, so long as the integration between security tool and the defect tracking tool is setup. The issue is then triaged with the application security and engineering teams, prioritized, and worked to closure.

> **EXERCISE** **Take a look at OWASP's page on Source Code Analysis Tools and review some of the available tools. There are several that are "Open Source or Free." One of these open-source tools is the APIsecurity.io security audit (https://apisecurity.io/tools/audit/). You can use this tool to upload an OpenAPI JSON file to detect possible vulnerabilities. If you do not have your own OpenAPI JSON file to use you can search for one online, or use a sample like this:**
>
> **https://github.com/OAI/OpenAPI-Specification/blob/main/examples/v2.0/json/api-with-examples.json#L31**
>
> **Take a look at the results that you get back from APIsecurity.io and determine whether the issues are true positives, or not. Begin to think about how to mitigate the issues that are found.**

1.3.1. Trying to protect at runtime

Although the aforementioned tools are detection tools, there are protection tools as well that will sit in front of a running application and attempt to block activity that looks malicious. Web application firewalls (WAF) provide protection against attackers looking to take advantage of weaknesses in a running web application. Run-time application security protection (RASP) will provide similar function as the WAF with the exception that RASP generally runs alongside, or even inside, the application. Bot mitigation software and denial of service protection both will attempt to stop volumetric attacks that send large quantities of malicious traffic that attempt to bring down the application or perform repeated tasks like brute force activity. Secure gateways will provide similar protection by blocking unauthorized access and activity as well as provide real-time monitoring.

Again, these protection tools use all the latest and greatest techniques to attempt to provide protection like machine learning, and artificial intelligence. Some of these tools are great at blocking unwanted activity by malicious actors but at the same time, some tools run the risk of blocking an organization's clients from using the software as it was intended if the tool is not properly tuned. An example of this that is common is when a batch job runs and calls an API or function hundreds or thousands of times in a short period of time. This could look like automated malicious activity to the protection tools and could block the legitimate traffic. To separate the two, the application security team and the engineering organization have to work together to pattern behavior into rulesets that block malicious traffic and allow the good traffic. This can come in the form of allow-lists for certain URLs and IPs. There is a steep hill to climb to enable protection tools since many organizations, understandably, will be concerned with performance and possible interruptions of legitimate traffic.

1.3.2. Getting output from tools is not enough

Like a comfortable blanket, security tools that are layered in during the development process and pipeline can become a reassuring to an organization. However, as with most tools, the effectiveness is determined by how well the tool integrates with the organization and how well it protects or provides legitimate results. Organizations that enable one or many of these tools simply to say they use them, or (*cringe*) say that they block the OWASP Top Ten, are not doing themselves or the organization any favors.

> **DEFINITION** OWASP is the Open Web Application Security Project that is an opensource community of application security professionals that develop standards, tools, and projects to assist organizations with the development of security in their applications.

Sure, during an audit the organization can say that they are using tool 'x' or 'y' during their development lifecycle. Regardless of whether the auditors pencil gets to work checking a box, the organization may or may not actually be more secure.

The reality is that these tools can create a lot of noise for both the engineering team and the application security team. The scanning tools churn out findings that need to be triaged, rated, and assigned. Many are false positives. The blocking tools create false alarms and raise concerns about the impacts on legitimate activity. And many times, there is an

overreliance on the security tools to provide protection especially when there is a vulnerability that is long in the tooth with no plan to remediate. For instance, an organization may rely heavily on a WAF to provide protection for a SQL injection vulnerability found in an application that has been designated as "sun-set" with a multi-year decommission. We'll talk more about that as we get into vulnerability management.

1.3.3. Sifting signal from noise in security tools

The security tools that are used in an organization can be expensive and misconfigured like any other tool. Further, those tools that are not finely tuned will generate an abundance of false positives. Like the ones that are turned on and walked away from. This not only creates additional work on the application security and engineering teams, but also reduces the confidence level in those tools and, by extension, the application security team. When false positives become normal, they become an easy escape route for engineering teams looking to find a way to say that their application is not riddled with vulnerabilities. If the last ten SQL injection issues flagged by a tool were false positive, why would this new one not be? Which brings the application security and engineering teams to a standoff on proving a finding to be a true vulnerability or a false positive. This can be extremely challenging for the application security team which typically does not have the extensive context that the engineering team has of their own application. It is also a time-consuming process to bring together the appropriate subject matter experts (SME) to pour over the details of the code in relation to the finding.

With the varying tools and the number of findings from each of them, mature application security teams focus on sifting the signal from the noise and providing quality results to the engineering teams. This raises the confidence level of the findings and establishes a more robust relationship between security and engineering.

The application security team will work closely with the engineering team and attempt to have as much application context as possible so as to take the burden of proof out of the engineering teams hands. In other words, the application security teams goal is to ensure the following:

- Results are true positives that have already been triaged by the application security team.
- The steps to remediate the vulnerability are clearly understood by the engineering team. If possible or applicable, the application security team should provide code samples that show exploitation and resolution.
- Clear expectations on timeline to resolution based on the criticality of the vulnerability.

We will dive into this more in future chapters, but for now let's look at how security can be integrated into the development lifecycle.

1.4 Shifting right versus shifting left in development

Whereas every organization releases software in their own manner, for most organizations the path from idea to production is relatively the same. Figure 1.2 shows the common pattern to release.

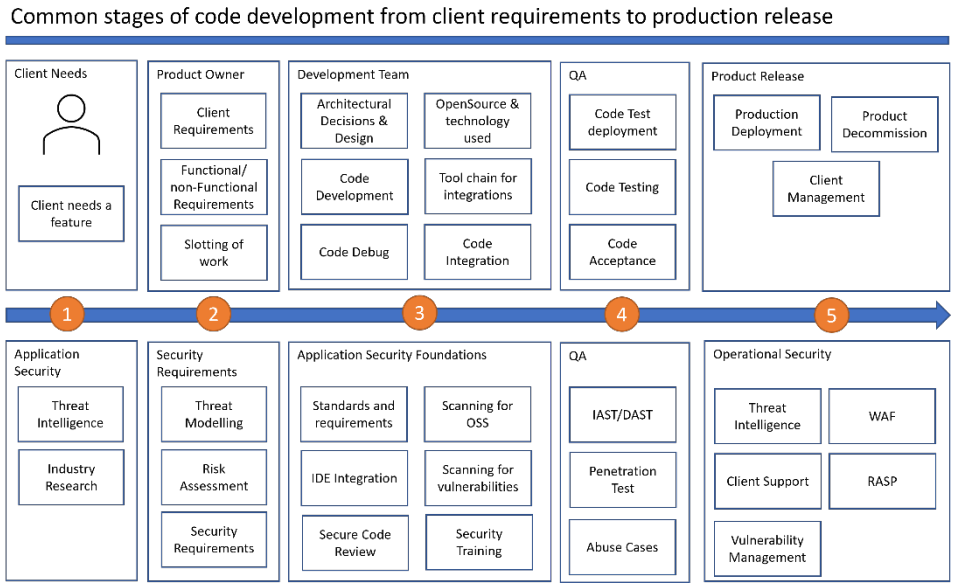


Figure 1.2 Example of a development pipeline for code deployment

The phrase *shift left* is the concept of moving security as close to the beginning of the software development life cycle as possible. In Figure 1.2, that means during the initial stages of gathering and building requirements as well as in the development phase. The term has been used frequently in the application security space as a way to describe building practices, and tools that can uncover security issues as soon as possible in the development lifecycle. Many of them I've mentioned previously. Sticklers will tell you that the best way to accomplish this is to ensure that security is there when the developer's hands are on the keyboard creating that new function. Those sticklers would be right. The time to correct a security vulnerability is when it is being created.

Shifting left is less visible than shifting right. In the shift right model, tools are placed strategically throughout the development lifecycle and production environment to ensure that vulnerabilities are identified and protected against. Penetration tests are executed to identify issues. You can verify that the tools are working, and you can generate reports that show you the effectiveness, or the ineffectiveness, of the tools that you implemented. This relies on a detect and respond paradigm that is very reactive and adds to the backlog of vulnerabilities that I talked about earlier where the critical and high ones are usually

prioritized while others go to the backlog. This can also disrupt the DevOps model that looks to move quickly with changes and doesn't handle broken builds or gates very well.

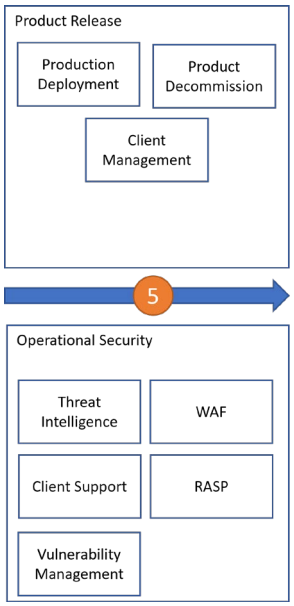
Getting to the goal of developers creating more secure code usually means using controls like:

- Security training.
- Top level security policies that are used to develop security procedures, processes, and standards.
- Tools that are integrated into the development environments and pipeline that offer faster feedback
- Building reusable secure architecture.

However, some of these can be circumvented. Like most training, security education can quickly be forgotten or pushed aside for the sake of speed of delivery of new features. Developers also change roles, jobs, and functions and are often overburdened with deadlines, tickets, requests, fires, and meetings which favors moving fast as opposed to secure. This leaves even the most well-meaning security conscience developer to push security further down in priority. Especially when there is a reliance on the protection tools, as described previously, that will detect and alert on security issues. Security quickly becomes someone else's problem. Additionally, the tools become a business blocker which opens the opportunity for the product owner to request exceptions when a feature release is at risk due to a found security issue. Architecture is frequently misused or not well socialized across the organization meaning that not all development teams are aware of frequently changing architectural patterns that offer more security.

1.4.1. Shifting right in the development lifecycle

When an organization decides that their security posture will be mostly a shift-right one, they integrate tools into the development lifecycle that will detect issues and open tickets to the security or development team. Most of these tools are used to finding issues in production, or late in the development process. These organizations will enable a few protection mechanisms like a Web Application Firewall (WAF) and primarily play defense by tracking the incoming vulnerabilities, triaging, and prioritizing them, and assigning them to teams to be resolved.

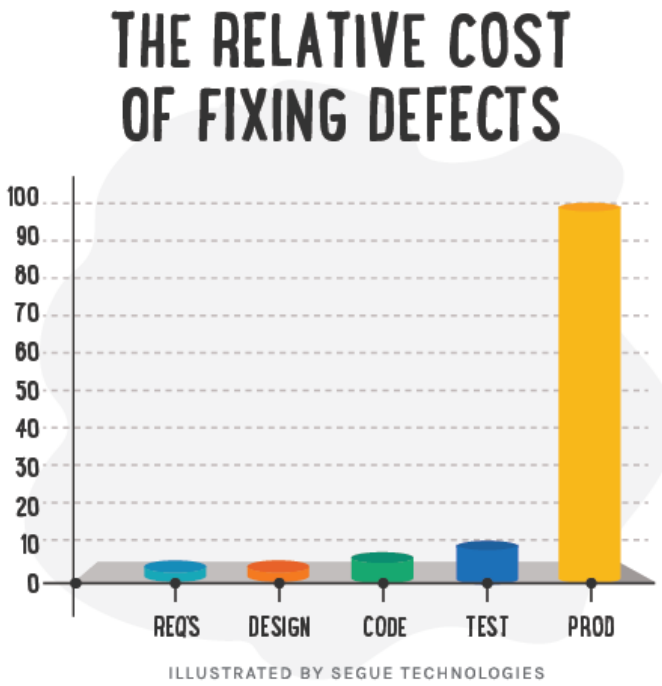


When shifting right, tools are used to detect issues later in the development lifecycle.

Figure 1.3 Security tools that are used in a shift-right approach

It is well known that resolving defects, in this case security vulnerabilities, cost more in terms of money and time than an issue that is resolved early in the life cycle. The effort and disruption that is required to resolve a defect that is already in a production environment can be multiple tens of times more expensive than resolving it at the requirements phase with each progressive phase of development becoming more expensive. There are also service level agreements that could be at risk when a vulnerability gets resolved in production if an outage is incurred through resolving the vulnerability. This further expose the organization to additional costs above and beyond the engineering cost of resolution. Security vulnerabilities have the added impact of potentially leading to a reportable event or even reputational damage should the vulnerability lead to a large-scale breach that exposes client data or takes an application offline.

However, shifting right does allow the development team to produce features at a rapid pace since security is largely a defensive position when the software is already running in production. This allows for the development team to spend less time resolving issues early, and instead rely on the protection mechanisms in place. The decision to rely on the right-sided tools and processes is one that is made by balancing risk versus reward since failing to deliver a feature on time has its own impacts on the organizations bottom line.



1.4.2. Shifting right fails

There is no shortage of stories where security controls were in place but failed to stop a larger breach or exposure. This happens for a myriad of reasons: alarm fatigue, security is someone else's problem, too many competing priorities. Those that work in the security industry know that there is always an open port, an insecure version of software, and a place where there is a lack of security controls. And attackers are just as in-tune to this as the security workers. Attackers only need to be in the right place at the right time, once. Defenders need to be right every time.

If you're a fan of zombie stories, you'll be familiar with the individual or band of living humans that find themselves inside of a building surrounded by the drooling, groaning undead. In most cases there is simply too many surfaces and weaknesses in whatever building they find themselves in. As the horde outside grows the defenses become weaker and the living inside have fewer and fewer options to keep the zombies out. Working in the cybersecurity field can sometimes feel this way. Every time, you shore up a weakness in your defenses a new one is discovered, and your team is tasked with devising a plan to close the weakness and provide a meaningful defense. Additionally, attackers are not always the

mindless zombies pressing your defenses, they are often smart, patient, and know exactly what they want. Good thing the defenders are too.

Bad actors are finding more ways to attack applications and to the defenders of those applications, it continues to feel like there are more vulnerabilities than they can manage. More features, mean more attack surfaces, which means more opportunities for a bad actor to find a way to steal data, impersonate a user, perform fraud, or other nefarious activity. Furthermore, the more integration with internal or external applications and services means that there can be exposures that the organization can't control.

Case in point: In 2018 a vulnerability in Facebook led to the compromise of tens of millions of Facebook accounts. The flaw was in a feature that allowed a user to view their profile from the point of view of a different account. No surprise, this feature was called "View As." Bad actors were able to steal the access tokens of Facebook accounts which allowed them to then log in as the user that the access tokens were associated with. They started with their own connected friends and from there stole the access tokens from their friends' connections until they had collected several hundred thousand accounts and then several million. They were able to collect personal data including the usual suspects of name, contact information, places the user checked-in, and other private data.

This example shows the difficulty in providing a secure product that is open to millions of users with a multitude of features. Even for a large organization that takes security seriously. The tools that we spoke of previously may have helped identify this issue prior to allowing the code to go out the door or would have detected and blocked it once it was running in production. However, one of the limitations with these tools is their inability to discover business logic, or workflow related vulnerabilities. Furthermore, it can be challenging to rely on tools to uncover these issues quicker than end users do. This is where having the proper processes, requirements, and testing early in the development lifecycle would raise the opportunities to uncover this issue early where the collective effort of tools, testing, and keen security eyes are brought to bear.

1.4.3. Shifting left in the development lifecycle

Where shifting right means that the organization attempts to put as much effort into protecting and detecting security issues later in the lifecycle, shifting left is pulling that effort earlier in the lifecycle. This is by far the preferred method of development security due to the fact that it is less expensive and more effective. However, it is more difficult to implement and can be bypassed rather quickly if the need arises.

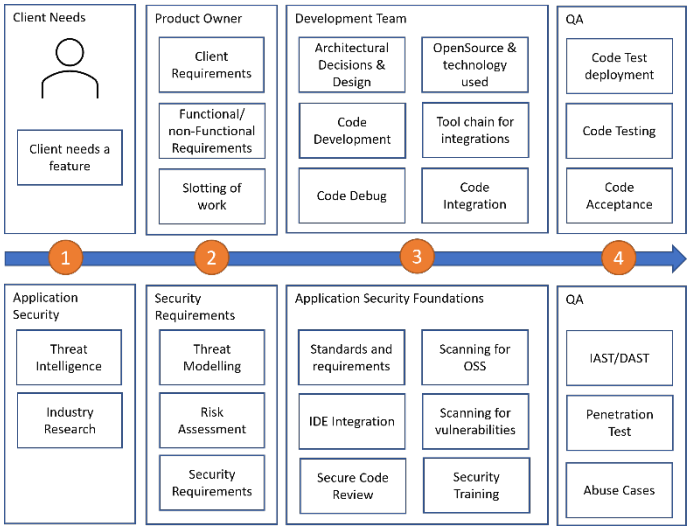


Figure 1.4 Shows the security tools that are used in a shift-right approach

Imagine building a house. You have an architectural drawing, a bill of materials, and the actual building materials. You get a group of laborers together and you get to work building the house. It is far more preferable to put the locks on the windows and doors, build the egress window, install smoke alarms while you're building the house. Waiting until the house is built, or after the house has been robbed, or burned down, is too late. Yes, you will save time and money during the building process, but you are less secure for it. This example sounds silly when stated, but the reality is that this routinely happens in software development. Sometimes it is because newer security patterns and architecture are discovered after the software is built, but it is commonly due to the lack of building security in at the beginning.

It is far more practical to layer in security throughout the development process starting with the design decisions being made and requirements being gathered. When this approach is taken, the organization is taking the necessary steps to build better habits and have longer lasting impacts. Take into consideration your personal health. Studies have shown that adjusting habits rather than going for a quick fix diet are not only healthier for you but provides a more sustainable path to better health. Shifting left builds those healthy security habits that will ensure that the organization is on the right footing and can sustain a more secure overall posture.

A healthy secure development environment starts with making the right architecture and design decisions that take security into consideration. This means picking the right security controls in areas like session management, encryption, authorization, and the like. It also includes leveraging tried and true patterns and standards from well-regarded and vetted organizations like OWASP. For instance, the time for picking the right data protection scheme should be determined while architecture decisions around data flow, and database technology are being made. Requirements for the encryption strategy should be well

documented and provide the appropriate level of protection based on the data classification. Additionally, requirements like field level encryption with proper encryption key lifecycle management are much easier to develop before there is terabytes of data in the database that would require applying encryption to a large dataset. Where this can get complicated is with legacy applications where most organizations cannot provide encryption beyond the disk level due to older technology that may not support more granular, robust, and modern encryption. This is simply due to the fact that the application may never have been designed to work with encrypted data.

Even the language that is chosen can have an impact on how secure an application will be. There are literally hundreds of development languages that developers can choose from. Each with their own strengths and weaknesses for the given use cases. However, many modern languages provide some guardrails that can keep developers from producing insecure code. For instance, it should be no surprise that Java, and C++ tend to rise to the top when it comes to vulnerable code. Much of this can be attributed to the power in each of these languages and the ability for developers to shoot themselves in the foot.

> **NOTE** One of the most common issues with powerful languages like C++ is its ability to manually manage memory. Most modern languages will take this ability away from the developer as a convenience. One specific example with C++ is the ability to call free() that allows the developer to free a memory address. If this is called twice with the same memory address, this becomes a doubly freeing issue. An attacker is able to leverage the memory leak that is made and inject code possibly allowing the attacker to have an interactive shell with elevated privileges.

Additionally, these languages are widely used in billions of devices across the globe. An increased footprint means more opportunity to find security issues. Other languages such as Python, Ruby, and Go show fewer overall vulnerabilities but there are also fewer lines of code written in these languages.

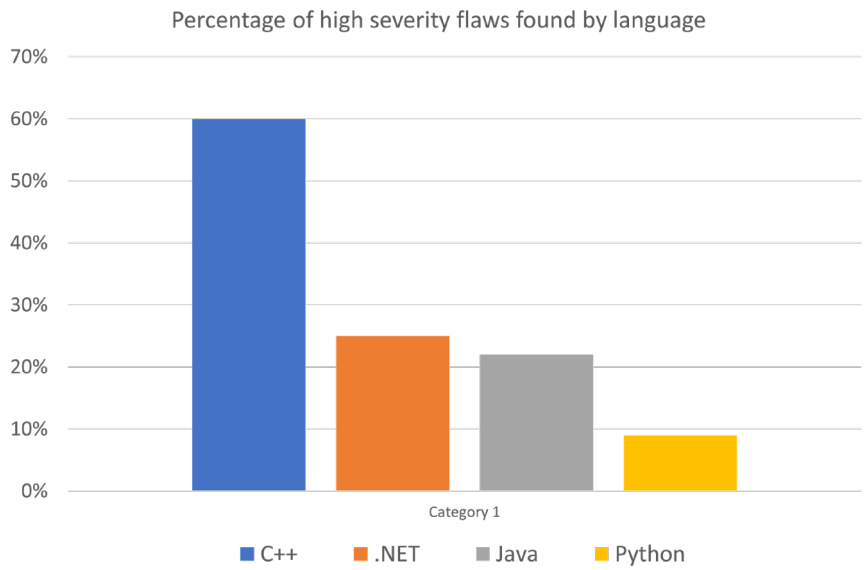


Figure 1.5 Application with high-severity flaws by language 2020

As much as language and design choices have an impact in the shift left strategy, so does testing. Unit, QA, integration, and system test can all be used to identify security issues early in the lifecycle and allow time for the development team to correct an issue before it goes out to a production environment. This does assume that the appropriate security tests have been created, ideally automated, and are alerting the team when an issue is found. It further assumes that the findings do not disappear into the security vulnerability abyss of a backlog.

We will talk more about shifting left in the coming chapters, including items like threat modeling, measuring risk, creating abuse cases, using development tools, raising awareness and more. Make no mistake, shifting left is the most cost effective and sustainable method of bringing application security to an organization near you.

1.4.4. Shifting left fails

Often, one or two members of a development team are designated as the "security person" (We won't call them champions. That comes later) for the team where much of the security related work is dumped. This person becomes responsible for being at the meetings where security decisions are being made, they perform code reviews on security related changes, have to make decisions for the team, and is generally responsible for correcting or setting a direction on vulnerability management. By the way, this person also has a day job that is usually as a developer or architect for the team. They may not even want the role of the "security person" but they may have been voluntold. This has a huge impact not just on that individual but on the team as a whole. This person is quickly overwhelmed without much opportunity for rest or objection.

This is generally where things like vulnerability management falls apart. A new vulnerability is discovered in one of the security tools, or a penetration test. The vulnerability is placed in the defect tracking tool and assigned to the security person who then puts aside the regular development work that they were possibly working on to triage the security issue and attempt to resolve it. Meanwhile several other vulnerabilities have been identified building the security debt we talked about earlier. This is usually when the product owner, scum master, or development manager comes in asking why a development deliverable is behind.

This type of failure has been seen several times in the past. One of the recent high-profile cases was with the Equifax breach that led to the exposure of over 143 million American's personal and financial information. Equifax is one of the big three credit bureaus in the United States providing consumer credit reporting to Americans. In March of 2017 a vulnerability was found in Apache Struts, a framework for developing web applications. A patch was released, and most organizations set out patching their software. This is generally easier said than done as some framework upgrades may need additional development changes and testing. In worst cases, an application may need to be re-architected or have major development work completed. By May of 2017, two months after Apache released a patch, attackers had gained access to the Equifax database and began to steal information. Equifax became aware of the breach in July. There were several failures in the security organization that led to the exposure of personally identifiable information (PII). Although the issue was internally identified, the email notifications for the known issue went to an old distribution list and therefore was never picked up by the appropriate team. Additionally, the databases were not segmented from the remaining network allowing the attackers to pivot to other servers where they found unencrypted credentials allowing them to escalate the attack.

It's easy for us to sit back and pick apart the lack of patching and other security controls in this case. However, two months from initial disclosure to exploit is a short window for many organizations. Some of these same organizations may have technology running that hasn't been patched in a much longer timeframe. More importantly, the time to exploit these days is much shorter where attackers are able to gather enough information to reverse engineering patches and build exploits in days or even hours. But the security person in the development team has the responsibility to jockey with the rest of the features that are slotted for a release and ensure that the application is protected. Given that the Apache Struts vulnerability was only two months old when it was exploited, there were most likely older and more critical vulnerabilities that were already on the team's plate ensconcing this Struts vulnerability in the annals of security history. This also highlights the earlier fact that every company is a software company. Equifax is not a technology company by trade but yet they find themselves in the software business due to being powered by layers of software that enable them and their customers.

1.5 Is going left better than going right

As I mentioned, shifting security to the left will result in better outcomes for an organization. It allows the development teams to build a culture of security that is more sustainable and able to manage the "when" not "if" of security vulnerabilities that are sure to be introduced.

No software is in any organization can be written to be 100% secure for all time. There will be vulnerabilities. The organization needs to have the culture, processes, people, and technology in place to manage this.

There is no wrong or right (no pun intended) way of approaching application security. The people, process, and technology related to application security is needed throughout the lifecycle regardless of the stage and its purpose. To put it into context let me describe two different organizations and their approaches to application security.

The first organization called Acme Services has decided to engage the application security team prior to release to perform simple scanning and penetration testing to make the determination as to whether there are any vulnerabilities introduced into the product during development. This is the shift-right approach that has been described previously. The second company, Superior Products, knows that brining in security earlier is not just easier, but more cost effective than waiting until later. They requested that the application security team is engaged earlier. Even better, they have a security champion, Dashing Danielle, in their team that can provide guidance throughout the development process.

Superior Products has a mature application security team that keeps their finger on the pulse of the security industry. They've integrated a security champions team across the organization and maintain open communication with that team to ensure that they are kept informed of changes in the industry. With this information and structure, they are able to perform threat intelligence that informs decisions on new requirements and technology. In many cases, this allows them to implement security features ahead of client requests to do so. Acme Services is often caught off guard by requests for stricter security and privacy from their clients due to the fact that they have decided to take the approach of implementing security later in the process.

Superior Products employs Dashing Danielle to review the use cases that come in from product ownership. She is able to perform a quick threat model on the feature to determine the open security concerns that impact the feature. Based on a risk assessment that she has done previously with the product and the application security team she knows that the application and the information in the report is considered sensitive information for the organization. This means that she will want to create security requirements that maintain the confidentiality of the information that is contained with the data and ensure that access is limited to a small audience of users within the appropriate organization.

Dashing Danielle is able to raise these questions about the access to the reports during the review of the user stories and requirements based on the information she has. After she speaks with the application security team on what she feels are concerns with the new feature she presents the following requirements that help protect the data in the generated reports as well as maintain access control:

- Security Requirement: "Application shall ensure that access to the report is limited to authenticated users of the organization that the report belongs to."
- Security Requirement: "Application shall log information related to the admin accounts that create and change the reports."

The development team agrees that these are important requirements and are capable of making them a reality without impacting the release timeframe for the product feature.

Dashing Danielle supports the development team in whiteboarding the workflow so that the requirements are clear and understood by the development team.

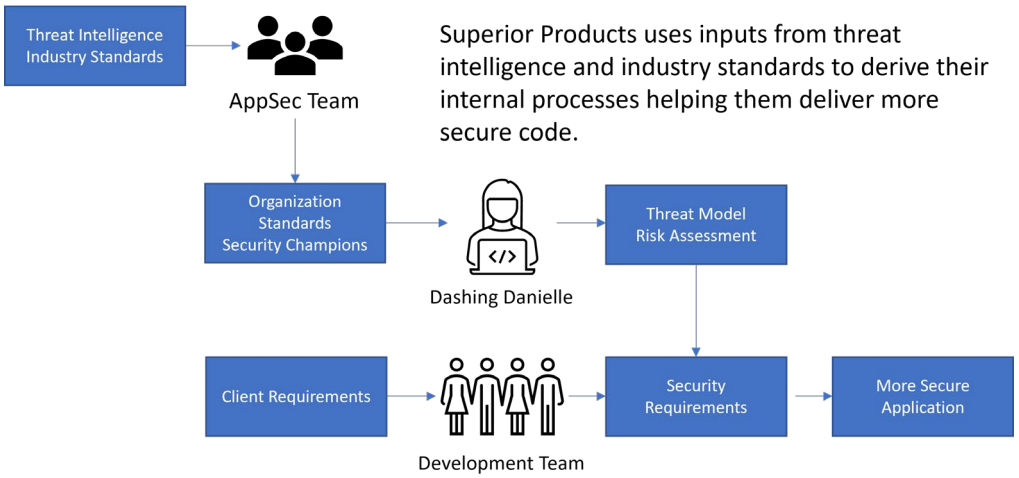


Figure 1.6 Superior Products path to more secure code

Over at Acme Services, one of the developers raises some concerns about whether unauthenticated users could access the report. This is quickly dismissed since the product owner promised this feature in a short time window, and the development team doesn't know whether the data in the reports is actually considered sensitive. Everyone shrugs and moves on.

This is a pretty clear distinction of where application security works and doesn't. It may seem like this story is far-fetched, but please understand that this type of story is typical. Picking up this book means that you want to be more like Superior Products.

1.6 Application security needs you!

It takes a village to do anything worthwhile. Security is no different. Generally, security teams are a slim percentage of the overall organization and relies mostly on automation and the goodwill of the engineering organization that it works with in order to scale to meet the demands. There is no "correct" size of the application security team, but size is not indicative of effectiveness. The variance in the size of application security to engineers varies from organization to organization.

> **DEFINITION** Building Security in Maturity Model (BSIMM) is a study on the posture of software security initiatives and programs by quantifying the application security practices of different organizations across industries, sizes, and geographies.

In BSIMM they define the software security group (SSG) as the team that is focused on software security within an organization. They found that an SSG group can be as large as 160 or as small as 1 with the average size of the team as 13.9 people. This of course depends on the size of the organization and the amount of coverage that the software security group needs to manage. And it's no surprise that application security continues to be a smaller part of the overall engineering organization when you see the total spend for security relative to the overall IT budget. In most cases this is around 5-6% based on a Gartner study in 2019. Application security is yet a smaller portion of that security budget since much of the funding goes to perimeter defense as well as detection and response capabilities.

EXERCISE Take a look at BSIMM's website for SSDL Touchpoints:

https://www.bsimm.com/framework/software-security-development-lifecycle.html

Based on what I've described about shifting left and shifting right, these touchpoints are all part of the shift-left model. If you were in an organization with limited budget, where do you think the best place to put you and your team's focus in order to build security into the development lifecycle? Think about the implications of the architecture analysis. This takes resources to be on the ground level. Security testing can scale but can be expensive and take time to implement.

1.6.1. Democratizing application security

Application security is less about a dedicated team and more about building the habits, culture, and infrastructure to support secure development. An application security team, regardless of size, relies heavily on others within the development organization to socialize and promote security within the broader organization. It is not possible for an application security team that is relatively small to be able to be integrated with every development team and be a part of every design decision. Without this borrowing of resources from the development organization, application security would rarely be integrated. It is critical for the advancement of application security to be able to find allies, build trust, and democratize security with the overall engineering organization. Throughout this book I will outline the methods used to achieve that advancement however, to be clear, application security requires a culture of security and requires buy in at all levels.

The critical part here is the helpers. Some organizations call them champions, evangelists, or coaches. The theme is the same regardless of what they are called. We will talk more about a champions program in future chapters; however, the basic principle is that these champions are the connection between the engineering organization and the application security program. They are there to represent the interests of the application security program as well as ensure that security standards, designs, and architecture are properly implemented in the areas that they represent. The champion is usually a senior or well-seasoned engineering resource within an application or business unit and comes from within the engineering organization. It is important for the champion to be there because they want to, and not because they have to since a successful champion will be one that appreciates security.

These champions help the application security team advance security by being present where the application security team cannot be. Many decisions get made at the stand-ups, in the hallways, or in impromptu conversations between developers. Having eyes and ears that are closer to where the code is developed helps ensure that security is considered in every part of the software development lifecycle.

One condition of a successful democratization of application security is to ensure that these champions are well versed in the organization's security culture, know where to find information related to requirements, standards, and architecture, and can ultimately feel comfortable speaking for the application security team. There may even be a formal training and assessment program before a champion can assume the role. To ensure these champions have the information that they need to be successful, the application security team needs to publicize their documentation and guidance and review and gain consensus on new items with the champions on a regular basis. This can be done in a formal reoccurring forum, or through electronic forums. It depends on the organizations culture and the most practical way of reaching an audience.

1.6.2. Users will be users

Champions are not enough to augment the application security team. It's not that rare to hear a developer, an end user, a leader, or other technologist say that "we have a security team for that". We can build all the technical security controls into a system, but once a user contacts a system an element of unpredictability is introduced. Users want to get their job done or find a way to complete some activity. We have all been in the situation where we were halted by system limitations due to security controls. Most of us do not throw up our hands and walk away. We look for other ways to complete what we set out to do. Users are doing the same thing with the applications you are building, they look for ways to accomplish their work, regardless of whether the application allows it or not. Additionally, malicious users are probing the application looking for weaknesses. They are not staying within the confines of the application that you developed. In fact, they are looking to do the exact opposite of what your application is designed to do in order to create an error condition that they can take advantage of. Application security is always working to keep up with the curious user and the malicious attacker and no amount of tools will give us the level of comfort we are looking for. It requires a village dedicated to the security of the application that is devoted to building a secure application from the start.

I stated at the beginning that every company is a software company which means that every company needs some level of software security. Likewise, software security is everyone's problem. This is a pretty common refrain, but what does this actually mean. This is similar to the public service announcements urging people that if they "See something, say something". This doesn't mean that a traveler should attempt to open a bag that they find in an airport terminal that has wires hanging out of it and is beeping. This means that the traveler should alert the nearest authority so that they may investigate. Nobody in security is asking for an end user to triage a SQL injection flaw in a web application and write code to resolve it. They're simply asking to alert the nearest champion, or security personnel. Leave the bomb disposal to the professionals!

1.7 Examples of failing to secure the software

The news is littered with examples of failures in security where applications were compromised for a slew of different reasons. The point of covering some examples is not so we can point and laugh at these organizations, but rather to focus on the how and why. Trust me, if your organization sells or uses software, it's not a question of if, but when you will encounter a security incident. It might be a small and non-reportable event, but your organization will have a software security incident somewhere, sometime.

> **EXERCISE** Go to your favorite search engine and look up articles on the latest security breach. You can also go to sites like www.threatpost.com or www.securityweek.com for recent stories. I'm pretty sure that if you don't see one for today, you'll at least see one for this week. Dig into the article and put yourself in the shoes of the security organization. How would you have responded? Speculate on where things went wrong if they are not spelled out in the article. I often look at these stories and ask myself 'how could myself or my team have avoided this?'

1.7.1. SolarWinds

SolarWinds was a chilling example that showed how a complex ecosystem of software, and the various components that make up that software, can lead to a massive and impactful breach. This type of compromise is commonly called a supply-chain attack for good reason. Attackers can compromise just one component used to build the overall software and have their malicious payload spread far and wide. This takes advantage of a fundamental complication with the supply-chain in that there is inherent trust between the developer of the primary software and the third-party software that is used to build the final product.

SolarWind's Orion product is a monitoring solution that is used to monitor an organizations network and applications. At the time of the attack in 2020, SolarWinds was being used by a majority of the top organizations across the United States including most of the Fortune 500 companies, universities, and many of the top government agencies as well as the military.

Attackers were able to modify a plug-in called SolarWinds.Orion.Core.BusinessLayer.dll. The attackers used a tool called SUNSPOT that allowed them to inject a malicious version of the dll and was digitally signed by SolarWinds and sent to thousands of customers. The malicious dll contained SUNBURST which allowed the attackers to communicate with command and control (C2) servers. The attackers went through great pains to hide their activity by lying in wait for a period of time before retrieving and executing commands as well as masquerading as legitimate Orion traffic and using block lists to identify forensic and anti-virus tools so they could evade them.

Once the plug-in was within a target system it would use a delete-create-execute-delete-create pattern that would hijack a legitimate task, run malicious activity, and then revert back to the legitimate task. This type of sophistication further shows how far the attackers went to keep their activity quiet.

The campaign went on for several months before it was eventually detected when the attackers compromised FireEye, one of the leading cybersecurity companies. The attackers

gained access to FireEye's attack simulation and other security related tools but were spotted which led to the detection of the more widespread activity.

Attacks that are this sophisticated are difficult to defend against. Encrypting there and scrubbing here will not suffice. The vigilance of FireEye, the ability for SolarWinds to rapidly produce a patch, and the impacted organizations aggressive patch management shows how our digital world has changed to where we can no longer rely on putting our efforts into protecting our software, but we need to pivot to a world where we always assume breach. It's how we respond and more importantly, how rapidly we respond when something malicious is detected that makes the difference.

1.7.2. Accellion

For those of us that own a home or any other type of item that requires maintenance and constant attention we sometimes willingly walk past that noisy appliance, or creaky door thinking that one day we'll fix it.

Software is little different. There are many reasons to keep old software running. Clients insist on continuing to integrate with older software and make it difficult for organizations to decommission it. Organizations hold on to old software because it's cheaper to keep it running than to upgrade or replace it. Regardless of the reason, old software is prevalent in almost all organizations.

Accellion develops software for healthcare, financial, and education organizations. Their File Transfer Appliance (FTA) product is used by healthcare organizations to perform large file transfers. The product was almost twenty years old and nearing end of life when it was the target of a cyberattack at the end of 2020. The attackers first stole data from Accellion and then pivoted to attack Accellion clients directly with the goal of stealing data and extorting money. The initial attack was leveraged using a SQL injection attack against the document_root.html file which allowed them to retrieve keys to generate legitimate access tokens. These tokens were then used to access the sftp_account_edit.php file where the attacker was able to then exploit an OS command injection which allows the attacker to make commands to the host system. This last piece gave the attackers the ability to create a shell. The attackers at this point had the ability to upload more sophisticated tools and begin siphoning information and pivoting to customer servers.

Accellion's healthcare clients were left to notify the affected patients, the media, and the Department of Health and Human Services due to the HIPAA Breach Notification Rule. Lawsuits followed and, in the end, nearly 3.5 million patients had their protected health information (PHI) stolen. What made matters worse was the attackers then sent threatening emails to students at UC Davis after the university discovered that their information was part of the breach.

The impact of an organizations inability to decommission old software is wide. It is also not rare for organizations to be mostly running the latest and greatest version of software but have one client running a version of software that is several versions back. This is a failure of product ownership to move clients forward and leaves both the organization, and the customers they provide service to at risk. It is critical for an organization to have, and stick to, a sunset and end-of-life policy. Two key takeaways:

- Decommission of software is the final stage of the software lifecycle
- Remember that technical debt is security debt.

Fun Fact

Microsoft announced the end-of-life date for Windows 7 as January 14, 2020. When January of 2020 came, the much-loved OS was still being used on 39% of PCs. Many speculate this was due to the lack of desire for end-users to move to Windows 10. This shows that even Microsoft has issues coaxing users to newer versions. Instead, they take a more forceful approach by eliminating patches for the older versions. Even then, this doesn't stop users from using it, and Microsoft even allows users and corporations to pay a fee to continue to receive updates.

1.7.3. Fake software

SolarWinds showed how attackers can take advantage of the trust between components that are used to build a final product. However, this story is not unusual. Attackers are always looking for ways to get software into the supply chain in order to maximize their reach. Why try to compromise one organization by specifically targeting them, when you can get into the supply chain and compromise multiple organizations.

There are other ways into the supply chain. Bigger ways if you can believe it. In 2021 two students from the University of Minnesota released a research paper on what they called "hypocrite commits". These commits were supposedly intended to provide value to the Linux kernel but instead introduced critical issues. Sort of like when your parents would hide vegetables in your meals to get you to eat them. Maybe you do this too your kids too. There is no shame in that!

This sort of commit was not well received, not only from Linux, but from the broader security and engineering community. The Linux kernel is used by billions of systems around the world from the smallest to largest computers. The two student's actions led to the ban of the University from contributing to the Linux kernel in the future. They also had their previous commits to Linux revoked.

The open-source community depends upon the submission of high quality, well vetted, and in good-faith, commits to its open branches. Although the example of what the two students from the University were able to accomplish shows that this system can be abused.

The explosion of open-source software that is used to build an overall application further exasperates this exposure for organizations. According to the SonaType State of Software Supply Chain Report in 2020 1.5 trillion open-source software components and containers will be requested by developers. Most software is an amalgamation of third-party libraries, code from software forums, and hand-coded logic by a small group of software engineers and architects. Per Synopsys Open-Source Security and Risk Analysis report in 2020, 70% of an average application is made up of open-source software. Reliance on the third-party software leads to exposure to malicious actors getting into the supply chain and adding nefarious code. Organizations that have this level of third party, open-source software need to take a defensive approach to managing their supply chain through scanning of third-party libraries for vulnerabilities, only using libraries from a reputable source, and maintaining a robust patch management program that allows them to rapidly patch a vulnerable library as soon as a vulnerability has been identified.

1.8 Summary

- Application Security teams generally come late to the party which leads to findings that get moved to a backlog.
- This backlog continues to grow as security debt.
- Scanning tools used by the Application Security team are generally detection tools and do not remediate or block bad code.
- Protection tools can be enabled but are sometimes hard to sell to engineering due to concerns with blocking legitimate traffic.
- Shifting right is catching defects late, while shifting left will find them earlier.
- Shifting left involves more than just training and champions.
- Fixing issues in production is significantly more expensive than fixing prior to production.

2

Defining the problem

This chapter covers:

- Defining the security tenants that software needs to adhere to
- Identifying and understanding risk that impacts software
- Overview of security in the software development life cycle

In the previous chapter, I used the example of building a house without the locks on the doors and windows. Using a house is a great example as it allows you to think about the controls you use to limit your risk of the house being compromised due to break-in, fire, flooding and so forth. We spend most of our time in security attempting to limit risk and counter threats, not eliminate them. Risk and threats can never be eliminated. Similar to a house, you can't eliminate the risk of fire, flood, or a break-in we can only detect and respond.

Risk

Risk is the potential for loss of an asset or damage to an asset.

Threat

Threat is the activity that takes advantage of a weakness in an asset.

Example: In the case of a house fire, the fire is the threat, the house burning down is the risk.

Whereas fire, flood, and break-in are risks that impact a house, our software has a different set of threats and risks. These range from the physical to the digital. Yes, physical threats exist that impact our software, a flood in a data center would be a physical risk to our running application.

In this chapter we will not dive into the various specific methods of protection, but rather outline the different places where our software and data need to be protected and some best practices to be on the lookout for.

2.1 The CIA Triad

There are three basic tenets in security that all protection mechanism that we integrate into our systems and software will come back to. It's commonly referred to as the *CIA Triad*. This stands for Confidentiality, Integrity, and Availability.

- Confidentiality – Protecting data and only allowing those that should have access.
- Integrity – Data is known to be correct and trusted.
- Availability – Systems and data should be available when requested by a trusted entity.

Most people in the security field are pretty familiar with these concepts, but I will provide further definition here to provide common ground for future topics in this book.

2.2 Confidentiality

Perhaps the most important security design consideration you can make when developing software is *confidentiality,* where you are designing the software to provide optimal protection of data. The reality is that this is much more difficult than it sounds. As I mentioned, data is the new commodity. This means that it is everywhere. Software may be processing data that it is not the custodian of.

> **Data custodians are responsible for the collection, processing, storage, and implementation of business rules related to the data.**

Some data is transient, some data is not persistent in a system and reside in memory for only a period of time. Figure 2.1 uses a simple data flow diagram (DFD) to show you how data moves through your system and others.

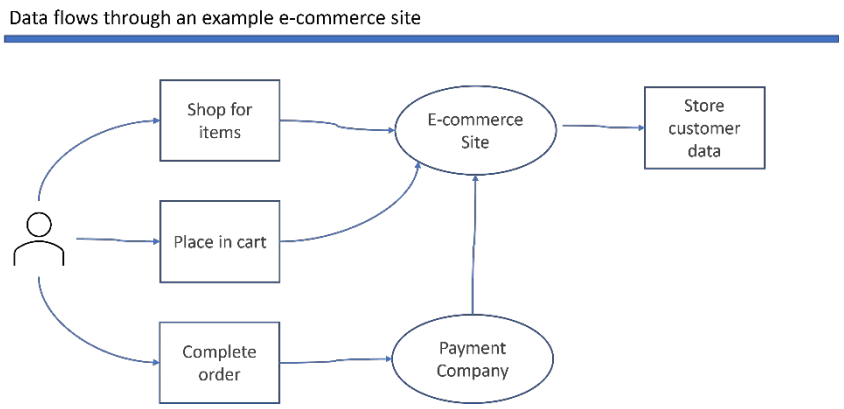


Figure 2.1 A basic data flow diagram (DFD) for purchasing online

In Figure 2.1, once the customer completes an order, a series of other processes will coordinate with the payment company, process the order, and store the data. At each point, the protection of that data is paramount. Especially if it is sensitive data like payment information. Today, we have methods to protect this information, primarily through encryption at rest when it resides in a database or other system, encryption in transit as it moves over the network, and encryption in use as it is processed in memory. These methods allow us to provide some protection of data as it traverses our systems, and in some cases even the services that your software sends data to.

2.2.1 Data protection policy

The first step an organization needs to take in order to maintain data protection is to create and maintain a policy that clearly outlines what data must be encrypted. Most organizations will call this their data protection policy and will define what data needs to be protected but will not get into the how aside from a high-level direction. The data defined in this policy will follow the organizations data classification scheme (more on that later) and call out that data must be encrypted, and access is limited to only required personnel. Most policies will additionally include the following:

- Clear definition of what must be done at each location that data can reside such as file system, end user devices, databases, and others.
- Definition of the encryption key lifecycle such as creation, distribution, and destruction.
- Define algorithms for hashing and encryption.
- Auditing of access to encryption keys.

Once this policy is established, the organization can get to the business of mapping encryption standards and architecture to this policy but must give thought to two primary considerations when developing these standards and architecture:

- Although encryption provides extra protection, there is additional time that is incurred when you need to access a key, encrypt data, and send it along its way. This latency is usually measured in milliseconds, but for critical applications this additional time needs to be considered.
- The system must also ensure that it is resilient enough to be able to overcome an outage that may occur due to the encryption keys being unavailable, or a failure in the encryption or decryption process such as the cryptographic service being offline.

This adds an additional layer of operational complexity and potential exposure to an availability issue and must be considered.

EXERCISE

Go through the assignment on Symmetric Encryption and get familiar with the way a single key can be used to encrypt and decrypt data:

https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:data-encryption-techniques/a/symmetric-encryption-techniques

Go through the assignment on Public Key Encryption to get familiar with key generation as well as encryption and decryption using a key pair:

https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:online-data-security/xcae6f4a7ff015e7d:data-encryption-techniques/a/public-key-encryption

2.2.2 Data at rest

One of the first steps to protecting data at rest is to recognize that you have a problem— a data inventory problem. Put simply, data inventory is the ability to know where data is located in an organization. During a security incident, an organization needs to be able to know what data may have been compromised and whether there is a risk related to the data that was compromised. Is it critical client information that was on that database that was just breached? If this can't be answered then there is a gap with understanding where the data is and what the nature is of that data. Review figure 2.1 again: Do you know what is happening to that information that is passed between your application and the payment company? Do you know whether they are sharing data with others, knowingly or not? Do you or the consumers of your application have a policy that disallows developers to copy data from a production environment to a lower environment? If so, is it enforced?

These examples are not intended to frighten, but to bring awareness to the fact that data is everywhere and that the first step in securing this data is to get an inventory. For smaller organizations, this can be done by keeping a catalog of data, location, and classification in a spreadsheet or database. More on classification in a bit. For larger organizations, a tool may be more appropriate that can actively scan for data, classify it based on the organization's rules, and provides reporting and dashboards to track the organizations overall data. However, regardless of how you identify your data, there is still fundamental steps in order to have a robust data inventory, including the following:

- Create an inventory, using an automated tool or the manual process described previously.
- Data inventory needs to include all data, structured and unstructured, across on-premise, cloud, and third-party locations.
- De-duplicate the data and ensure accuracy.
- Ensure that the inventory is maintained and kept up to date through the tools and processes used to initially gather the data.

Although there are tools available to assist with collecting a data inventory, creating this inventory is more of an organization and process problem than a technology problem. In lieu

of a tool, one method is through surveys or questionnaires that are used to determine the type, classification, and location of the data. These questionnaires are simple in nature and should be completed by the data custodian who is responsible for the technical implementation and maintenance of data within the system. Each organization should create their own survey that attempts to determine the type of data that is moving through the system and should focus on the sensitivity of the data that aligns to classification that the organization uses. They are also responsible for ensuring that this survey is kept up to date as the architecture changes. Below is a sample of what a questionnaire might contain.

Does the application retain payment card information?	Y/N	If yes, list the locations where the data is stored.	Data encrypted at rest?
Does the application retain social security numbers?	Y/N	If yes, list the locations where the data is stored.	Data encrypted at rest?
Does the application retain protected health information?	Y/N	If yes, list the locations where the data is stored.	Data encrypted at rest?

If your organization is like many large organizations today, there will be a data lake or data warehouse that is a single location for most of your structured and unstructured data that all of the organization's applications can leverage. This certainly makes things much easier when attempting to locate and classify data considering that it is centralized and easier to apply tooling or processes to collect where sensitive data is located. In fact, many cloud providers offer the ability to automatically identify and classify data through the services they offer in their ecosystem.

> **NOTE** Macie is a service offered by Amazon Web Services (AWS) that uses techniques like machine learning and artificial learning to detect sensitive data in AWS. Currently it has some limitations on where it can find data within its services, but this is expected to improve over time.

However, there will always be legacy systems that are disconnected from the data lake, and applications that need to maintain local copies of data in order to operate. It is hard to imagine any large organization having their data exclusively in a central location. Which means that the organization will still have to rely on questionnaires or on-premises tools to located and classify data.

Regardless of how an organization inventories data, they then need classify it. This allows the organization to provide the appropriate level of protection based on the classification of the data. Although it varies depending on the industry and needs most organizations will have four ways that they classify data.

Table 1.1 Basic data classification in most organizations

Column heading	Column heading
Public	This is freely accessible to anyone including individuals that are external to the company. An example might information in a marketing release.
Internal	This information is intended for just individuals internal to the organization. This doesn't classify the access based on role, the individual just has to be a part of the organization. This could be communications regarding business plans.
Confidential	This is information that is sensitive and requires authorization to access. For instance, certain data that is in an internal database like social security, or account numbers would not be accessible to everyone in the organization and is limited to those that require access. This should be a very limited audience.
Restricted	This is critical information that could lead to severe damage to the organization should it be released. This is something like source code or other intellectual property.

Having this level of classification allows the organization to apply, broadly, encryption methods based on the classification. For instance, the organization may require that all Confidential data be encrypted at rest and transit and that all Restricted data be shared only with a small audience of people in the organization with a need to know.

2.2.3 Applying encryption

So, you've identified where your data is located, and you've used the methods described previously to classify it. Now what?

Encryption

We will talk more about encryption as we progress, but a simple way to think about it is the ability to change plain text to cipher text through an algorithm with an encryption key. One main point to remember is that encryption is reversible so that either the same key or a paired key will be able to decrypt. This is by no means a book on encryption. Trust me, I can't explain it like a cryptographer. However, one of the preeminent books on encryption as it applies to engineering is Applied Cryptography by Bruce Schneier. If you are looking for some more basic information regarding encryption, look at the Wikipedia on encryption that follows the history, future, and uses of encryption (https://en.wikipedia.org/wiki/Encryption)

To encrypt properly you need a few things. You need an encryption and decryption key (sometimes the same key), a secure method to create an encryption key, and a secure location to store the key, and a way to distribute and access the key in a programmatic way.

Depending on the architecture and deployment of your application the ability to securely generate and store a key may be limited. Most legacy systems are limited in their ability to store an encryption and decryption key. The key or keys are often stored in a configuration file, or worse, hardcoded in software. More modern architectures will take advantage of a Hardware Security Module (HSM) that allows the application to make secure API calls to a physical device that stores encryption keys and can be configured to auto expire, rotate, and generate keys. Today, many organizations are under regulatory or contractual obligations to provide better encryption methods such as the use of an HSM to encrypt sensitive data and even allow for their customers to manage their own keys. However, although an HSM will provide the highest level of secure access to data encryption and decryption keys, they are expensive and can be a single point of failure if the hardware becomes unavailable.

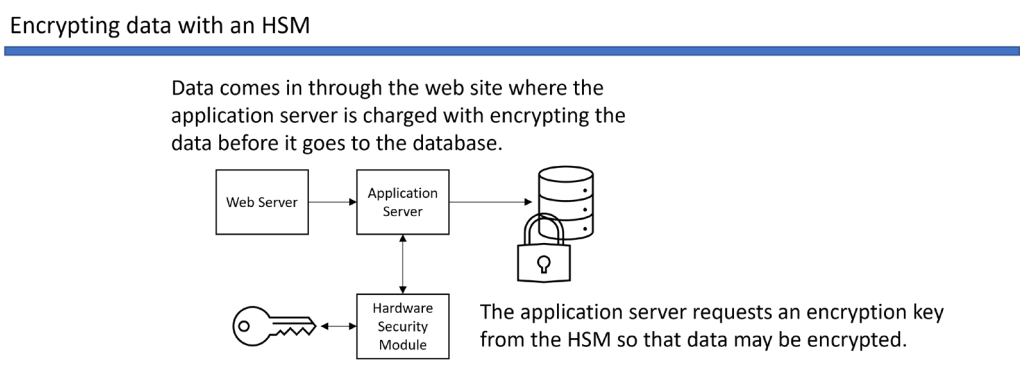


Figure 2.2 – HSM in a simple workflow to encrypt a database

Depending on your organization and the industry you are in, you will most likely have to encrypt at least restricted and confidential data. For instance, in financial organizations, datasets that contain account numbers, social security numbers, and similar data will have to be encrypted based on compliance requirements. In the healthcare industry, you will have to encrypt insurance information, and patient record information. One of the considerations for these organization is that information could be located outside of a database and could instead be in unstructured data like text documents, PDFs, or image files. For instance, a printed form from your doctor's office may have your social security number and would be scanned and uploaded to a health information exchange (HIE). Organizations have a few options when dealing with structured and unstructured data encryption:

- Full Disk Encryption to provide encryption of the disk that the data sits on. However, this does not give you more granular encryption at the file level.
- File level encryption that encrypts the individual files that hold sensitive information like confidential and restricted data.
- Column or row level encryption that encrypts a set of data in a database like the social security number, or account number column.

Proper application of encryption is not as easy as waving a wand. Based on the location of the data, the ability to encrypt might be limited or infeasible. Legacy systems often struggle to access more modern encryption technology like HSMs. However, the organization needs to constantly consider what might occur if data is exposed. This can happen physically when a database drive is improperly disposed of and falls into the wrong hands, or digitally through an insider or malicious activity that is able to access the data in the database through the system. Some examples of data escaping the organization are:

- Improper destruction or retirement of old drives that contain sensitive data.
- An insider of the organization could have access to the database through improper access management.
- An attacker could gain access to the host that the application database resides.
- A developer could copy data from a production database to their local environment or a lower test environment instead of using mock data that is more suited for these non-production environments.

With the mindset that the data in the organization could be exposed to an adversary at any time means that the organization has to ensure that they have applied the appropriate level of encryption based on the classification of the data. If the physical drive that the database is on or a device with access to the database falls into the wrong hands, does the organization have enough confidence that its security controls will work? Can an adversary or competitor access the data on that database, or has the organization applied the proper level of encryption so that it cannot be accessed?

In addition to proper encryption techniques the organization needs to ensure that there are proper access controls around the data at rest and access controls to the keys that can encrypt and decrypt. If a user has access to the database management tools, they will most likely have access to view the plain text of the data in the database. Additionally, if a user has access to the encryption or decryption keys, they are also able to view this data or encrypt data as if they were a legitimate service. Traceability of these sensitive activities can be met by using tools that provide access to the privileged account while also providing auditing capability. Usually, the auditing within the tool is completed through screen activity captures, key stroke logging, and auto filling passwords so that it's not accessible to the end user. In this scenario an organization will know who performed sensitive activity, and more importantly, what they did while provided that access.

2.2.4 Data in transit

The job of protecting information would be much easier if we had to protect data only when it is sitting nice and still in a database or a file system. The reality is that data moves. In fact, the exchange of data is how organizations make money. It's how organizations gain competitive advantages. In short, data needs to be in motion in order to be profitable.

It wasn't that long ago that most web sites were using basic HTTP or FTP to transmit data. Using plain HTTP or FTP means that as the data moves from one location to another, it is in plain text and open to view by anyone that has access to see the traffic. Secure Sockets Layer (SSL) was the first attempt to bring encryption to data in transit, turning HTTP to HTTP over SSL (HTTPS). Although version 1.0 was never released due to security concerns, 2.0

was released in 1995. Since then, we have had many versions of SSL and then Transport Layer Security (TLS) that replaced SSL. TLS is primarily used to protect HTTP and FTP traffic.

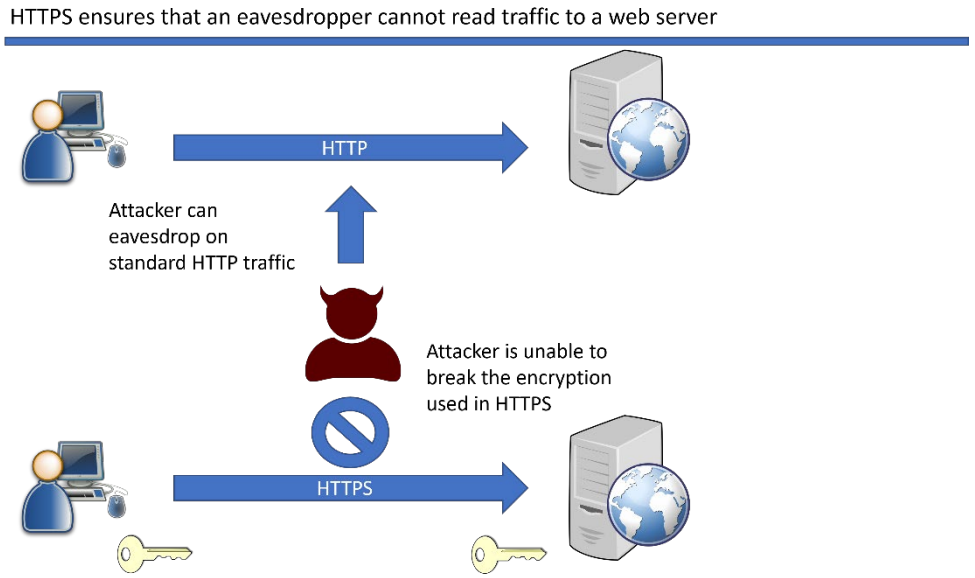


Figure 2.3 HTTP vs HTTPS

Prior to the time when everything was HTTPS, sites used the secure transit only during sensitive activities like login. Today, almost every website you go to is now TLS enabled and you will be redirected to HTTPS if you attempt to go to a site with just HTTP. Several factors played into the adoption of TLS. The browsers played a large role in this as they began to mark sites "insecure" if they did not use TLS or even if they used a known weak version of TLS. Additionally, security over the years has been become not just an obscure corner of an organization, but increasingly a concern for everyday users. Users may not know, or want to know, how TLS works. But they do want to know that when they input their credit card number into a website, that the number will be protected. Most users today are becoming accustomed to making sure that they see some version of a padlock or other indicator that the site they are on is using a secure connection.

The great thing about TLS is that it is easily enabled. This is completed through a simple configuration in the web server and a signed digital certificate from a trusted Certificate Authority (CA).

> **DEFINITION** A public and private key pair, also known as asymmetric key pair, is a set of digital keys that are used to encrypt and decrypt data. These keys are a generated together, but one is intended to be kept secret and private, where the other is used publicly. They are used to encrypt and decrypt, and provide digital signatures,

A private and public key pair is generated on the web server. The private key is retained by the web server and ideally, never leaves the server. The public key is used to create a certificate signing request (CSR) that is then passed to the CA to sign with their private key. The product of this is a public certificate that is used to show that the web server has control of their private key and has been verified by the CA.

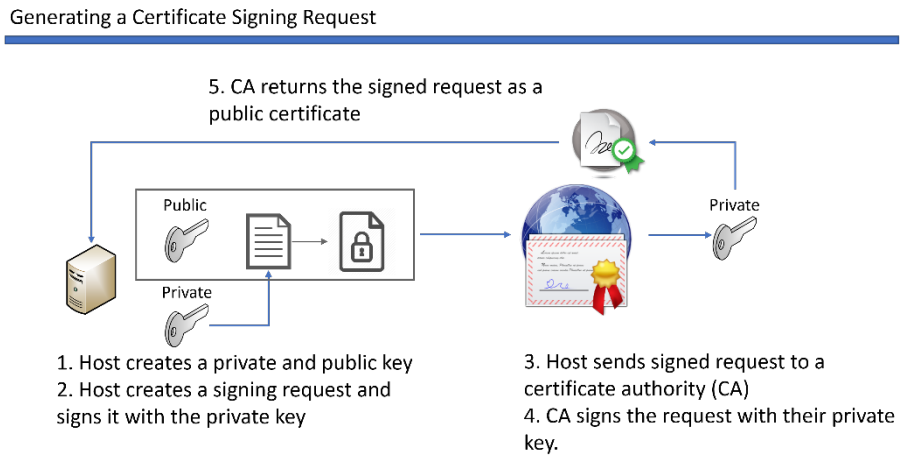


Figure 2.4 – Generating a certificate for HTTPS using Entrust as the CA

Once this certificate is signed it is used to announce to any user agent, like a web browser or command line HTTP tool, that they can trust the web server. The user agent will look in its trust store and determine whether the certificate authority that signed the certificate can be trusted. If the certificate is signed by a well-known CA, the browser will already have a trust relationship and will accept that the web server is trusted. In some cases, if it is not a well-known CA, the user agent needs to establish the trust by placing the certificates in its trust store.

Certificate management is fun. Said nobody ever. As you can tell, enabling TLS can be mostly simple, but there are operational costs. Similar to the key management problems discussed earlier, certificates have a lifecycle and need to be renewed or replaced periodically. Additionally, some of the fundamental underpinnings of the technology can occasionally change and lead to the need to make drastic changes. In fact, in 2016 browsers began to display security warnings for certificates that were generated using the SHA-1 hashing algorithm with users being able to click through the message and continue to the site. By the end of 2018, browsers like Chrome were alerting users that they would disallow access to sites using the less secure SHA-1 at the beginning of 2019.

I had the pleasure of leading a project at a large organization with the directive to update all the certificates we had to SHA-2. This was no small task. Identifying all the use cases where these certifications were used like single sign-on SAML certificates, TLS, FTP, digital signing, and others was difficult. Additionally, locating where these certificates were stored

was a further complication. They are found on file systems, key stores, web servers, and databases. Changing them without creating an outage required tight coordination between the development, security, client, and operational teams. Inevitably, there were rollbacks when something did not go as planned.

Never fear though, as with encryption keys there are tools that can be used to manage, rotate, issue, and alert on the expiration of certificates. There are also protocols such as the Automated Certificate Management Environment (ACME) that can be used to create CSRs and manage keys and certificates. It is part of the business model for Let's Encrypt certificate authority which allows them to issue certificates quickly, efficiently, and provide short expirations of around 90 days. This short expiration date provides the ability for web servers to frequently change certificates and reduce the amount of time that a certificate is in circulation. Not that long ago, certificates would have an expiration of five, ten, or even thirty years. For the sake of security, the industry has begun moving away from this practice and is looking for much shorter timelines to provide better security. This only works effectively if there is automation in place to make the rotation of expired certificates a smooth process.

2.2.5 Encryption prior to transmission

A method for ensuring that data is secure when sending between two organizations or two applications in an organization is to encrypt locally before sending the data. This does require the two parties to have either a shared encryption key that they have already determined or to use a key pair that allows one party to encrypt and the other to decrypt. The data will be transferred between the two entities and is protected regardless of whether it is transmitted over a non-secure channel like HTTP. However, you still have a dependency on the key management. Transmitting the key opens each end to a possible compromise. Another consideration is if the key is compromised there needs to be another exchange and re-keying effort by both entities.

Pretty good privacy is a well-known tool developed in 1991 that is used to provide encryption prior to transmission. It can be used for email and file transmission. PGP can use either a single encryption key or a private and public key pair. The latter offers the ability for two entities to send information to each other even if they never met or don't have the means to exchange a single key.

2.2.6 Data In Use

Where encryption in transit and rest get a lot of attention when it comes to protecting data, doing this for data in use is much more complex. Protection of data in use, means when it is being accessed in non-persistent states like memory or in the CPU. Additionally, this means that data is protected throughout its entire lifecycle when combined with encryption of data at rest and transit. In general, well-designed systems will ensure that access to data in use is only accessible to the parts of the system that should have access. This will limit access to memory by malware and other processes. This should protect against not just writing and reading, but also executing of code when not allowed.

Many of the methods for protection are through the host level or operating system level controls like using segmentation, protection rings, or paged virtual memory. Additionally,

address space layout randomization is set at the operating system level where memory is randomized to limit the ability of malicious activity from finding specific addresses to jump to.

Enclaves can be used at the system level that ensures that the data in use is encrypted and only available to the CPU or CPU cache at read time, but at all other times it is encrypted and not readable to any other parts of the system. One other protection method worth mentioning is the ability for the CPU to manage encryption keys in a register as opposed to the keys being stored in RAM. This makes the window of opportunity smaller for an attacker or malicious code to access encryption keys.

2.2.7 Not so confidential

Hardly a day goes by where a breach or some type of cyber event occurs. A breach of confidentiality means that the data that was intended to stay confidential or restricted was released unintentionally. This can occur through malicious activity, or accidental release.

Email is a quick and efficient method of leaking data. While information contained in an email can be concerning, it's also important to know that by most standards, an email address alone is considered PII. In February of 2020 the father of U.K. Prime Minister Boris Johnson caused a bit of an international stir when he accidentally copied the BBC on an email to British officials regarding the lack of contact between Boris Johnson and the Chinese state over the coronavirus outbreak. Later that year, Australia's Department of Foreign Affairs and Trade exposed 1,000 citizens personal data when an employee failed to use BCC to send information regarding emergency loans and reentry quotas for citizens stuck in other countries due to coronavirus restrictions.

However, where email is an excellent avenue for leaking information, it tends to be smaller than the big breaches. In May of 2021 the company Peloton who makes connected exercise equipment and is most famous for their high-end stationary bicycles, had to release a statement regarding an application program interface (API) that allowed anyone to pull private data from Peloton's servers, even if the users profile was set to private. This API allowed unauthenticated access, meaning that you did not have to have an account or special access. Peloton initially only limited access to the API to users with a Peloton account, however, this only limited the audience, and anyone could sign up for a Peloton account and access the API to gather the personal information of other users. Eventually Peloton corrected the authentication issue fully, but they were not able to confirm or deny whether their over three million users had their data accesses or stolen due to the issue.

2.2.8 Do I even need this?

You may be familiar with Marie Kondo who is famous for her show on Netflix called "Tidying Up." She uses the KonMari Method to identify things that you should keep and things you should get rid of. Often, she asks the question: "Does this spark joy?" What does this have to do with security and specifically data security? One of the first things I usually ask when I get involved with reviewing security decisions and architecture that involves sensitive data is whether the data needs to actually be collected and retained. If you don't need to keep it or act upon it, then you don't need to worry about securing it. To be clear, securing it means more cost and effort. Something that not many product owners will be willing to spend if

they don't need the data. Therefore, limiting the collection of this sensitive data is the best way to secure it. However, organizations and application development teams will err on the side of collecting more data because you may need it at some time, and storage is cheap.

Some Questions that you need to ask regarding data:

- Do you need to store that data?
- Do you have regulations or contracts that require to maintain the data?
- Can you properly classify this data?
- Can you provide the appropriate level of protection for the collected data?
- Do you know when you no longer need it and can destroy it?

If you are comfortable with answering these questions in the affirmative, then you are on the right path. If not, then follow the previous recommendations of classifying, inventorying, and encrypting the data.

> **EXERCISE** If you currently work in a development team or a team that collects data. Take a look at the database schema identify at least one set of data like a column in the database that contains sensitive information. Ask yourself or your team whether that data that is being collected is imperative to the operation of the application and whether not having the data would impact the application. The answer might be yes, in fact odds are it will be yes, but asking these questions regularly will get you into the habit of questioning whether you need to retain sensitive information.

2.3 Availability

Confidentiality takes the limelight when it comes to the CIA Triade primarily because you can get the biggest bang for the buck by properly encrypting data. Availability sounds like something that is more for IT Operations or site reliability engineering (SRE) who are primarily concerned with the uptime of the systems. The reality is that availability can be critical for applications where being down can be a matter of life and death. It's not hard to imagine scenarios where this is evident. Hospitals, emergency services, critical infrastructure, and so forth have high demands for uptime.

Not all uptime demands are for public safety. Organizations that are solely an online retailer stand to lose money should their systems be unavailable. In early 2021 Amazon, the online retail giant, was taking in roughly $830,000 per minute. It's pretty clear that an outage at Amazon would cost the company millions of dollars depending on how long it persisted.

You may not be running Amazon or a hospital but make no mistake that application uptime is still important for the organization and does have a financial impact. Organizations regularly report on their uptime, and it is a critical metric that needs to be met. Most organizations have contractual obligations that require them to meet Service Level Agreements (SLA) otherwise penalties will be incurred. Usually in the form of monetary compensation to the clients.

2.3.1 DoS and DDoS

Most people think of one thing when they hear about attacks that bring down a service or application. DoS, or denial of service, is an attack that purposely floods the service or application with a large number of requests. Its bigger brother is the DDoS, or distributed denial of service where the requests are from many different sources instead of just one in the DoS model. The system is then overwhelmed and is unable to complete requests for other legitimate traffic. One of the greatest complications with protecting against DDoS attacks is that the traffic often looks legitimate which makes the job of blocking this traffic difficult.

There is an old American television show called "I Love Lucy". One of the most iconic episodes was when Lucy and her friend Ethel work in a candy factory to make some money. They were assigned to the candy wrapping station where candy would pass by on a conveyor belt and the two would have to wrap each one. Naturally, at first the conveyor belt is moving along, and the two friends are able to successfully wrap each piece of candy that goes by. Soon the belt begins to move much quicker with more candy coming. The two friends are quickly overwhelmed and begin to stuff candy in the pockets, and mouth while missing most of the candy that goes by.

This is a simple example of a DoS attack where the system can be tricked into accepting what appears to be normal traffic, at high volume, by a malicious actor. In other words, the candy coming down the belt is still legitimate and should be there, but the system setup to properly handle that candy is unable to handle the large amount being sent. Now, in this example, it's easy to laugh it off because it's a simple yet visually accurate example of a denial of service. In the real world these attacks can be dire. Increasingly over the years these attacks have also grown in size. Early DDoS attacks were small in size and would use something like a SYN flood that would send a large number of TCP synchronize (SYN) packets without closing them out. It was simple, yet effective. More importantly, almost anyone could do this. Attacks today are much more sophisticated and larger, reaching over two terabits per second in some cases. To put that into perspective that is 1,000,000,000,000 bits per second, or roughly 1,000 hours' worth of movies.

DDoS attacks are not limited to just network level attacks. Layer-7, or application layer attacks are common as well. These types of attacks are generally smaller in nature as it is targeted at disrupting the flow of the application or processing of data by going after the specific resources that serve up the application. This could be by targeting the database to make it unavailable or flooding the application with HTTP traffic that keeps it from processing legitimate requests from users.

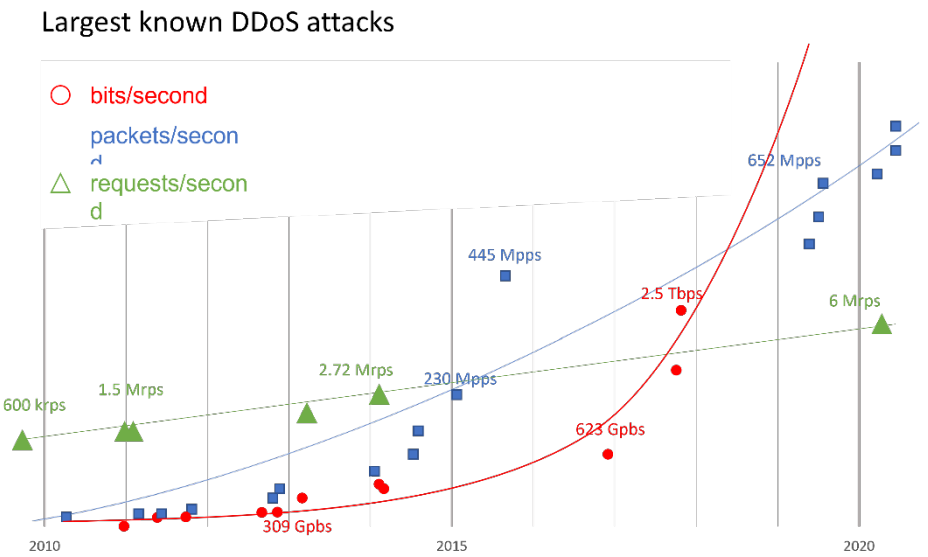


Figure 2.5 – Growth of DDoS attacks since 2010

2.3.2 Accidental outage

Not all availability issues are due to malicious activity. If the application is not built to be resilient, it is possible that something as benign as a software update could bring down an application. Other possible actions are system reboots, patches, or failed software installations. Most organizations will perform system maintenance during times where the client impact is low.

In 2019, while Britain was moving towards an exit from the European Union, many citizens of Britain that wanted to remain in the European Union attempted to sign a petition on the UK's Parliament website. The sudden spike in traffic that was unusual for the site presented many of the citizens with a HTTP 502 error signifying that the site was incapable of handling the large number of requests.

There are also cases where the protection mechanisms in use can create their own availability issues. Akamai is one of the leading companies in DDoS protection. They offer a content delivery network (CDN) that includes protection for many companies against volumetric type of attacks like DDoS. In June of 2021, an outage at Akamai lead to several sites around the world becoming unavailable. Most were financial institutions in Australia and New Zealand, however both Southwest and American Airlines were impacted as well. The cause of the outage was not due to a cyber-attack, but rather a misconfiguration at Akamai related to a routing table value. This came shortly after similar outages at some of Akamai's rivals Fastly and Cloudflare showing that an overreliance on third parties to deliver protection can be an additional risk to an application.

2.3.3 The role of ransomware

Protecting against volumetric attacks that flood your network or application with legitimate or junk data is one thing, but an entirely different approach to availability chaos is by encrypting the devices or data that your application depends on. Ransomware is not new, but in the past few years it is gaining popularity. We could spend an entire chapter on ransomware, but I'll summarize it here.

Ransomware is the outcome of a successful malware attack with the sole intention of encrypting a device (locker ransomware) or the more popular method of encrypting data (crypto ransomware). The methods of delivery of the malware vary from phishing to more sophisticated remote code execution. So, what does this mean for your web application or service? Obviously, an encrypted database will render your application useless in most cases. Returning to normal operations will typically mean paying a ransom to the attackers in the form of anonymous cryptocurrency in order to gain access to a decryption key that will unlock the data. Mature organizations may be able to overcome this disruption by restoring from backups that have not been encrypted.

Ransomware has continued to rise in the past several years and has catapulted cybersecurity to the mainstream with such famous attacks like WannaCry, the City of Atlanta, the Port of San Diego, and the Colonial Pipeline attacks. With Ransomware as a Service (RaaS) on the rise and affiliate attackers reusing popular ransomware software the trend will continue to go against organizations.

Ransomware is a persistent and growing threat to organizations. The idea of having your data encrypted with no method of decryption is paralyzing to think about. Cybersecurity & Infrastructure Security Agency (CISA) has several recommendations to avoid the risks of ransomware:

- Ensure that your organization has a procedure for patching software.
- Backup data on a regular basis and test the backups.
- Restrict access to systems and software following the principle of least privilege.

2.3.4 Casino Betting Offline

Not all organizations face DDoS attacks equally. Some, due to the nature of their business, face an increased risk. Imperva is a leading provider of application layer DDoS protection through a suite of tools including a Web Application Firewall (WAF). Their Global DDoS Threat Landscape Report released in 2020 showed that gaming industry and the gambling industry continue to be the most attacked websites on the internet. It is easy to see why this is the case. These sites need to be universally available especially during big events or risk losing revenue.

In 2020 an attacker used Datagram Congestion Control Protocol (DCCP) to slip past DDoS protections that are geared toward other network protocols like TCP and UDP in order to perpetrate one of the largest DDoS campaigns ever seen. They used the new attack vector to perform what is known as DDoS extortion or ransom. RDDoS (Ransom DDoS) is, as it sounds, a way for the attacker to threaten an organization to either pay a fee or be the subject of a DDoS attack. By February of 2021 these attackers were able to muster over 800 Gbps of traffic to direct toward their victim. In this case, a gambling site in Europe.

2.3.5 Health organizations are still fair game

Despite a global pandemic, the healthcare industry was still too juicy of a target to keep attackers at bay. Many of the most well-known cyber-criminal gangs (more on these later) claimed that they would avoid attacking health organizations to show their sensitive side. However, there are plenty of fish in the sea as it were that did not feel the same.

There were countless attacks and attempted attacks against health organizations during the Covid-19 pandemic. Both the HHS (Health and Human Services) in the United States and the Paris AP-HP were in the crosshairs of DDoS attacks. In the case of the HHS, the threatened DDoS attack whose aim it was to create disruption in the pandemic response, was thwarted. However, the Paris AP-HP which operates dozens of hospitals across France as well as providing research and disease prevention was impacted by a DDoS attack that was absorbed by the network provider.

One thing that is clear with the targeting of healthcare organizations is that attackers will always go for the lowest hanging fruit, the least amount of effort, and the most likely to pay a fee. Healthcare organizations are not engineering organizations. Their primary purpose is to provide healthcare services to patients in need. Any disruption to their capabilities puts lives at risk, which will drive decisions to move quickly to a resolution. With technology that is usually years behind the state of the art, and small or outsourced technology teams, the quickest resolution is usually processing a payment to an attacker.

2.3.6 Building in resiliency

Availability has one best friend in the world. That's resiliency. One of the most iconic bridges in the world is San Francisco's Golden Gate Bridge. Known as one of the Wonders of the Modern World, the Golden Gate Bridge connects the San Francisco Bay with the Pacific Ocean. Construction began in 1933 and was completed four years later. The Golden Gate Bridge is a great real-world example of building for resiliency. Not only was the construction of the bridge an extreme modern marvel, but there are unique considerations for places like San Francisco. Namely earthquakes, as the Golden Gate Bridge is within close proximity of the San Andres Fault which produces frequent seismic activity, some of which can be devastating to the area. It has survived several large earthquakes over its eighty plus years of existence, but not without the need of constant review and rework. Several projects are ongoing to make it not only more secure against earthquakes but also high winds. What does this have to do with software? Risk management in architecture is not much different whether you are building a bridge across a peninsula or developing an application to deliver value to your customers. Architecture, design, and development all need to consider the what-if scenarios and plan for possible attacks, failures, or errors. Some of which could be intentional.

Your application should be designed and architected in a way that takes into consideration the type of risks and threat actors that may be looking for weaknesses in your specific application. Similar to the Golden Gate Bridge needing to be built to withstand winds as well as potentially large earthquakes, a healthcare application needs to be built to withstand a cyber-attack that is attempting to ransomware your client data in order to turn the downtime into a profit. Your gambling application must consider criminals looking to

DDoS your application to extort your organization. Your critical infrastructure application must be prepared to handle an advanced persistent threat looking to shut down your system to cripple key parts of a nation. We will talk more about these in the next sections.

One of the simple methods of building in resiliency is to add more processing power. Scaling vertically means you are creating bigger systems. Scaling horizontally means you are adding more systems. This is easier said than done. However, the migration to cloud first architecture makes this easier, albeit expensive. Additionally, building segmentation into your architecture reduces the "blast radius" of a potential attack. If you always assume that one of your systems is compromised, you will look at your architecture much differently. Similarly, there have been books written on the principles of chaos engineering which in its simplest form means injecting a bit of chaos into a system to identify issues in order to prevent full-on outages. Think of trying to find a pin hole in a tire. You might rub some soapy water around it and put it under pressure to look for where the bubbles show. This is similar to the concept of chaos engineering.

- Develop a hypothesis: "The application will respond gracefully if an external service is not available".
- Test the hypothesis: Route HTTP from the application to a non-existent service and observe the failure condition.

Fix: If required, identify the issues, resolve them, and improve processes.

One last comment regarding building resiliency in. Things will break, processes will fail, systems will crash. What is done with that information is critical to improving the system long term. Learning from mistakes and failures is the best method of building more resilient systems. A typical component of learning from failures is in a post-mortem or a root cause analysis. Here, the team reviews what happened and where the failures were. From there they build incorporate the findings into an action plan to ensure that controls, processes, and automation are in place to ensure the failures don't occur again.

2.4 Integrity

When we open a bottle of our favorite beverage, we expect to hear the sound of the cap snapping away from the protective top. Data is little different in the sense that we want to have confidence that it is correct when we view or process it. Integrity is the ability to know that data is known to be good and trusted. This requires the application to trust that data has not been tampered with whether in transit or at rest. Integrity needs to be confirmed in communication between two applications, in databases or file systems, or a piece of hardware or software. Anywhere data resides or moves requires it to continue to be trusted by the applications that use it.

As with availability and confidentiality, integrity issues can be intentional or accidental. It can be from an attacker injecting junk into a business process that corrupts data that is then stored or processed, or it can be a write failure that then corrupts those backups you were planning on using to recover from a potential ransomware attack.

2.4.1 Integrity starts with access

Have you ever walked to your desk, got in your car, or entered a room in your house and noticed that something was moved that you *know* you didn't move? Your mind begins to walk back over the last time you were in that space. Did you really put that pen there? Was that book always on that side of the table. If it wasn't you, then who was it? If only you had a method to see who was in that room other than you and catch the perpetrator red handed. I got news for you. It was probably you.

Think about a simple web request from your browser to a web server. The request leaves the browser in a GET or POST, traverses the network as a packet, which lands on several network devices, it is received by a load balancer or proxy or a WAF, it hits a web server, an application server, and a database. By the way, it's most likely been logged in a few of those locations and sent to separate, centralized logging system. Not all paths through a system are the same, but in general this holds true for most HTTP requests made today. At each one of those steps there is the potential for a user or an account to gain access to view, copy, or corrupt the data.

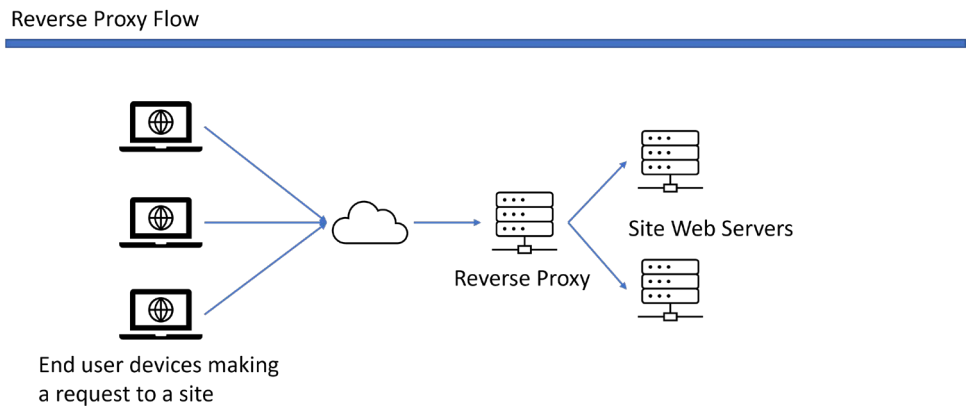


Figure 2.6 – Simple HTTP request with a reverse proxy

Access control and monitoring is primarily used to determine what accounts have had contact with data as it has gone through a system. It can be achieved through means as simple as providing a monitoring system for logging access to certain files, directories, and resources. Or it can be as complex as a privilege access management (PAM) system that requires a user to check out account access to sensitive systems where actions and keystrokes are logged. There are systems and security models in between that will provide varying level of access control. The key takeaway is to ensure that you can trace access, down to the individual file, to a physical person or user account. The moment that you have something like a shared account you lose the ability to trace activity to a single account or individual. These accounts are ones that have a shared username and password that is used by a group of individuals.

Non-repudiation is a form of access control that grants the application the ability to track an action or activity back to an identity. The most important concept with non-repudiation and access control can be summarized by the FDA's definition of an *audit trail*.

> **DEFINITION** An audit trail is a secure, computer-generated, time-stamped electronic record that allows for reconstruction of the course of events relating to the creation, modification, or deletion of an electronic record.

Attackers may look for ways to ascribe their activity to a different account or otherwise pin the blame somewhere else. When an attacker performs an attack on an application there is a goal in mind, perhaps data exfiltration or DoS. In these cases, they will also attempt to hide their activity through poisoning or corrupting the logs so that the team that reviews the incident is not able to piece together the attacker's activity. It is important to validate the input that is coming into your logs and make sure that access to logging workflows is tightly controlled. It is also important that enough information is logged to aid in the forensics of a potential incident. Without this audit trail, the ability to trace behavior back to an entity will leave the organization unable to determine root cause of activity.

2.4.2 The role of version control

Version control is a method for making certain that if data is corrupted, that there is a means to return to a good state. For those of us that have done any software development, we know that there have been times that, while coding, you broke something else in the process. Or maybe that only happened to me. You would usually attempt to roll back to a known good state or at least be able to view an older file and compare it to the current so that you can determine what broke. Version control software performs the function of providing control and visibility into files over time. As mentioned, this provides the ability to compare and rollback in the event of data becoming corrupted, deleted, or just bad coding.

Version control also offers the opportunity for a team of software personnel to work on a large application. Rarely is an application developed by one individual. It takes a team to build an application that is developed for commercial use. The use of a version control system (VCS) allows for the team to work simultaneously on the same application while providing the means to track code check-ins, resolve conflicts, and enable the ability to revert changes.

Many newer VCS include the ability to perform code reviews, track defects, and leverage task controls that can perform jobs related to continuous integration like merging code, running test suites, and creating a software package that can then be deployed to an environment where it can be tested. In relation to software security, the VCS that is used allows for the application security team to perform code reviews on sensitive code. For instance, a change may have been made in the code that impacts the authentication of the application. In this case, the engineering team may request that the application security team review the code for any increased risk due to the change. Additionally, if the VCS provides the ability to run tasks, the application security team can use this point to perform automated security testing to uncover vulnerabilities, like using a static analysis tool.

2.4.3 Data Validation

An additional consideration when maintaining data integrity is to maintain the data as it is initially brought into your system through a user or service. In the application security space, we typically call this data scrubbing or validation.

> **NOTE** If there is only one thing that you remember from this book, let it be the fact that you can never trust your end-users or the services your application work with to send you correct data. Whether intentional or not, data that is not properly validated can lead to failures in your application or a malicious actor being able to compromise it.

This means simple concepts like only accepting numeric values of certain length when expecting a phone number or social security number. It can be as complex as writing regular expressions that look for specific characters and other requirements. Many development frameworks may even have data validation built into the framework. However, ultimately, the validation strategy should ensure that the input data is both within the constraints of the value it is expecting and also makes sense from a logic perspective.

Depending on your development framework, validation techniques are natively available and can be leveraged to check common constraints on size, type, schema, and others. One consideration when developing validation either using the native elements of your framework or developing your own is that taking the allow list approach is more effective than the deny list. What this means is that the application should specify what it allows, and not what it blocks. It is apparent that a deny list will need constant maintenance and will not always catch novel approaches to circumventing the validation. An allow list will only permit the values that the application is expecting to work with and is a more secure method.

One last point on the input validation is that it is not sufficient to only examine the input on the client side. This check must also be done at the server side. Proxy tools, and projects like cURL allow an attacker to manipulate an input or send requests to your application without needing to go through the UI. In other words, most attackers will easily circumvent your client-side validations that might be in place.

2.4.4 Data Replication

Data replication is usually thought of as data backup. However, there is more to it than that. Data replication entails making multiple copies of data in different locations to be able to absorb potential data loss or corruption in one copy. This provides not only better availability and lower latency but also the ability to restore to a known good state should something go wrong. Although there is file and system backups that should be in place, the application architecture should include the ability to distribute data through a clustered database. While in this scheme data is split so that different fragments go to different nodes and are retrieved in a manner that will then consolidate the different fragments back to the original data. Replication of the data will provide the additional layer of making sure that a failure at one node does not limit the ability for the cluster to retrieve the data.

2.4.5 Data Checks

If you have worked in technology long enough you are well aware of the concept behind checksums. These are values that provide a fingerprint for a given input. This is essentially a hash of an input which means that it can only be done in one direction. In other words, the plain text is turned into a hash value, but the hash cannot be returned to the plain text. Checksums and hashes are not necessarily different from each other except for their purposes. Where hashes are an output of a mathematical function that is intended to create a unique value, a checksum is used to make a comparison and then decide whether a value has changed or not.

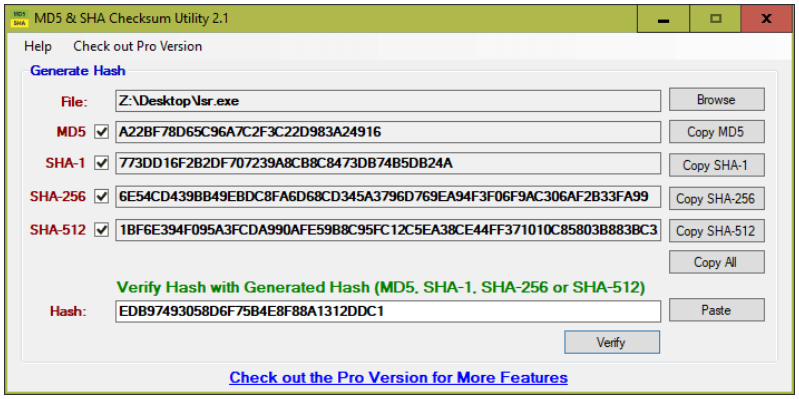


Figure 2.7 – Hash validation utility

Another example of a generated code for that provides data integrity assurance is the message authentication code (MAC). These are cryptographic checksums that are used to not just prove that data has not been tampered while it was saved or transmitted, but it also provides a means to authenticate who it came from when a cryptographic key is used during the MAC generation.

2.5 Authentication and authorization

Authentication and authorization are two additional tenants that are vital to maintaining the security of an application. They are components of what is typically thought of more broadly as identity and access management (IAM). Diving deep on this topic is out of scope for this book, but it is important to know what they mean and their roles in security.

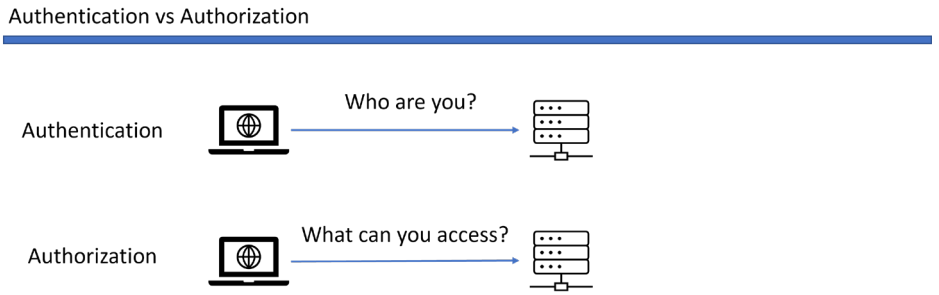


Figure 2.8 Authentication vs Authorization

2.5.1 Authentication

Authentication (or AuthN) is the practice of confirming that a user or account is who they say they are. This is accomplished by proving identity through the following:

- Username and password which accounts for something you know.
- A token through either a hardware device or software which is something you have.
- Biometrics like a fingerprint or retinal scan which is known as something you are.

In most cases, a single factor is enough to verify the account. Our username and password combination is widely used to allow us to login to applications. More strict applications or services may require additional factors like token or biometrics (or all three) to provide enough verification of the account. The use of more than one factor is considered multifactor authentication. For instance, password combined with a token would be two factors that are used to provide authentication. It is important to point out that once an account has been authenticated, it does not grant access to everything within a system. That is an important distinction between authentication and authorization. Your ability to login, and therefore authenticate, to a site like Amazon.com does not entitle you to access everything including administrative functions.

2.5.2 Authorization

Authorization is the process of granting access to a user or account to certain features or activity within an application. This occurs during or after the authentication process. It's important to consider that during the authorization process that the least privilege approach is taken to ensure that the account only has access to the features that it needs to perform its tasks. That can be achieved through several access models.

- MAC (Mandatory Access Control) – Gives the access control to the owner and custodian of the system or data.
- RBAC (Role based access control) – Provides access based on the account's participation in a group or role.
- DAC (Discretionary access control) – Gives complete control of the access control to the owner of a system or data.
- RBAC (Rule based access control) – Defines access for an account that is defined by the custodian or system administrator.

One common way to think of authorization and its role in identity is the common access card, passport, or license. These types of identifying items usually have a photo, your name, address, and other identifying information. Having one of these helps someone confirm your identity since you are the person in the picture. This is authentication. However, having that identifying item doesn't mean that you are granted access to certain locations. For instance, your license will not allow you access into sensitive areas of military base. Sure, the picture and information on the license identifies you but it by no means grants you access to off-limits areas.

2.6 Adversaries

One of the foundations to developing a defense in depth approach to addressing security is to know what types of attacks you should be expecting. The following is an often-quoted line that suits our needs here:

> **If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle.**
>
> **Sun Tzu The Art of War**

Of course, we are not headed to battle every day when we head off to work, but this still works in the context of cybersecurity. Knowing the attackers, their methods, and what their targets are in your organization allows you to know where to spend your effort and money. We'll talk more about general risk next, but first let's talk about who the various adversaries are that you will likely encounter.

2.6.1 Script Kiddies

One of the most prevalent adversaries that we can expect to see in the cybersecurity space is the "script kiddie". It's not a great name, but these are typically low skill attackers that have little motivation outside of revenge or fame. They will look to purchase, or reuse exploits that others have developed with no knowledge or understanding of how the exploit works. They simply want to point it to a target and click a button.

Although the skill level of these attackers is low, they are generally using automated type of attacks that allow them to run these attacks at scale. Make no mistake, organizations are under near constant, daily attacks. Some of this is noise from the general internet, other

activity is due to these adversaries that are just looking to test their skills or gain bragging rights with their friends. This type of motivation may make their attacks less impactful, but they are still widespread.

Botnets

Botnets are a common attack tool used by script kiddies. This is a method used to perform DDoS attacks at scale. These networks of bots are compromised devices that have malicious code injected into them so that they may be used in a future attack. Most of the time they are used to target organizations for volumetric types of attacks like DDoS. They can be Internet of Things (IoT) devices, laptops, desktops, printers, or any other device that has an internet connection and can be used to make web calls. Some of these botnets are even for sale or can be rented for a period of time. For this type of attack most organizations are not equipped to mitigate it themselves and need to look for outside help in the form of firewalls both at the application and the network layer. Other controls require your organization to know the type of traffic that it is expecting to see and where it comes from so that the controls can be put in place to limit activity to just those known locations. This might require geofencing that limits traffic to known good countries or locations.

2.6.2 Insider

The insider threat is an often-overlooked category. These are users that usually have privileges to data and systems that outsiders would not. Think about the system administrator for your organizations domain controller. Now that you have that person in your mind, think about that person being disgruntled. What about the back-office admin in a medical facility that has access to patient records? What if they are no longer "feeling" their job. These things do happen. Although it is fair to say that most insider leaks are accidental and not due to the activity of a malicious actor.

The numbers vary on insider threats since not everything is required to be reported. However, Ponemon Institute produced the "2020 Cost of Insider Threats: Global Report" that showed that criminal insiders accounted for around 23% of insider incidents and 14% were due to criminals posing as insiders. These types of attacks are usually thought of as a problem for the Information Security team to deal with, but keep in mind that software engineering teams not only have access to the organization's intellectual property in the form of proprietary code, but also usually have elevated rights to environments that may have sensitive information.

The motivations for an insider can vary. As mentioned, some leaks at the hands of an insider are accidental. This is due to the lack of controls around access to information like a lack of proper segmentation or a least privilege model that enforces access to only those that need it. Controls should also be in place to guarantee that sensitive data and production data is not able to be moved from a production environment to another location. When motivation is to take revenge on the organization for a perceived slight, this becomes more complicated to defend. However, getting to a zero-trust type of model means that you should, unfortunately, suspect that anyone that has access to your most sensitive data is malicious and provide the appropriate auditing and protections around access to that data.

2.6.3 Cybercriminal

Depending on the industry that you are working in, cybercriminals may be the most worrisome. Most organizations that are in industries like the financial or healthcare industries perhaps face the most difficult challenges when it comes to cybercriminals. There is one thing that motivates cybercriminals: Money.

As mentioned, many times, data is the new oil. It's a commodity and organizations collect, trade, and monetize it. There is some folk lore that attributes a comment to Willie Sutton, a famous bank robber in the 1900s. As the lore goes, Willie Sutton was once asked why he continues to rob banks. His famous reply supposedly states, "Because that's where the money is." Turns out that may not have been totally accurate according to Willie Sutton himself and was perhaps just an editor attempting to be pithy. Regardless, the statement is true and sums up, pretty well, why cybercriminals continue to target organizations and their data.

Their techniques vary. Ransomware, DDoS, data exfiltration, or even physical theft of devices. The goal is to get to the data, put it on the black market, and make some quick money. Their technical abilities vary, but they are more adept at infiltrating an organization from the outside than the previously mentioned actors. They will use off-the-shelf or purchased tools, or they can go as far as developing and customizing their own exploits to target specific entities. They are more targeted in their approach and will look for low hanging fruit in order to compromise an organization.

2.6.4 Hacktivist & Terrorist

I don't mean to lump hacktivist and terrorist in the same category since they have different goals, but they are both propelled by the desire to publicly make an example of someone or some organization in order to advance an agenda. For both actors this tends to be of a political nature. But that's where comparisons usually diverge.

Hacktivist will take up causes for people or movements aimed to effect change towards a shared goal. They are usually a loose band of individuals with a high level of skills. They will look to deface websites of organizations they don't agree with or even DDoS them. Their goal is to raise awareness around a cause and their methods will vary.

Cyber terrorists move beyond political motivations and will often take up religious or ideological causes and target organizations accordingly. The targets can be individuals, or organizations and will often look to target key infrastructure of nations with the intent of causing physical harm.

2.6.5 Advanced Persistent Threat

Advanced persistent threats, or APTs, are gaining more recognition recently as nations such as Russia, China, North Korea, Israel, and the United States find their cyber differences (putting it mildly) out in the public space. APTs are deep pocketed, nation backed entities whose sole intention is to gain tactical and strategic advantage over adversaries. This shows up in their effort to gain access to organizations in specific industries, critical infrastructure, military entities, and so forth. In most cases, the APT will lie in wait and stay hidden until an order comes.

As mentioned, these are deep pocketed groups and are the highest skilled adversary that any organization will face. APTs are usually branches of an already established organization within the nation such as intelligence services or the military. Their focus is to gain a foothold into another nation with the intention of creating domestic chaos, take out military capability, or cripple a nation's economy.

Defending against these types of attackers is not easy and the best advice is often to do the basics of monitoring and protection. However, Rob Joyce who led the NSA's elite Tailored Access Group whose purpose was to break into adversaries' systems, summed up every organization nightmare during a conference in 2016:

> **We put the time in …to know [that network] better than the people who designed it and the people who are securing it," he said. "You know the technologies you intended to use in that network. We know the technologies that are actually in use in that network. Subtle difference. You'd be surprised about the things that are running on a network vs. the things that you think are supposed to be there.**
>
> **Rob Joyce**

It's important to highlight the last statement here. Systems are complex, and in any large organization the amount of applications and services running can be immense. Sophisticated attackers are counting on large organizations not having the bandwidth, personnel, or tools to detect malicious activity quickly. Their often right.

2.6.6 Why do we care?

Defending against these threat actors requires varying techniques. Organizations often find themselves as victims to broader geo-political attacks that have nothing to do with them. Especially when it comes to hacktivist, cyberterrorist, and APTs.

Figure 2.9 Relation of threat actors vs defense

Knowing your adversary is key to survival. Knowing that script kiddies will largely use already identified, automated attacks means that you're off the shelf tools used for scanning

and protection will usually suffice. Defending against more sophisticated attackers like the hacktivist, cyberterrorist, cybercriminal, and APT mean that your defenses need to be more robust and a plan for business continuity is required in order to recover from a potential attack. For these types of attackers, you need to move from the preventative mindset and pivot to a detect and respond mindset. The more sophisticated the attacker is, the more resources are needed to defend. Regardless of the attacker, the basics of security need to be integrated. Scanning, patching, vulnerability management, visibility, and defense in depth are all required regardless of the threat actor.

2.7 Measuring risk

This book is not about risk, though it is an integral part of secure software development. There are plenty of resources available to help organizations understand and balance their risk. The goal of this section it to highlight some of the concepts with risk as it relates to an organization. Understanding this relationship will help application owners know why certain controls are used and needed. This leads to a stronger defense in depth model. Without knowing the risks that are posed you run the risk of overcorrecting for risks that are not of a legitimate concern.

As mentioned, there are many different methods for measuring risk. For the purpose of this book, we will use the Open Web Application Security Project (OWASP) methodology aptly named the OWASP Risk Rating Methodology that allows you to identify risks through a series of steps. It uses a number range from 0 to 9 to assign a value to a particular rating. The lower the number, the lower the level and vice versa. Although this is not a perfect system, the goal is to create a means for scoring security issues.

OWASP

Throughout this book we will leverage many of the projects from OWASP. When it comes to application security, OWASP is generally the first stop for most application security professionals. It is an open-source community of thought leaders that have built many of the foundational security practices used today. One of the most widely recognized application security projects is the OWASP Top Ten Web Application Security Risks. This documents the top 10 most impactful security risks that a web application faces. (https://owasp.org/)

Using the OWASP Risk Rating Methodology the measured risk comes down to a simple equation:

```
Risk = Likelihood * Impact
```

The likelihood is calculated by asking questions related to the threat agent and the found vulnerability. More on this in a bit. The aim of these questions is to uncover how likely an exploitation of the found vulnerability may be. With the impact calculation there are another eight questions that are geared toward identifying what will happen to the organization and technology should the exploit be successful. This will lead you to the ultimate rating of the risk which allows you to prioritize and apply the appropriate controls to eliminate or mitigate the risk properly.

Before diving into the risk rating methods with OWASP, it's important to callout that once a risk has been rated there are several things that an organization can then do with that information and that risk.

2.7.1 Remediate, mitigate, accept

In general, there are three methods to managing an identified risk. The first is to remediate the risk. This requires the organization to take corrective action to fully implement risk elimination. An example would be a case where the application faces the risk of *formjacking*.

> **DEFINITION** Formjacking is an attack where an attacker is able to inject code that skims data from an HTML form. One of the most well-known types of formjacking is an attack called Magecart which specifically targets checkout pages to steal users' information by injecting malicious code in third party supplied code to a website.

This is where an attacker is capable of taking over a form on a website that allows them to inject code that steals information. To remediate this risk, the organization will need to perform regular testing through tools and penetration tests to identify opportunity of code injection into the forms they use. They will monitor traffic and create an allow list that only allows outbound traffic to known good locations. Additionally, they may look to leverage subresource integrity tags (SRI) to create a hash of content that is used by the application so that it can determine whether the content has been tampered with. They may even go as far as eliminating forms if they are no longer needed.

If remediation is not feasible for business or technology reasons. The next step is to identify ways to mitigate the risk by placing in compensating controls. This can also be considered reducing of the risk since it may not completely eliminate it, but instead makes it less likely and raises the bar for an attacker. In most cases, mitigation involves additional tools like a WAF or by reducing the size of the attack surface by limiting the audience down to as few accounts as possible and monitoring those accounts closely. Again, the goal of remediation is not to eliminate, if this isn't possible, but to reduce the attack surface to as small as possible.

The last option is to accept the open risk. This is less than an ideal option as it means that the organization is aware of the risk but has chosen to leave it open and accept it. This is should still be coupled with risk reduction so that the risk is as minimal as possible. A prime example of where risk is accepted is in the case of an older application that simply cannot be shutdown. It may be due to the fact that the organization has been unsuccessful in getting clients to move to a new product or a new version of the given application, but whatever the case is the organization requires the risk to remain open due to a business decision. In many cases, the acceptance of risk is taken when the impact to the business is low. In other words, the business is willing to accept a breach and knows that the total cost to the business would be low.

There are other facets of risk that can come into play such as cyber insurance and risk transfer. However, the purpose of this section is to cover the primary ways that organizations treat risk as it applies to the products and applications they create.

2.7.2 Identify the risk

The first step in rating the criticality of a risk using the OWASP Risk Rating Methodology is to identify what the actual risk is to the application or organization. This could come through due to the nature of the organization, the type of application in use, or geo-political factors. Risk is ever evolving and may even be eliminated by doing nothing as both risk and technology changes.

This is where knowledge of your threat actors (adversaries), a well-documented architecture, and strong knowledge of how your application is deployed and used will come in handy. Identifying risk can come from conversations with the business and technical people within the organization, it can be made apparent by a client who reports a risk they identified, it can come from fellow industry partners, or it can come from your internal tools. Regardless of method of risk identification the most important part is to know you have risk.

In the OWASP Risk Rating Methodology there are scores associated with each component. The higher the score, the higher the risk. When performing a risk rating it is important to have the right resources involved with the measurement. This includes not just technical resources, but also business resources that understand what the impact to the business and organization would be for a given risk. Measuring of the risk can take anywhere from a few minutes to a few hours depending on the resources involved and the complexity of the risk. It is also important to highlight that much of the risk rating is subjective where, depending on the resources involved, it is easy to go down a rabbit hole on conversations over the risk. An example of this is where participants in the process may differ on their perception of the risk or aspects of the mitigations that are available. Key to success here is to lay out ground rules, have previous examples handy, and keep the participants on topic.

2.7.3 Estimating likelihood

Likelihood is as simple as identifying when a risk may be exploited. We are all familiar with the statistics around driving a vehicle and the likelihood of a potential accident. Although many of us are aware of this risk, it doesn't stop us from getting into our vehicles on a regular basis. It's a risk we are willing to or need to take. The bottom line is that we never know when or even if an accident will occur on a commute to the office or another location, we simply know that statistically it may occur. Additionally, certain factors come in to play with this analogy. How fast we drive, other vehicles on the road, the safety of the route to our destination, and so on. The same applies with understanding the likelihood of a risk. OWASP gives us eight factors for the threat actor and the vulnerability to help us measure the likelihood. The scores associated with each item is a weight that represents the impact of that item. This is used in the final calculation of the actual risk.

Threat Actor Factors:

- Skill Level - How technically skilled is this group of threat agents? No technical skills (1), some technical skills (3), advanced computer user (5), network and programming skills (6), security penetration skills (9),
- Motive - How motivated is this group of threat agents to find and exploit this vulnerability? Low or no reward (1), possible reward (4), high reward (9)
- Opportunity - What resources and opportunities are required for this group of threat agents to find and exploit this vulnerability? Full access or expensive resources required (0), special access or resources required (4), some access or resources required (7), no access or resources required (9)
- Size - How large is this group of threat agents? Developers (2), system administrators (2), intranet users (4), partners (5), authenticated users (6), anonymous Internet users (9)

Vulnerability Factors:

- Ease of Discovery - How easy is it for this group of threat agents to discover this vulnerability? Practically impossible (1), difficult (3), easy (7), automated tools available (9)
- Ease of Exploit - How easy is it for this group of threat agents to actually exploit this vulnerability? Theoretical (1), difficult (3), easy (5), automated tools available (9)
- Awareness - How well known is this vulnerability to this group of threat agents? Unknown (1), hidden (4), obvious (6), public knowledge (9)
- Intrusion Detection - How likely is an exploit to be detected? Active detection in application (1), logged and reviewed (3), logged without review (8), not logged (9)

As you can see the likelihood factors are looking at the threat actors as it relates to the vulnerability. The organization can then take this information to determine how likely a vulnerability is to be exploited based on the skill level and knowledge of the vulnerability.

2.7.4 Estimating impact

Impact is a bit different when it comes to measuring as it is not based solely on what can be determined by just technical folks. Impact must leverage information related to the business considering that the impact is a measure of what is likely to happen to the organization should a vulnerability be exploited. One important consideration with impact is that there are two types of impact. Technical impact is a risk to our core security concerns of confidentiality, integrity, and availability. This focuses primarily on the systems that run and manage our application. Second is the business impact which prioritizes what is important for the business that is running the application and is usually financial in nature.

Technical Impact Factors:

- Loss of Confidentiality - How much data could be disclosed and how sensitive is it? Minimal non-sensitive data disclosed (2), minimal critical data disclosed (6), extensive non-sensitive data disclosed (6), extensive critical data disclosed (7), all data disclosed (9)
- Loss of Integrity = How much data could be corrupted and how damaged is it? Minimal slightly corrupt data (1), minimal seriously corrupt data (3), extensive slightly corrupt data (5), extensive seriously corrupt data (7), all data totally corrupt (9)
- Loss of Availability - How much service could be lost and how vital is it? Minimal secondary services interrupted (1), minimal primary services interrupted (5), extensive secondary services interrupted (5), extensive primary services interrupted (7), all services completely lost (9)
- Loss of Accountability - Are the threat agents' actions traceable to an individual? Fully traceable (1), possibly traceable (7), completely anonymous (9)

Business Impact Factors:

- Financial damage - How much financial damage will result from an exploit? Less than the cost to fix the vulnerability (1), minor effect on annual profit (3), significant effect on annual profit (7), bankruptcy (9)
- Reputation damage - Would an exploit result in reputation damage that would harm the business? Minimal damage (1), Loss of major accounts (4), loss of goodwill (5), brand damage (9)
- Non-compliance - How much exposure does non-compliance introduce? Minor violation (2), clear violation (5), high profile violation (7)
- Privacy violation - How much personally identifiable information could be disclosed? One individual (3), hundreds of people (5), thousands of people (7), millions of people (9)

2.7.5 Risk severity

Now that we identified the likelihood and impact factors, we are able to put it together to understand the overall severity. As mentioned previously, the higher the rating for each of the factors the higher the overall risk.

0 to <3	Low
3 to <6	Medium
6 to 9	High

2.7.6 Risk example

Reusing the example earlier of the formjacking we can make some assumptions and walk through a scenario. I will use the example organization from the previous chapter, Superior Products. Dashing Danielle has been made aware of the formjacking issue impacting one of their flagship products that was raised up by an internal penetration test that was recently

completed. Although, after some research, she knows that this issue is significant, however completing a risk rating will help her prioritize the issue with the engineering team. The impacted application has a section for making purchases and submitting reviews. There is a form in the section where the user can provide their credit card details in order to make purchase.

Given this basic information we can make some assumptions about the threat actors and vulnerability in order to come up with the overall likelihood. For this example, the ability to perform a successful attack requires moderate skill for a big reward in the form of stealing credit card information. The threat actor category would be cybercriminals and would be generally widespread. For the vulnerability itself, the ability to act on it is determined by the fact that automated tools are available to not only detect but to also create the script that can be used for the attack. As of now, Superior Products has minimal ability to detect an attack. With this in mind we can determine the likelihood in the following table.

Table 1.2 Sample of a threat likelihood using a formjacking attack

Threat agent factors				Vulnerability factors			
Skill Level	Motive	Opportunity	Size	Ease of Discovery	Ease of Exploit	Awareness	Intrusion Detection
4	8	6	6	7	7	5	3
Overall likelihood = 5.75							

For the impact we will break this into two. One for the technical impact and one for the business impact. Looking at this particular risk, the loss of confidentiality is high due to the fact that the threat actor can gain credit card information. There is no impact to integrity, or availability. Accountability would be difficult since the attacker would be able to perform this attack without being known to the application. For the business impact, this would be significant given that if this is not well detected it could go on for a long period of time before being noticed. Although financial impact would be low, the reputational and compliance impacts would be high as well as the possibility of this being a privacy violation depending on what information would be stolen.

Table 1.3 Sample of a threat impact using a formjacking attack

Technical Impact				Business Impact			
Loss of Confidentiality	Loss of Integrity	Loss of Availability	Loss of Accountability	Financial damage	Reputation Damage	Non-Compliance	Privacy Violation
8	1	1	8	7	7	7	3
Overall technical impact = 4.5				Overall business impact = 6			
Overall impact = 5.25							

What does this example show us? The likelihood of occurrence and the impact are both medium. However, the business impact is low. This allows the Superior Products to take the approach that although there is a higher technical impact, the cost of the risk might be low enough that they would approach the resolution differently. Dashing Danielle works with the product owner and security organization to prioritize this vulnerability as low, opens a ticket with the appropriate development team with all the information needed for them to resolve.

Having this type of methodology will allow the organization to look at the risk, define a remediation or mitigation strategy that could eliminate or at least reduce the risk and properly prioritize the resolution.

> **EXERCISE** Use the online version of OWASP Risk Rating (https://www.owasp-risk-rating.com/). Use a scenario from a cyberattack news story. Take a particular threat from the story and walk through the risk rating calculator. Document your scenario, the final score and what you learned. You will have to use your imagination to fill in the unknown data. This is a chance to be creative.

2.7.7 Other methodologies

While the focus in this section is on OWASP Risk Rating Methodology, there are several other well-known methodologies that should be considered. The goal here is not to say that one is better than another but to simply outline that there are multiple options when it comes to risk rating methodologies.

Two of the other methodologies worth mentioning is the National Institute of Standards and Technology (NIST) Guide for Conducting Risk Assessments in Special Publication 800-30 and the Mozilla Rapid Risk Assessment. For those that are familiar with NIST you will know that this is a well-documented and thorough approach to risk assessment. The NIST approach is broader and encompasses more than a simple activity and instead focusses on the overall ability of an organization to frame the risk then monitor, assess, and respond.

DEFINITION NIST is another organization that has contributed greatly to the advancement of security. It is an organization based in the United States and is tasked with providing innovation and technical advancement. Through this effort NIST has defined many of the practices that are used not just in application security but in organizations who want to raise their overall security practices. (https://www.nist.gov/)

The Mozilla RRA (Rapid Risk Assessment) takes a similar approach to measuring risk as OWASP does in the sense that it aims to be discrete and quick. It looks at the risk from the point of view of whether the given platform has the appropriate level of security controls to host specific data. The input into an RRA is a data flow diagram (DFD) that includes the type of data that is used by the service being assessed. Additionally, an understanding or documentation, of how the service works. From here, the process is similar to a threat model (we'll talk about these later) where basic discussions occur on the service and its purpose. Data is highlighted with attention on how it's stored and used. Then a methodical approach is taken to review possible threat scenarios that focus on the confidentiality, integrity, and availability of the data. Once this is complete, recommendations are made on how best to provide protection.

Identifying risks allows the organization to prioritize and frame vulnerabilities that are presented. This means the organization can focus on the risks that have the biggest impact on what matters most to them.

2.8 Summary

- The CIA triad (confidentiality, integrity, and availability) is the foundation for every decision that is made in protecting data and ensuring that our systems are available when needed.
- Knowing your potential threat actors will assist in the definition of the appropriate level of protection that is needed. You don't need military grade protection if your only adversary is a script kiddy.
- Attacks get stronger as the threat actors get better. Defenses need to align with the treats that they are protecting against.
- Organizations such as NIST and OWASP are great resources for standards, and projects to help with ensuring your applications build security in.
- The OWASP Risk Rating Methodology provides a means to define the risk posed to an organization through a calculation that takes into consideration the technical and business impacts as well as the threat actors.
- Risks can be remediated, mitigated, or accepted. Each have their own benefits and disadvantages.

3

Components of application security

This chapter covers:

- **Building a threat model**
- **Using developer tools to mitigate software security issues**
- **Security Analysis tools used in the development pipeline**
- **Protection tools that are available for running applications**
- **Vulnerability collection, correlation, and prioritization**
- **Bug Bounty and Vulnerability Disclosure programs**
- **Where security fits in the SDLC**

So, you have seen the issues that are caused by not having application security integrated into your lifecycle and you're starting to ask the great question of where to start. There is a not a one size fits all package that works for all organizations. A lot depends on the following:

- Size of the organization
- The industry and the regulations impacting the organization
- The culture of the organization
- The security budget at the organization

It's often easy to overlook the organizations' culture having an impact on the effectiveness of the application security being applied, but this is a huge component. You cannot effect change if the engineering organization and the broader organization does not want to be more secure or is passive about security. Building security into the development lifecycle depends on the organizations ability to rally its engineers to the cause. But people in the organization can only do so much on their own. Providing the right tools and processes is critical to successful application security. Through the remainder of this chapter, I discuss several of the more common tools and processes that make up of a successful application

security program. This is by no means an exhaustive list and new tools and novel ways of solving application security issues are coming each year. However, the basics tools and processes are outlined here to give you an understanding of where it fits in the overall application security picture.

3.1 Threat modeling

There are many books that have been written on *threat modeling*. The intention of this section is to familiarize you with the different techniques and tools that are used to perform a threat model.

> **DEFINITION** Threat Modeling is a structured approach to identify, quantify, and address the security threats and risks associated with an application. It is an investigative technique for identifying application security risks/hazards that are technical (and even implementation specific). Threat modeling is an early-stage activity that is used to define security requirements for a design. Ideally, threat modeling would occur during the initial stages of the architecture development.

> **NOTE** Some additional reading that is helpful with threat modeling is Threat Modeling: Designing for Security by Adam Shostack https://www.amazon.com/Threat-Modeling-Designing-Adam-Shostack/dp/1118809998.

Threat modeling is one of the most fundamental parts of security. It is not just a specific part of application security but is used in all parts engineering and security including in networking, and operational teams. It can be as simple as asking a question like, "What could happen if a malicious user does this?" and can be as elaborate as gathering the appropriate subject matter experts to review a complex architecture with clear action items and takeaways with a list of associated risks and vulnerabilities.

The purpose of threat modeling is to identify the potential threats that might impact a system or architecture and define the counter measure that can be used to address the found risks. As I mentioned, it can be as simple as just asking what could happen, but more complex architecture needs more attention. Most modern architecture includes multiple external connections that are coming in and going out of the application. It may also have reusable components from other internal applications within the organizations. This represents a large set of moving parts that are often changing and present a unique challenge when completing a threat model as the attack surface is much larger than a simpler architecture. For these more complex architectures there are tools that can be used such as *blockatecture* tools like Microsoft Visio or another graphical tool that allow you to drag and drop blocks down on a canvas and draw lines. There are also specialized commercial tools that can be used to not only draw the architecture, but to also help identify the risks that can impact the drawn architecture. One of the most comprehensive methods of threat modeling is more manual and requires time and resources to spend the effort to whiteboard the architecture and manually identify the risks. Each of these methods have their strengths and weaknesses.

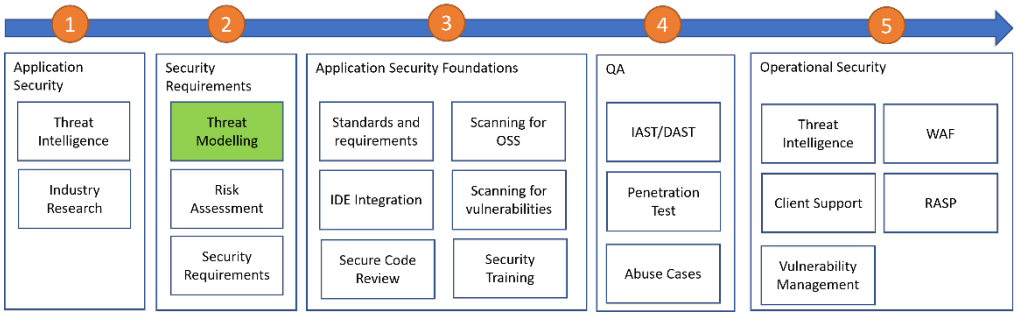


Figure 3.1 Threat modeling in the Secure SDLC

3.1.1 Basic threat modeling terminology

Before we get into actually performing a threat model you need to know a few terms that are used during the process.

- Attacker - Those who would misuse intentionally or unintentionally an element of the system under consideration. This could be one of the threat actors I spoke of earlier like a script kiddie, hactivist, or others.
- Asset – Anything you deem to have value and something that the system must protect from an attacker. Some physical examples are money and precious metals. Digital examples are data, especially sensitive data like protected health information, or personally identifiable information.
- Threat - A means by which an attacker might compromise an asset that has potential for success. Threats can include everything from hackers and malware, from earthquakes to wars. Additionally, intention is not a factor when considering threats. A mechanical failure of a hard drive in a data center is an equal threat to a coordinated attack by an attacker.
- Risk – The potential for loss, damage, or destruction of an asset as a result of a threat exploiting a vulnerability. In the example previously about the mechanical failure of a hard drive and an attack by an attacker, the mechanical failure is lower overall risk since there is usually redundancy built into an architecture to manage a failure meaning the impact is smaller.

When you want to understand risk and treats, it's important to ensure that you take emotion and gut feeling out of the equation. Take the two visuals in figure 3.2. Many people have a very visceral reaction to the bear and view the stairs as just another daily activity, but what are the actual risks?

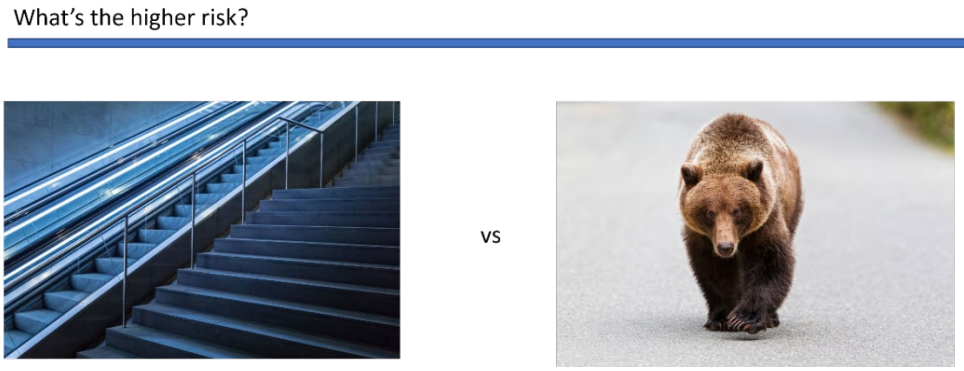


Figure 3.2 – A set of perfectly normal stairs and a grizzly bear.

The reality is that stairs kill far more people than bears do. On average, 12,000 people a year die from falling down the stairs. A few dozen people are killed per year by bears. However, most people will feel instant fear when faced with bear due to many different factors. What does this have to do with risk and security? When we look at the risks that impact our organization and software, it's important to put our risks in perspective. Most people that fall down the stairs may get right back up with some bumps and bruises, while the off chance of being attacked by a bear is likely to be much more fatal. Being under attack by an advanced persistent threat (APT) like a nation state is of course concerning for any organization. However, most organizations should be more concerned about the daily noise that comes from automated attacks and less sophisticated attackers. Although this is not as flashy as the attacks that come from an APT, the organization is more likely to see automated attacks.

Now that we understand some basic terminology and have some perspective on risk, let's turn to threat modeling. For many, it's easier to understand the concept of threat modeling by first taking a look at the manual method.

3.1.2 Manual threat modeling

Imagine that an organization has determined that they need to identify potential threats and risks to a new feature that they want to deliver to their clients. Although threat models can be done later in the development life cycle, as with most security tools and techniques, the most benefit will come from performing this activity as early as possible. There are several inputs that are required for a successful threat model:

- A completed architecture diagram and description of the feature.
- A data flow diagram that shows how data will flow through the system.
- A software bill of materials (SBOM) that provides a list of the software components used in the development of the application.
- Web service integration points such as APIs and other web services with third-party or internal systems.

Once these items have been gathered, members of the engineering team, security team, and business representatives would set aside time, potentially several hours or even days depending on the complexity. To make these sessions as effective as possible, the right people need to be involved. The engineering representatives need to be familiar with the overall architecture and the way the application is used in normal activity. More importantly, they should be familiar with the data flows. The security representatives should have knowledge of which vulnerabilities and risks that the organization is most concerned about as well as have familiarity with the application and the issues and risks that impact the technology stack that is used. This should include the web server, database, deployment methodology, and coding language. The business representative should be there to help identify the way the application is being used in the real world as well as weighing in on the identified risks and their impact to the business.

With materials in hand, the appropriate people will locate a room with a whiteboard. Although this can work in remote settings using virtual conference technology, it is far more effective to be physically together. For this exercise they will be using STRIDE to identify the risks that can impact the application. STRIDE is a threat modeling methodology developed by Microsoft (https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach) and is used in their free Threat Model Tool. STRIDE is an acronym that stands for Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege.

Other Threat Modeling Techniques

I will use STRIDE throughout this book since it is one of the more familiar ones used today. However, there are several other methods that can be used like OCTAVE, PASTA, Trike, and VAST. Which one to use really depends on the organization and the goals.

Threat modeling methods like Operational Critical Threat Asset and Vulnerability Evaluation (OCTAVE) focus on the non-technical risk that resulted from breached data and was developed by Carnegie Mellon University. In this method, assets are identified and classified which helps define the scope. Through three stages this method develops requirements, identifies vulnerabilities and gaps in policy or practices, and then develops an overall strategy to address the security risk.

In the Process for Attack Simulation and Threat Analysis (PASTA) method the organization takes and attacker view and then develops a threat management, enumeration, and scoring process. This can then be elevated to key decision makers to determine what risk to tackle as opposed to developing requirements at the SDLC level. This method is primarily asset focused especially with the mitigation strategies.

Again, there are several options when it comes to using a threat model methodology however, the STRIDE method is broad and is a good way to learn about threat modeling. Additionally, Microsoft Threat Modeling Tool uses STRIDE when it classifies threats.

3.1.3 Starting the manual process

The organization has the information gathered, the appropriate stakeholders, and has found a room with a whiteboard so that they may begin the process. Although the security representative does not have to lead the exercise, they are usually the most appropriate

person to keep focused on the task and make sure that the team is working toward the appropriate risks.

There are not a lot of ground rules for this effort with a few exceptions:

- Don't assume that your environment is secure or reliable. Hardware and software fail. Attackers will be pushing on your defenses. It is better to assume that your environment is neither secure nor stable.
- Don't assume that your environment is properly configured. Similar to the secure and reliable, configuration drift is real, and not all systems are configured the same way.
- Don't assume that your defense in depth will catch everything. Not all security tools are evenly applied or configured.
- Keep the conversation on to realistic scenarios and not "Hollywood" ones. It's fun to think that secret agents from an underground organization will physically breach your defenses with a stolen truck and steal your servers. This is extremely unlikely. Stick to the more practical and likely scenarios like a hardware failure, an injection attack, or elevation of privilege.

With the ground rules understood the security representative starts to draw on the whiteboard. They begin by drawing a few items:

- A simple copy of the architecture as blocks representing the different assets and technology in the architecture. This should include any third-party services. The purpose of this is to have a visual map that everyone can see.
- The acronym STRIDE with each spelled out as spoofing, tampering, repudiation, information disclosure, denial of service, elevation of privilege.
- Some exercises will put a "Hollywood" box on the whiteboard as well for any scenarios deemed too extreme.
- Lastly a grid with the following column headers and space to add items below:

What	Who	Why	How	Impact	Counter Measure

The grid serves as the working area for the remainder of the exercise. The group will begin by identifying a risk and completing the rows below the header. The headers are defined as:

- What: What is at risk in a given scenario. This should be specific such as "credit card numbers in the database."
- Who: What threat actors can potentially impact the identified object in the "What?" This should be specific such as "a developer with access to the production database."
- Why: What is the motivation of the "Who" to put the "What" at risk. Such as: "The developer wants to sell the credit card numbers on the dark web."
- How: This is a bit more difficult and should avoid unrealistic scenarios. A plausible scenario would be "The developer copies the production data to their developer device."

3.1.4 Threat modeling with linking bank accounts

We can use our examples from previous chapters of Superior Products who is launching a new feature in their e-commerce site that allows users to link their bank account in the application so that they can get paid for reselling items within the application. Dashing Danielle, the security representative in Superior Products for this application, begins by gathering the items needed as inputs from that development team. She reviews the architecture, and dataflow diagram as well as the software bill of materials to understand how the application is built and used. She follows up with the development team on a few outstanding questions and then organizes a work session with the development teams lead architect, two developers that have been working on the feature, and the product owner.

Dashing Danielle begins the session by drawing the simple architecture for the application along with, STRIDE, "Hollywood," and the grid on a whiteboard. The group agrees that the architecture is complete and includes the critical components of the new feature. She then asks a basic question of "what are we trying to protect?" The obvious first answer is that the organization must protect the bank account information that is sent and stored. Dashing Danielle adds bank account information in the "What" column.

What	Who	Why	How	Impact	Counter Measure
Bank Account Numbers					

Next Dashing Danielle ask, "who would want this information"? Similar to the question of "What", there is little debate on "Who". Clearly the attackers would be motivated by the financial gain that they would achieve by stealing bank account numbers. With this information, attackers would be able to access the bank accounts of the users of the application, gain additional information on the user, and potentially link the bank account to an attacker owned account. With this in mind, Dashing Danielle adds the information in to the "Who" and "Why".

What	Who	Why	How	Impact	Counter Measure
Bank Account Numbers	Cyber-criminals	Financial gains with the information.			

With the easy part done, it is now up for the group to determine how an attacker could steal bank account information and what impact it would have on the organization. Additionally, they will consider what counter measures they could put in place to protect against this specific attack, and whether those already exist. If they do already exist, then Dashing Danielle will cross it off the list under "Counter Measures".

This is where the team has to get creative. There are several ways an attacker can gain access to this information so it's good to start with higher level themes. Here are a few cases they think about:

- Someone could accidentally, or intentionally, move the data to a developer environment for testing with live data.
- The data could be stolen directly in the database by an attacker who manages to deploy malicious code within the network to gain access to the database.
- The application could be susceptible to attacks that leak data like a SQL Injection, Cross-Site Scripting, or Cross-Frame Scripting.
- The bank account information could get logged to a logging system in plain text.

For the purposes of this exercise, the team decides to first focus on the ability of the attacker to take advantage of a weakness in the application by leveraging Cross-Frame Scripting (XFS). The other attack opportunities can be reviewed in sequence after the first one.

What	Who	Why	How	Impact	Counter Measure
Bank Account Numbers	Cyber-criminals	Financial gains with the information.	Cross-Frame Scripting (XFS)		

3.1.5 What to do with the found threats

Dashing Danielle turns to the product owner to understand what the impact would be of a breach of this information with regards to any contractual stipulations that might require the company to pay a fee to clients. The product owner acknowledges that there are clear financial impacts directly linked with data loss, and also raises the concern of brand damage that might be difficult to overcome since there are other vendors and solutions in the market that are direct competitors with Superior Products application. Dashing Danielle also raises the likely support and recover cost associated with the attack as well as the potential for having to purchase credit monitoring for the impacted accounts.

What	Who	Why	How	Impact	Counter Measure
Bank Account Numbers	Cyber-criminals	Financial gains with the information.	Cross-Frame Scripting (XFS)	Financial payments to clients, credit monitoring, support and recover costs, brand damage	

Since the team is not familiar with the specifics of an XFS attack, Dashing Danielle is able to describe it to the team. In this case, an attacker will use malicious JavaScript in an iframe

that loads a page with the intent of stealing data. There are several mitigation techniques that can be used to protect against XFS. Considering that it is similar to a clickjacking attack, Dashing Danielle suggests the following mitigations:

- Preventing the browser from loading the page in frame using the X-Frame-Options or Content Security Policy (frame-ancestors) HTTP headers.
- Preventing session cookies from being included when the page is loaded in a frame using the SameSite cookie attribute.
- Implementing JavaScript code in the page to attempt to prevent it being loaded in a frame (known as a "frame-buster").

Dashing Danielle puts the mitigations on the board under the "Counter Measures". The team discusses these mitigation techniques and review the current coding and architecture in place.

What	Who	Why	How	Impact	Counter Measure
Bank Account Numbers	Cyber-criminals	Financial gains with the information.	Cross-Frame Scripting (XFS)	Financial payments to clients, credit monitoring, support and recover costs, brand damage	X-Frame-Options, SameSite cookie attribute, Frame-busting

After reviewing the architecture and code, it was recognized that the application already sets its session cookies with the SameSite attribute set to strict:

```
Set-Cookie: CookieName=CookieValue; SameSite=Strict;
```

However, the other mitigations are not in place and require a resolution. The product owner asks whether the proposed additionally countermeasures will be sufficient to resolve the open issue and whether the likelihood of an attack is high enough to warrant the additional effort. Dashing Danielle is able to produce research that shows automated tools that are used to attack their competitors. The product owner agrees to proceed, and Dashing Danielle describes and documents the steps that are needed to set the X-Frame-Options to "Deny" as well as the Content Security Policy setting frame-ancestors 'none'. Dashing Danielle also works with the development team to create proof of concept code that can be implemented in the code to deny the framing of the feature into another site.

The team is satisfied with the results from the threat model for this issue related to the stealing of bank account numbers. However, they don't stop here and instead move on to the next possible threat. Although the manual method of threat modeling is time consuming, you can see that it can be pretty thorough, especially when compared to the method using a tool which we will talk about next.

3.1.6 Threat modeling using a tool

Just like with the manual method of threat modeling, there are several options when it comes to tools that can be used to develop a threat model. Each of these tools has their own benefits and drawbacks. Some are free, some are commercial tools, some can even be as simple as using a graphical tool to just draw the architecture and annotate the potential threats. One of the biggest benefits of using a threat modeling tool that is purpose built to define threats is that they will identify the threats for you making these tools more efficient at identifying issues. The results should still be reviewed with the appropriate stakeholders to ensure that the findings are indeed valid.

> **THREAT MODELING TOOLS** Take some time to review the available tools that are out there for threat modeling. There are several commercial ones Like SecuriCad, ThreatModeler, and Irius Risk. However, this corner of application security has fewer tools and less mature ones than in other spaces. Two threat modeling tools that are freely available are the Microsoft Threat Modeling Tool and Threat Dragon by OWASP.

Both Microsoft Threat Modeling Tool (https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool) and Threat Dragon by OWASP (https://owasp.org/www-project-threat-dragon/) offer the ability to drag-and-drop items to a board in order to build the architecture out and show the dataflow. From there, each will generate a list of potential threats to the architecture. A simple diagram from OWASP Threat Dragon is shown below.

OWASP Threat Dragon – Simple Diagram

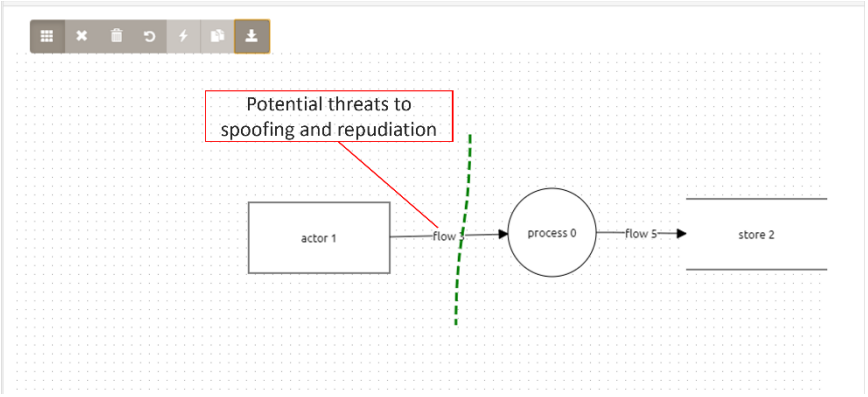


Figure 3.3 – Simple diagram from OWASP Threat Dragon

Threat Dragon can place threats in the STRIDE model, and with this simple diagram there are two simple threats identified automatically by the tool. One is related to spoofing and the

other repudiation. Both are on the interaction between the actor and the process. What this means is that there is potential for the actor to impersonate another user and potentially access components of the system that they would not typically be allowed to do. To fix this, the application must put in place a means of authenticating the user and knowing that it is indeed the correct user.

I won't go through the steps of the threat model using one of these tools since the effort is similar to what I described in the previous section using the manual method. With a tool in hand, the organization can scale the threat modeling process and centralize the review and storage of the threat model. This also allows the organization to threat model reusable components once. For instance, many of the applications in an organization may use the same authentication architecture. In this case the organization can threat model the authentication once and reuse that threat model for each application.

Using the example of Superior products, Dashing Danielle has reviewed several commercial tools for threat modeling, and has decided that the best tool for the job, and budget, is OWASP's Threat Dragon. She has created a process diagram and documentation that walks the developer through the use of Threat Dragon. She created a repository where all threat models from Threat Dragon will be stored.

Side notes about using a threat model tool

This approach to decentralizing and democratizing threat modeling with a tool allows for most technical resources in the organization to create a threat model using something like Threat Dragon. If an application security resource creates it, they will review their model with the engineering team. If a resource from the engineering team creates their own threat model, then it will be assigned to the application security team for review in the repository.

Dashing Danielle begins to gather the information on the common architecture that is used in Superior Products so that she can threat model those architectures and determine the open threats in order to get them remediated. Once she completes the threat modeling of the common architectures, she works with some of the resources in the engineering organization to evaluate her findings to make sure she was not missing anything. She also discusses the remediation options for the open threats that are found. Taking the feature that was used in the manual threat modeling session, she works with the appropriate engineering team to define the architecture in the tool. She generates the threats and compares that with what she found in the manual effort.

Chances are, in the story above, that Dashing Danielle will find discrepancies between what Threat Dragon found and what the team found in the manual process. This is to be expected as they are very different processes. Furthermore, the findings from the manual process will tend to be more specific and tailored to the architecture. The findings in Threat Dragon will be more generic. The use of both methods may need to be used to first identify the broad picture using Threat Dragon, and then diving into the details with a manual session using the output from Threat Dragon as an input into the manual effort.

Threat modeling is an early tool that can be used in the secure SDLC, but I'll cover how to identify security issues while coding next.

EXERCISE Download either Microsoft Threat Modeling Tool or OWASP's Threat Dragon from their respective download sites.

1. Get familiar with the tool and how to navigate through it.
2. Create a simple model similar to what you see in figure 3.3.
3. Once your model is complete with several stencils and drawn interaction between them, locate the threats identified by the tool.
4. Takes some time to think about the suggested mitigations and whether you agree that they would be effective. If not, what would be stronger mitigations against the threats.
5. You can document your mitigation strategies in the tool.

3.2 Security analysis tools

During development there is potential for security issues to be introduced unintentionally, and less commonly, intentionally. These security issues can come in all levels of risk and technical implications. However, organizations do not need to rely solely on penetration testing and other tools and techniques to uncover issues later in the development life cycle. There is an abundance of tools that development teams are able to use in order to locate an issue before it becomes a production incident.

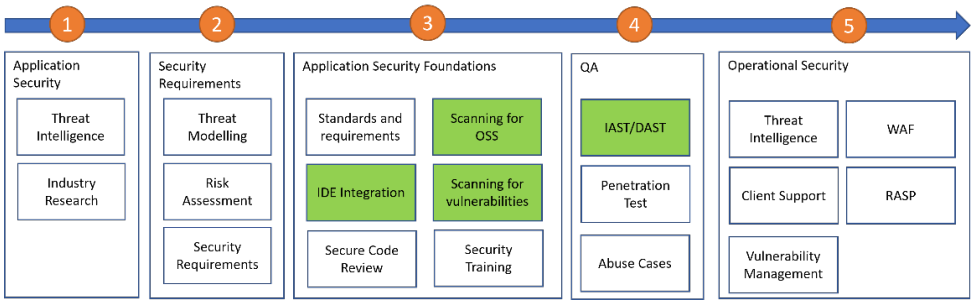


Figure 3.4 Scanning tools in the Secure SDLC

Before we jump into the available tools, you should first know that a lot of tools on the market are *noisy*. This means that there is a lot more noise than signal and it is up to the organization to ensure that they have reduced the noise or run the risk of a failed adoption of the tool. This noise is often referred to as *false positives,* which means that the issue identified by the tool is not an actual security issue. Determining whether output from a scanning tool is indeed a security issue versus a false-positive takes effort by the security team and the development team. For example, a tool may identify a SQL Injection issue from the scanning tool. Depending on how the team manage results from the tool, the development team or security team will first triage the issue to understand whether it is indeed an issue. If you've ever worked on a software support team or otherwise have been involved with reviewing defects or bugs in an application, you will be familiar with this

process. It requires knowledge of how the application is actually being used as well as the access to the code to follow the logic.

> false-positives may seem like it's just a matter of just working through the issues and closing the false-positives while keeping the true positives. The truth is that false positives have a larger impact on the organization. Time is spent identifying them instead of working on other priorities. Additionally, a large number of false positives will reduce the confidence in the tools being used, and by extension, the confidence of the security team.

Similar to false-positives, false-negatives need to be considered when using analysis tools. This is where the tool failed to identify a true-positive. Consider that you have integrated an analysis tool into your development pipeline, this will give your development team and your security team the confidence that issues will be identified and resolved. However, down the road perhaps a penetration test is completed on the application and a cross-site scripting (XSS) issue is found. Depending on the analysis tool, this most likely should have been found earlier and resolved. This is an example of a false-negative.

> Make no mistake, false-negatives can be as bad, if not worse than, false-positives. Whereas false-positives will grind teams down with the amount of work that is required to filter out the issues, false-negatives have the result of giving the organization the false sense of security. You expect your tools to provide the comfort that they are uncovering the issues so that you may resolve them.

There are many different tools out there to analyze code and applications, and many of them fall into more than one category. However, when it comes to security analysis tools there are three main ones: static application security testing, dynamic application security testing, and software composition analysis. We'll cover static analysis first.

3.2.1 Static application security testing

Static application security testing (SAST) tools look at code as it sits. In other words, it is doing a code analysis on the source code and is looking for security issues. One common finding with SAST tools is plain text passwords that are hardcoded in the code. SAST will do this by source code by using techniques such as *taint analysis* and *data flow analysis*.

- Taint Analysis – This allows the analysis tool to follow user input throughout the application to determine whether it is ever sanitized before it is used.
- Data Flow Analysis – This is where the analysis tool attempts to gather run-time information while the code is static.

SAST tools are primarily used at the time of development so that issues can be uncovered and resolved earlier in the development lifecycle. Many of the SAST vendors today have integrated development environment (IDE) plugins that allow the developer to run a scan when code is being written. Some of these plugins are free, but I often say, "you get what you pay for", so always take the free tools with some healthy skepticism. These free tools are often offered without support or are not as frequently updated as a commercial tool.

Many of the tools I will talk about are used in conjunction with one another as there is not silver bullet tool that will solve all of your security issues. For instance, static analysis tools are great to get an understanding of "hot spots" where the application appears to be weak from a security point of view. It can also be input into threat modeling exercises to allow the participants to focus on mitigations in trouble areas.

Each SAST vendor has a different way of scanning however most follow a pattern of compile, model extraction, pattern matching, and flow analysis as shown in figure 3.5

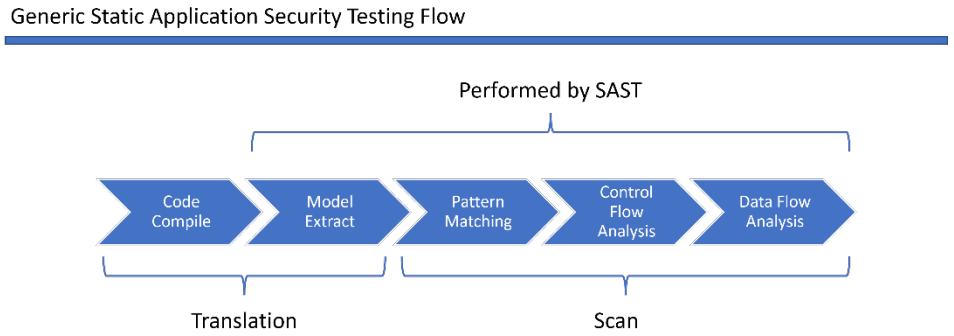


Figure 3.5 High Level SAST Process

There are certainly strengths and weaknesses with SAST tools. Some of the primary considerations are that SAST tools are usually bound by limitations in the languages that they cover which means that if you work in most large organizations, you will have several languages that are used by the different teams. You may be forced to use more than one SAST product to get coverage of all the languages in the organization. Additionally, SAST products are the most notorious products for producing a lot of false positives. You may be able to tune the product to reduce this noise, but this will take time and coordination with the security and engineering teams. Lastly, the SAST tools are only able to see the static code and cannot see how it runs in an environment. This limits its ability to locate potential run-time security issues like parameter tampering.

There are a few things that SAST does well. It is ideal for locating low hanging fruit like hard coded secrets or passwords as well as locating poor secure code practices like SQL injection type flaws or poor encryption methods. It can also pinpoint exactly in the code where an issue will manifest and give recommendations on how to fix it. This is extremely helpful for developers, and security folks, who want to fix a security issue as close to the code as possible.

Ultimately, SAST tools are as close to the developer as it gets. For those, myself included, that want to ensure that the right tools are available to the developer while they are creating the code, nothing really beats SAST. This needs to be weighed with the strengths and weaknesses mentioned above. Static analysis tools are decent at finding vulnerabilities early,

but in order to get information to the developers as early as possible, you need to go to where they are.

3.2.2 Tools in the development environment

When we look at the development pipeline the goal is to identify and resolve potential security issues before they are deployed to a production environment. This requires that tools are available, properly tuned, and that the developers know how to resolve a found issue when it's been identified. This also means that the organization should strive to find issues as early in the development pipeline as possible in order to provide the appropriate mitigations before it becomes a production vulnerability that puts the organization at risk.

To make this a reality, many of the tools that I mentioned previously have developed plugins or other tools to provide developers guidance on secure code as early as possible. For instance, many vendors will have an integration with development environments (IDE) that developers work in to write code. Others will have standalone tools that developers will be able to leverage to do things like locate secure third-party libraries before they use them in their project. This can come in the form of internet browser plugins or other services.

As with the other tools in the develop pipeline I covered earlier, there are commercially developed ones as well as free and open-source ones. However, you get what you pay for. The free ones, in general, will have less features and less support, but will still create a quick feedback loop for developers and identify the low hanging fruit.

The goal of these developer tools is to enable the developer to get in front of a potential vulnerability while they are in the process of writing the code. There is one tool that we can look at as an example here. It's a plugin called FindSecBugs that can be enabled in several well-known IDEs like Eclipse, IntelliJ, and NetBeans. This plugin provides a general static analysis security scan to locate potential vulnerabilities in the code. Developers can initiate the scan and FindSecBugs will produce a listing of the vulnerabilities that were found and when the developer clicks on the finding, FindSecBugs will take them to the line of code in the IDE.

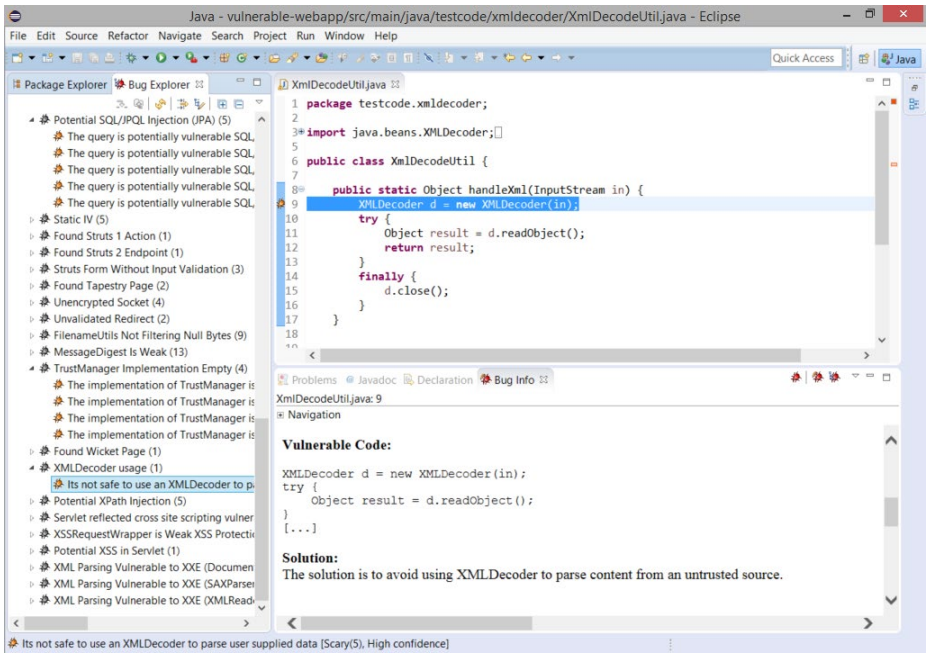


Figure 3.6 – FindSecBugs in Eclipse

Along with showing the line of code that has the vulnerability, FindSecBugs will provide reasons why it is vulnerable and possible solutions. There is a myriad of other similar tools, and as mentioned, most vendors with security solutions, especially in the SAST and SCA space have developer tools.

The goal here is to provide the developer the resources to resolve a potential vulnerability prior to them checking the code in to the code branch. Other tools in the pipeline, like the ones I outlined previously, are still needed however having the ability to catch an issue before it goes to a code branch means that there is less reliance on these tools to find vulnerabilities when the code has already moved out of the developer IDE, and may reduce the amount of effort spent in these other tools to find and resolve issues.

3.2.3 Dynamic application security testing

Where SAST and IDE integration tools are looking at the code in its static form, dynamic application security testing (DAST) will scan the application as it is running in an environment. It can be thought of as a penetration test that is performed by a tool considering that it is a security test that is looking from the outside in and attempts to locate weaknesses such as parameter tampering, cross-site scripting, improper redirects, and so forth. DAST tools are often used by penetration testers to identify low-hanging fruit in a running application and take some of the manual work out of a penetration test.

DAST tools are mostly technology independent since they are looking at the running application and attempting to find weaknesses. They are taking the outside-in approach and rely on the HTTP conversation as the common ground and therefore is not concerned about the underlying language or framework.

DAST tools rely on being language independent, and able to scan running applications to deliver the ability to scan applications in production environments and even applications that are not owned by the organization. Although both of these require prior approval from the application owner since these scans can be destructive and create disruption for the application. Additionally, DAST tends to produce fewer false positives over SAST tools since many of the findings are identified in the running application.

However, with the additional flexibility that DAST provides over SAST, there are some drawbacks. For instance, DAST tools will not be able to tell you the line of code where an issue is found unless it has been instrumented into the application. More on that in a moment. DAST also tends to be run later in the development lifecycle meaning that the issues are found farther right in the life cycle as opposed to SAST. It is possible to run DAST earlier on a developer's local environment, however, this is not frequently done. DAST will also not discover code specific issues such as hardcoded password. Lastly, the findings from DAST still need to be triaged by a security subject matter expert to determine whether it is a true positive.

Open Source DAST

There is an open source DAST tool available from, surprise, OWASP. It is called Zed Attack Proxy (ZAP). It is free to download and use and is a great way to get your feet wet with a DAST tool. You can usually set it up to run an un-authenticated scan against an application in under a few minutes by just providing a URL. However, the real power of ZAP comes from using an authenticated session that crawls the site, determines the site map, and then begins to run through common attack patterns to report on vulnerabilities. It is a great tool to learn with, but many organizations use ZAP as their sole DAST tool as well. (https://www.zaproxy.org/)

There is another variation on DAST called interactive application security testing (IAST). The uniqueness of IAST is that it combines the strengths of SAST and DAST. It assesses the application from within through instrumenting the code. This means that the vendor will provide a library that the application then uses in its overall build of their application so that the IAST tool is running as part of the application. This allows it to have access to code, the HTTP conversation, library information, backend connections, and configuration information.

One of the drawbacks of IAST is that some tools available require the application to actually be attacked in order to detect a vulnerability. This may not be a huge deal for the organization so long as they have robust testing that enables the bulk of the application to be tested. Failing to have this robust testing means that a vulnerability could go undetected until that feature is exercised. A prime example of this would the case where a reporting function is only periodically run. If this is not run as part of normal quality assurance or regression testing, then it is possible that a vulnerability in the report function would pass through to production. In this case, if the organization has good integration with its logging

and reporting functions, the vulnerability would be picked up in production and allow the organization to respond accordingly.

Where IAST shines is the ability to work well in a DevOps model. I will cover DevOps, and more specifically DevSecOps in coming chapters but for now just know that IAST provides constant monitoring of the application for vulnerabilities, and a much lower case of false positives. You are also able to tailor IAST to focus on specific areas if time is a factor. This means that you can focus on a small section of the application for testing and review the results from the IAST tool.

It's not all great news though. IAST means more complexity with your build. That library that you integrate into your software needs to be updated periodically and could cause build failures. This means more development and deployment work by the engineering team. As mentioned IAST can only report on what it sees. If the part of the application is not exercised, then the tool will not uncover any issues.

DAST and IAST are great ways to uncover issues in a running application and can augment the overall tool chain in a secure software development lifecycle and attempt to model the behavior of a real attacker. It's important to remember that these tools are only a component of the defense in depth model and are not used to provide assurance that all vulnerabilities have been found. What about finding vulnerabilities in software that you use to build your application, but it's not actually owned by you? That is where software composition analysis comes in.

3.2.4 Software composition analysis

A house or a car from the outside looks well put together and looks like one object. We all know, however, that there are multiple components that are used in building that final product. Some of them are small and discreet items like screws, bolts, and nails. Others are more complex individual systems like assemblies with electronic systems that are sold as a total package.

Software is no different. A small percentage of code is actually written by a developer. In most cases the developer pulls libraries and packages into their overall project that meet a need. For instance, a developer is not going to create their own project that handles math equations, there is already a library for that either existing in their framework or from another source. This is convenient, but it becomes difficult to manage the sprawl of libraries that are used in an overall project. How do you know the libraries being used are secure or are not running afoul of license use?

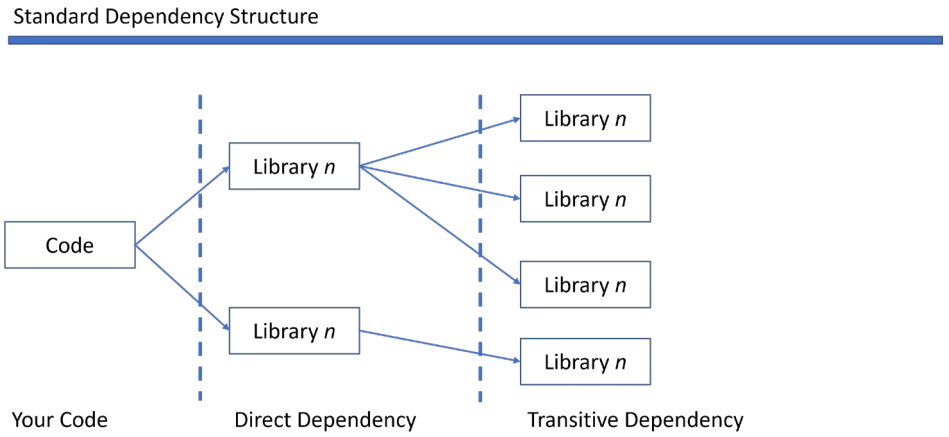


Figure 3.7 – Example dependency structure

That's where software composition analysis (SCA) can help. Most typical SCA software is used to manage open-source component use and tracking of the licenses. SCA tools perform:

- Scans of an application's code base, including related artifacts such as containers and registries.
- Identify all open-source components to help build a software bill of materials
- The libraries license compliance data and any security vulnerabilities that may be known.
- Some SCA tools also help fix open-source vulnerabilities through prioritization and auto remediation.

Sounds pretty good. However, one of the issues with using an SCA tools is that it can often flag a library after it is running in production. Image that you built and deployed your application in January of the year, and in July your SCA tool flags one of the libraries you used as being insecure. At the time you initially built the software there was no issue. It gets even more complicated, what if the library requires you to upgrade other libraries or otherwise make a larger change to the architecture in order to resolve the finding? Another possible scenario is that the library that is flagged by the SCA may itself not be insecure, but rather a library that it depends on.

> **A *direct dependency* is functionality exported by a library, or API, or any software component that is referenced directly by the program itself**
>
> **A *transitive dependency* is a functional dependency which holds by virtue of transitivity among various software components.**

SCA can be a huge benefit to the development team in identifying potential issues with third-party libraries. However, without the means to provide an updated library in a short period of

time the development team will be stuck knowingly running a vulnerable library. Finding out whether something vulnerable in a direct dependency or a transitive dependency can by difficult and often requires the development team to debug or do a thorough code review to determine whether the application is truly vulnerable. Once the vulnerability has been confirmed, the most likely resolution is to upgrade to the latest release of the library. Rarely is it possible to "neuter" the library so that the application is no longer susceptible.

One other drawback of SCA is that the majority of them rely on known weaknesses, primarily from sources like the National Vulnerability Database (NVD) that catalogs known vulnerabilities through a common language called the Common Vulnerability Enumeration (CVE). These identified vulnerabilities are submitted through various sources and made available for consumption by tools. You can see more by going to the https://nvd.nist.gov/ and review the latest opened vulnerabilities, review older ones, and search by component. What needs to be considered with SCA tools that report CVE's is that this is only for the known vulnerabilities and does not cover zero-day vulnerabilities in a library. This is where the vulnerability has not been reported and therefore there is no fix that can be released.

As mentioned, this is all great information, but without being able to act on the information the organization is only able to know that they are running in a vulnerable state. We'll talk more about DevSecOps in future chapters, but the critical takeaway here for SCA is to know that once a library has been detected as being insecure, a path to deliver the newer and more secure version of the software needs to be quick and clearly defined.

SCA is one critical component in the defense in depth model to provide not just security scanning, but also aid in the collection and cataloging of the various libraries used by an application.

EXERCISE

1. **Find a CVE on the NVD. You can find CVE's by browsing by Year and Month. Click on an individual CVE to get the information on that specific vulnerability.**
2. **Use the following CVSS Examples**
3. **Use the CVSS Scoring Calculator to determine the CVSS v3.0 Base Score.**
4. **Document your finding using the following format on the CVS Examples Site :**
    a) **What's the Vulnerability?**
    b) **What's the Attack?**
    c) **What is the CVSS v3.0 Base Score**

3.3 Penetration Testing

One thing about the tools that I just covered is that they will never replace a good old fashion penetration test. These are performed by highly skilled researchers that are skilled at finding ways into a system and application. They are not limited to the confines of the rules that govern the tools that we previously talked about.

> **EXERCISE** Do a quick search for a job description of a penetration tester. The range of skills needed is quite impressive. Not only do you need to have technical abilities, but you will also be able to perform social engineering and physical security testing. It's no surprise that these jobs are in high demand.

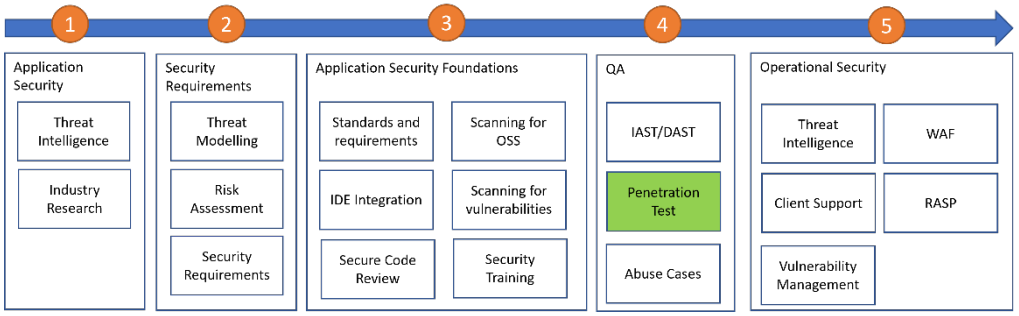


Figure 3.8 Penetration testing in the Secure SDLC

There are several types of penetration tests. At the high level there are penetration tests that occur from an internal team, like a red team within the security organization. There is also a penetration test that can be coordinated with an external party.

> **DEFINITION** *Internal penetration testing* is completed by a team inside the organization, often referred to as a *red team,* who are employed at the target company. This is a team/group that has other duties at the company but is engaged for a period of time to target a specific system/application. *External penetration testing* is an external party that is engaged to test the system/application. The scope is defined, and the party is given a timeframe for completion.

Which testing path the organization goes depends on a few factors. In many cases, the organization is not staffed to support a full or part time, internal penetration testing team which makes it difficult to complete internal testing. Additionally, the organization may be required for compliance, or contractual reasons to complete a penetration test from an external organization. Most organizations will opt for the external source of penetration testing which allows them to pull a vendor in when needed. Regardless of the direction, there are several high-level approaches to penetration testing:

DEFINITION *White box testing* is where the organization provides information about the system to the tester. This can include code, credentials, network maps and other system information. *Black box testing* is where the organization provides little to no system information. This resembles a typical attack where the information that can be gathered is generally only public information. *Grey box testing* is the in-between state. Some information is offered, but it is limited to just essential information.

Each of the approaches can be used whether it is an internal or external test. The outcome is more important. For instance, in a black box test the tester is given no information to start with, which will closely mimic an outside attacker. This is much harder to achieve with an internal team given their alignment within the organization. This testing tends to be more closely associated with a grey box test where the attacker has some, but not all information. Most testing that is done through an engagement with an external vendor, or with the internal team, is a white box test. Especially if the output from that test is used to meet an obligation such as compliance and contractual agreements. There is no right answer on which is best. Each type needs to be considered against the goals of the organization. However, as mentioned, white box testing tends to be more for compliance and black box is generally more for a true understanding of the security of the application and organization.

The great thing about penetration testing is that the findings tend to be true positives that are actionable considering it was found by a simulated attack. However, in the case of white box testing, the true external security controls may not have been in place. For instance, it may be required that to exploit a particular vulnerability, that the attacker would need access to an elevated account. If that elevated account information was given to the attacker at the time of the test, which is well within the parameters of a white box attack, this means that a true attacker would need to compromise that account. Perhaps there is good privilege access controls and multi-factor authentication that are associated with that account and therefore the risk is very low.

Other benefits of a penetration test are that they can be scoped to specific areas and time. This means that you can request that a test only focus on a particular feature for a set amount of time, like 24 hours. The penetration test can also be used in combination with other security methods like threat modeling and scan reports. Providing this information to the penetration tester will help them cut down on steps and provide them a map to weaker areas of the application so that they may be able to focus their time and effort there. Additionally, some of the work that was done in your threat model can then be verified through the penetration test, like the security controls that you described as being in place to mitigate a found threat.

3.4 Run-time protection tools

Where the tools and processes that I talked about before are geared toward identifying vulnerabilities and risks, there are tools that are used to provide protection against application-level attacks during run-time in a production environment.

Run-time application security protection (RASP) is a security technology that uses runtime instrumentation to detect and block computer attacks by taking advantage of information from inside the running software. This will sound very similar to IAST where the tool has the ability to see into the application and watch attacks as they happen. There is not

a lot of difference between RASP and IAST with the exception that RASP functions as a run-time protection tool and IAST is focused on observing and reporting on found vulnerabilities.

RASP technology can improve the security of software by monitoring its inputs, and blocking those that could allow attacks, while protecting the runtime environment from unwanted changes and tampering. RASP can also prevent exploitation and possibly take other actions, including terminating a user's session, shutting the application down, alerting security personnel and sending a warning to the user.

Another common tool used to provide runtime protection is a web application firewall (WAF) or sometimes referred to as application security manager (ASM). A WAF is an application firewall for HTTP applications that applies a set of rules to an HTTP conversation and analyzes bi-directional web-based traffic. Generally, these rules cover common attacks such as cross-site scripting (XSS) and SQL injection. When the WAF recognizes a pattern that looks malicious it will either report or block the attack depending on the configuration. Most WAF vendors also offer protection against robotic attacks (bots) like DDoS. This is largely a volumetric protection, by many vendors are becoming savvier by including machine learning and artificial intelligence to be more proactive in its overall protection against bots and other abusive behavior.

Web Application Firewall (WAF) Flow

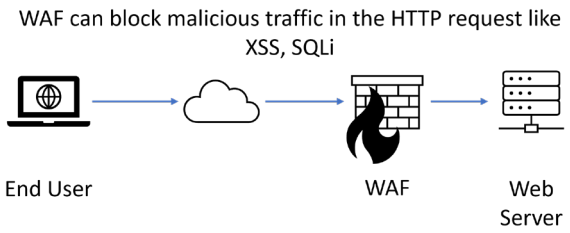


Figure 3.9 WAF integration with web servers

This is similar to RASP in the spirit of blocking malicious traffic. However, where RASP is typically run within the application itself or on the same host system, WAF can be cloud-based or otherwise external to the application. Most organizations today are moving to a cloud-based solution for WAF so that they have a managed platform for protection and a much faster adoption path. WAFs may come in the form of an appliance, server plugin, or filter.

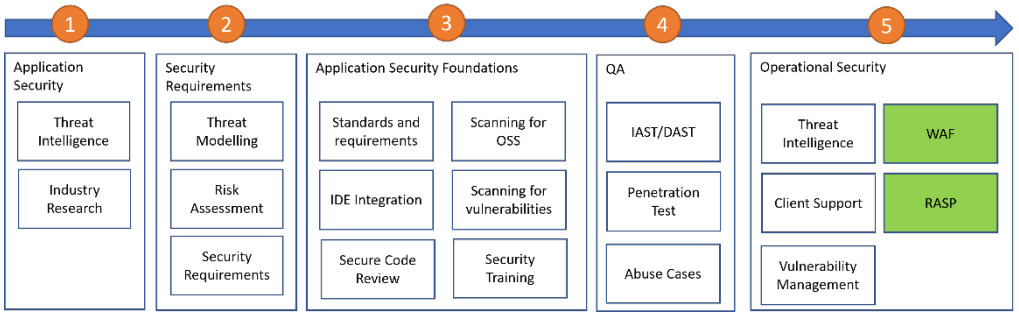


Figure 3.10 Runtime tools in the Secure SDLC

These protection tools, whether WAF or RASP, help the organization by providing the runtime reporting and blocking when an attack is discovered in real-time. The drawbacks are few and simple. The tools need to see something happening in order to report or block so they cannot provide any information to the development team so that they can prevent a vulnerability from going out in the first place. There is also the very real potential for one of these tools to block legitimate traffic. There are plenty of scenarios where a customer may be using the system within the parameters and still trigger an alert or a block. Especially when volumetric types of activity are detected like running large batch jobs.

Another consideration when running any of these protection tools is that the rules that govern these tools need to be well vetted and managed. Both WAF and RASP require rules to be configured that tell the tools what to look for and what to do when it sees something that has been configured. For instance, if the WAF sees an HTTP conversation that includes patterns matching a SQL Injection attack, should it alert, or block. If it alerts, where does the alert go and what is the expected action. If it blocks, what is the user experience. Taking the out-of-the-box rules that come with the tool are often too broad and will alert on everything that it sees. Most organizations are not prepared for the flood of potential alerts. Additionally, some behavior that looks suspicious for one application may be completely legitimate for another. The rules for these tools must be well understood, tested, and managed between the application security team and the application development team.

3.5 Vulnerability collection and prioritization

All the tools have been integrated, and you are successfully running penetration testing. Great! You now have, most likely, a ton of vulnerabilities to process. Each from a different tool, found in a different part of the process. You may not even know whether they are true positives or false ones. In most organizations you are running at least 2-3 of the tools that I discussed. For instance, the organization is probably running a SAST, SCA, and DAST tool at the least. Others might be running just a SCA, DAST, and a WAF. Some may be running all of them. Each of these can be noisy if they are not properly tuned to the organizations needs and processes. Additionally, the may each have their own user interface with a dashboard of some sort. This means that the development and security team will have to log in to multiple tools just to get the vulnerability information that they need.

3.5.1 Integrating with defect tracking

In order to be effective at managing vulnerabilities the first thing that most organizations will do is integrate the security tools with their defect tracking tool. That might be Jira, or Bugzilla, or something else, but in any case, this will allow the developers to see the vulnerability information as it becomes known from the various scanning tools in the organization. For instance, the DAST tool that is integrated with the testing environment may detect a vulnerability when the developer's code was deployed to the testing environment. This DAST tool can then call an API with the defect tracking tool and open an issue to either the application security team to triage, or directly to the development team to resolve. Which one depends on the confidence of the security tool that detected the vulnerability and the maturity of the organization.

Opening defects on teams will help get the organization to resolution on the open vulnerabilities but being able to have visibility in to the all the open vulnerabilities across an organization can be challenging. There are usually different defect tracking tools being used unless the organization has tried to standardize. This means that the application security team cannot necessarily rely on building a dashboard in a single defect tracking tool. Often the application security team will build their own dashboards that pull information from the different defect tracking tools, or the security analysis tools that are being used. The application security team is then on the hook for managing this dashboard and enhancing it along the way as technology changes.

Assuming that the organization has access to a centralized dashboard that shows the various open vulnerabilities, the hard part becomes apparent:

- How do you get these vulnerabilities closed?
- How do you prioritize the vulnerabilities?
- Who owns the vulnerabilities?
- What if the vulnerability cannot be closed?
- Is there enough information for the development team to provide mitigation?

These are all questions that each application security team faces when the begin to review the open vulnerabilities. One of the most critical questions above is regarding ownership. Software is complex and is pieced together by several different components as I discussed earlier. When a vulnerability is found, for instance, in a specific part of an application, it is often difficult to locate the team that has developed that code. This is especially true where an organization has something like a team that develops common components that are leveraged by most of the applications in the organization. The application security team needs to have an inventory of the applications in the organization as well as resources to contact that are responsible for resolving found vulnerabilities. This may be not just the engineering leader for the application like the technical manager, but also the project manager and even the product owner. Both of the later resources typically maintain the control over what gets worked on in a given timeframe, not the engineering leader.

3.5.2 Prioritizing vulnerabilities

Before even being able to request that the engineering team tackle a vulnerability, it's important to ensure that they organization has clear direction on timeframes for resolution.

Although every organization is different, and there is not a well-established industry standard on timeframes, the organization should strive for resolving critical and high as quickly as possible, while still having expectations on closing the medium and low.

Severity	Time to closure
Critical	< 30 days
High	< 60 days
Medium	< 90 days
Low	< 365 days

Although the above is generally what is seen in the industry, each organization may have different closure expectations and may even be directed by regulation, contracts, and other external pressures to have tighter timelines. Keep in mind that these vulnerabilities require code changes. This is separate from vulnerabilities that are found at the host level, like a Windows patch, which might have much shorter time frames for resolution. However, without clearly describing the expectations on closure, the organization will have a difficult time giving the engineering organization a target with the open vulnerabilities.

3.5.3 Closing vulnerabilities

Getting to resolution on the open vulnerabilities will take a very close partnership with the engineering teams that are responsible for resolving them. Frequent touch points to review the vulnerabilities and ensuring that they are true positives, have clear expectations on timeline for closure, clear steps to resolve, and a method in place to retest are the responsibility of the application security team. Application security will help the engineering team prioritize what is the highest risk to the organization using the methods I talked about before with a risk rating process. This can be done through weekly meetings between the application security team and the engineering team where consensus will be reached on when the vulnerabilities will be resolved in an upcoming release. To accomplish this the engineering team may take on a burst of effort to resolve a batch of issues like in a bug blitz or a security defect release which are concentrated and focused efforts to remediate open vulnerabilities.

It is important for the application security team to identify common threads in the vulnerability data that they are collecting. For instance, are there frequent issues with encryption that should be addressed holistically by a more general approach like a centralized encryption program that addresses the key management, and sets standards on how, where, and when data should be encrypted. With this type of approach, it is possible to eliminate several vulnerabilities in on larger project a while creating a more sustainable security model going forward.

The most important component of being able successfully tackle the abundance of vulnerabilities that will be produced from the security tools that are integrated is getting senior leadership buy-in on the resolution effort. To facilitate this, it is critical to provide data that is trustworthy, complete, and shows the impact to the organization. One pitfall to avoid with bringing vulnerability information to senior leadership is to ensure that there is

confidence in the data that is being presented. Nothing will hurt your cause more than having numbers that are frequently different with little explanation. Numbers will change as you resolve issues and bring on more or different tools. But if your method of gathering information is unreliable, your data will become unreliable, and therefore undermine your effort to convince senior leadership that you need them to help you drive down vulnerabilities. If the senior leaders feel that you will be wasting their team's time and resources, they will be less likely to support your effort.

3.6 Bug bounty and vulnerability disclosure program

Vulnerabilities that you generated from your tools and processes are pouring in, but what if you are looking for a different set of eyes and hands on your application? A common method for mature organizations to receive vulnerabilities from a broad audience is through a bug bounty program (BBP) or a vulnerability disclosure program (VDP). Both programs open the organization to receive vulnerabilities from external sources that are not affiliated with the organization. For instance, in a VDP, the organization will create the boundaries, communication paths, and expectations for individuals that want to locate security issues with the organization's applications. The organization will post their policy on their main website, usually on a security page, so that it is easy to locate for those that find issues within the organizations applications. These individuals, often called security researchers, will be able to look for security issues in the organizations applications and then submit them through the appropriate channels in order to get a resolution. This differs from the BBP where researchers are paid a bounty for issues that they find in the application, sometimes by an intermediary organization.

3.6.1 Vulnerability disclosure program

VDP's follow the simple idea of "see something, say something" and allow for researchers to uncover security vulnerabilities without the fear of retribution. Additionally, a VDP simplifies the process of getting this security information to the right team. One of the issues that many researchers have is knowing where to send information. Additionally, once the information has been submitted, the path to resolution is often opaque to the researcher. In many cases, the researcher may not even get a response from the organization. The VDP attempts to solve this by providing the following five components of the policy.

Table 3.1 Components of a VDP

Component	Details
Purpose of the policy	This section should include the purpose of the VDP for your organization and to demonstrate the commitment to more secure applications.
Scope	Probably the most important part of the policy. This is where the organization sets the boundaries for what is allowed and what is out of scope for any issues reported. For instance, the policy could specify that client data is never to be accessed or removed from the applications and that only certain parts of the application are allowed to be tested.
Safe Harbor	This provides assurances to the researcher that they will not be prosecuted or have legal actions taken against them by the organization as long as they stay within the scope identified within the VDP.
Process for submission	This section sets the expectations for how the researcher should submit their findings as well as the expected quality and content that is expected in the report.
Expectations	This is what the researcher should expect from the organization as it responds to the submitted report. Examples would be the time between submission and response, when the finder may publicly disclose their finding, and what the communication will look like between the researcher and the organization.

3.6.2 Bug bounty program

The BBP builds on the VDP by providing an incentive structure to the submitted vulnerabilities. In the VDP the organization will pay based on a predefined set of guidance. This will put a price on specific vulnerabilities like SQL Injection, or the organization may structure the cost by severity where critical issues are priced higher than low ones. Regardless of the structure this need to be determined ahead of time and the leadership team needs to be ready to provide periodic payouts for bounties. The common BBP has two options:

> **DEFINITIONS** *Private BBP* is a VDP program that is only available to researchers that are invited to the program. Whereas a *Public BBP* opens the program to any researcher that wants to participate.

As vulnerabilities are submitted, the organization will need to triage them. This is typically completed by the application security team in conjunction with the engineering team. The vulnerability should be determined to be a true positive, and the severity agreed upon. Depending on how the policy is written, it may allow for the public disclosure of the vulnerability should it pass the allowed timeframes. This means that it is imperative that the organization has a clear process for resolving the issue within the timeframes that are in their policy and maintain communication and coordination with the researcher. The news is

littered with cases of a vulnerability disclosed to an organization where that organization failed to stick to their own policy, and it instead became a very public matter.

3.6.3 Third party help with vulnerabilities

There are several third-party programs that exist that will assist the organization with setting up and maintaining both a VDP and BBP. These services will assist you in creating the VDP and host it on their site. They will also be the middleperson that accepts reports, checks them for quality, and in some cases they will de-duplicate them against a list of vulnerabilities that you may have already identified through your internal tools. For instance, a researcher may uncover a XSS issue in your application. In this case, the third party will match that against vulnerabilities that you have already identified and provided to them in order to determine whether this is a new issue or existing. They will then respond to the researcher that the issue is already identified and provide expectations for remediation.

There are several prominent companies that provide this type of service like HackerOne and BugCrowd. However, there are also many companies that run their own bug bounty like Intel, Facebook, Google, Apple, Microsoft, and many others. In these cases, the payouts and rules of engagement are clear and provide these organizations with the ability to have another channel for getting vulnerability information.

EXERCISE Take a look at Microsoft's:

https://www.microsoft.com/en-us/msrc/bounty?rtc=1

and Google's:

https://www.google.com/about/appsecurity/reward-program/

Bug Bounty Program (BBP). Understand the scope and the rules of engagement and think about how this would work in your own organization.

3.7 Putting it together

I covered a lot of ground in this chapter on the different tools and processes that may be used during the secure development lifecycle. You may be wondering how all of these fit in and where they make the most amount of sense. As I mentioned previously, security is the business of applying the right amount of mitigation to reduce risk to as low as possible for an organization. It's critical to ensure that you are not overspending. The simple example that most of us can relate to is a car repair. Most of us would never pay for a repair that cost more than the actual vehicle worth. Security is no different. You can apply every tool that I mentioned in this chapter, every process, build a massive application security team and pay for a massive BBP. Will that make you more secure? Yes. Will it cost you more than what it is that you're actually trying to protect? Most likely.

In previous examples I showed the development lifecycle, the various components, and where you can apply technology, process, and people. Taking another look at that example SDLC with the tools from this chapter integrated looks like the following:

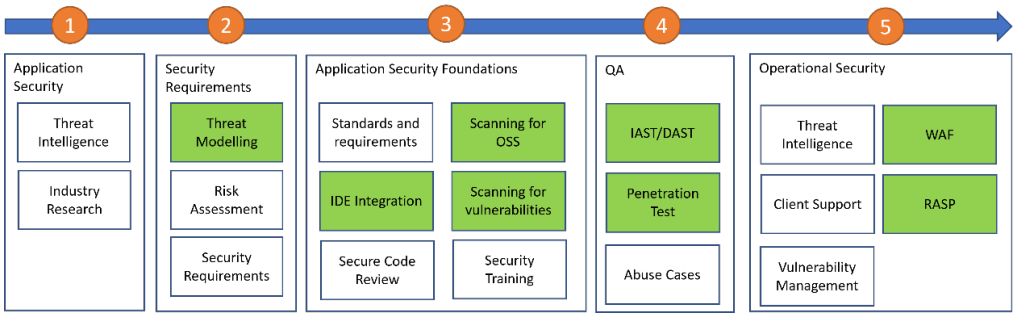


Figure 3.11 – SSDCL with tool integration

In figure 3.11, the tools I identified in this chapter are highlighted in green. We can use the example of Superior Products from earlier to help illustrate how these tools should be integrated.

I spoke earlier about how Dashing Danielle from Superior products was able to perform a threat model with the development team in order to identify open threats. Based on that threat model, the development team was able to define several requirements that will mitigate against the identified threats. Superior Products then begins the coding effort on the feature that will allow users to link their bank account in the application so that they can get paid for selling items. While coding, the developers take advantage of IDE plugins that show them when code that they are developing might potentially introduce a security issue. This reduces the introduction of additional vulnerabilities that may not be caught until further in the development lifecycle. One developer has additional questions regarding a found vulnerability from the IDE tool, and Dashing Danielle is dispatched to work with the developer to determine a more secure approach.

The developers are able to use the browser plugin available from their SCA tool to determine whether the library they are preparing to use is secure or not. The SCA tool integrated with the developer's local build identifies a few old components that are being used in the application. Although, these findings are already existing and not related to the current coding, the developer now has the opportunity to locate a more secure version of the components and integrate them.

The code is now moving to a testing environment where test analysts are able to run pre-defined test plans. In this stage, security tools that test the application in a running state like IAST and DAST will be used to identify security issues that need to be resolved before an application gets moved to a production environment. Superior Products has decided to use a free and open source DAST tool to determine runtime issues with the application. This tool identified several issues that need resolution. Because this was located in a test environment, the development team has an opportunity to resolve it before code is pushed

to production. Dashing Danielle is alerted of the findings when the DAST tool opens three vulnerabilities in the development teams defect tracking tool. She takes some time to triage the issues in order to determine whether the issues are true positives. Luckily, only one of them was determined to be a true positive, however, it will take time to fix and may delay the release. She does a risk rating on the issue and builds out the recommended solution with an approximate time to resolution. She then takes this to product owner to get it prioritized.

There is some discussion regarding the issue found in the DAST tool with the product owner and the engineering manager. Resolving the issue will require rewriting a component related to how authorization is managed in the application. This is not a small task, but to not do it means that a user may be able to link another user's bank account. This is too significant of an issue to allow in the product. The product owner makes the decision to prioritize the vulnerability and leave another feature out of the release in order to balance the remaining work.

The new features are available in a pre-production environment, and Superior Products has engaged with a third party to complete a penetration test. They use the threats that were identified in the threat model as the basis for the scope of the penetration test, as well as the authorization finding in the DAST tool. The penetration test is completed with no significant findings, and the team is able to validate that the counter measures they identified in their threat model are indeed providing mitigation.

The features are finally ready to be deployed to a production environment. Given the vulnerabilities that were identified and the team's ability to remediate them, Dashing Danielle recommends that integrating with the enterprise WAF will provide sufficient run-time protection against what they deem are the most likely threats that they will face.

The product goes to production behind the WAF, and the customers are now using the features in the most secure manner that Superior Products could provide. All is well!

3.8 Summary

- Threat modeling is the process of identifying threats that may impact an architecture. It's used to understand the who, what, why, how, and counter measures as it relates to threats.
- There are two prominent methods to performing threat models. One through a manual process, the other by using tools.
- Analysis tools throughout the software development lifecycle provide feedback to the development team as early as possible. These tools are used to find issues in static code, running applications, and in the software components that are used to build the final product.
- Developer security tools are used to give developers quick access to potential vulnerabilities as early as possible. Many of these are available as plug-ins within the developers IDE.
- Run-time protection tools are used to provide protection when an application has been deployed to a production environment. Two of these tools are RASP and WAF.
- Penetration testing is the closest method to mimic an actual attack by a threat actor. These tests are performed by skilled people who are able to locate weaknesses in the application within scope.
- Vulnerability management becomes quickly overwhelming when multiple tools are finding vulnerabilities. The organization needs a common approach that builds confidence in the data and enables developers to act on the information. This gives the organization the greatest chance of getting ahead of the vulnerabilities.
- Bug bounty and vulnerability disclosure programs provide an avenue for researchers that are external to the organization to submit reports that identify vulnerabilities within the scope that the organization has provided. In the BBP, researchers are paid for their findings.

4

Releasing secure code

This chapter covers:

- How organizations can release secure code leveraging people, process, and technology.
- What a DevSecOps pipeline looks like and why it supports security better than other release methods.
- What differentiates a DevOps model compared to other models like Waterfall, Agile, Lean.
- How to take advantage of a fast feedback loop in order to provide security issues to the development team as rapidly as possible.

In this chapter I will show some of release methods that are in practice in most organizations. While some of these methods have been in practice for a long time, they can still be found in most organizations. While each have their pros and cons, release methods such as DevOps can support a more secure method of delivering software. If you are not familiar with DevOps, it is a set of practices that bring together development and operations to deliver software in an efficient manner.

> **DEFINITION** Microsoft defines DevOps as a compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

What has been historically the case for software development and release is that there is a product team, a development team, a testing team, and an operational team that all take part in the delivery of features and products to a production environment. This is typically done in a way where each team has gates that start and finish their part of the process. DevOps intends to streamline that process and reduce or eliminate the hand-offs between teams. Most organizations will accomplish this by creating a single team that owns not just the development of the code, but the testing and delivery as well. In some smaller organizations there may only be a few people on the team that do all of these activities

themselves, in larger organizations specialists would be used to perform specific tasks within the team. For instance, in these larger teams, there would be one or more people focused solely on the deployment, or just the testing, as well as having developers focused on the code. However, in all of these cases, the team would share the work items as one team.

DevOps is primarily a function of automation, tooling, and processes. This model works best when there is little to no manual work or processes. Some of the most successful organizations that use a DevOps model are able to deploy code from a development environment to a production environment in minutes. And they can do this multiple times a day. To achieve this level of speed and confidence, the organization will build a *CI/CD pipeline.*

> **DEFINITION** A CI/CD pipeline is where code is continuously integrated (CI) using an integration tool like Jenkins, and continuously delivered or deployed (CD) using deployment tools like Octopus or Ansible. Code is built into packages that are automatically pushed to a production environment (continuously deployed) or could require a manual step to push code to production (continuously delivered).

The confidence in deployment is built in through strong controls around how source code is tracked, merged, and versioned. Testing must occur all throughout this model, from the developer's unit test at the lowest level, up to the testing of API integration and system testing. Lastly, monitoring of the application in production for potential issues and having a rapid feedback loop to the development team will reduce the time to fix issues as they are found. This gives greater confidence to the development team that they can deploy and fix code rapidly. This certainly enables the developer to feel empowered, but where does security fit?

4.1 Security in DevOps

The general appeal of DevOps is that it allows for a rapid resolution of discovered issues. As an application security person myself, this makes me happy. This means that if the developer has the right information and knows how to resolve a vulnerability, then they can fix a vulnerability and have it running in production in a short period of time. Maybe even within a day! But getting to this point requires a few things to be in place and working well. Without getting into the specific tools, a successful DevSecOps pipeline provides the following capabilities:

- Educating, enabling, and empowering the development team to make security decisions during the development process.
- Fewer security gates in favor of a faster more targeted testing using automation.
- Vulnerability management that provides a fast discover>triage>report>resolve process.

I'll break each of these down here in this chapter. But first it's important to understand that the basic security tools and processes that I've covered in the previous chapters still hold true even in a DevOps model. There is still room for threat modeling, SAST, and penetration testing for example. However, where they are integrated and by whom changes.

4.1.1 DevOps pipelines

As described previously, a DevOps pipeline consists of automation that integrates and delivers code to a production environment. The tools that are part of that pipeline vary in each organization, but the fundamental pipeline for code to be delivered will resemble figure 4.1.

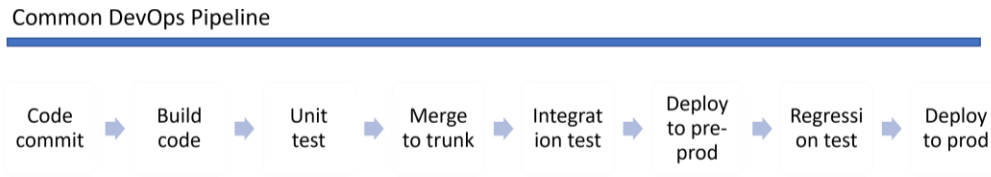


Figure 4.1 Standard components of a DevOps pipeline

In figure 4.1 the automation occurs at every step with the exception unless the organization has decided to add a manual step for deployment. For simplicity, in the rest of this chapter we will assume continuous deployment where the code is sent to production automatically as long as it has cleared all automated testing and checks.

For the pipeline as shown in figure 4.1 the code goes through the following process:

- The code is committed by the developer once they have completed development.
- It enters a build process that pulls together the different libraries and components together in order to create the application.
- The unit tests are run that test the code changes for any potential regression issues.
- If the build and test complete on the local level the code is merged to a trunk where additional tests are initiated in order to perform tests with the code change in the context of other components of the overall system, like API's and other products in the organization.
- If the merge and integration testing complete without failure, then the artifact like a jar, or war file are deployed to a pre-production environment where additional testing is run as the final stage prior to production.
- Lastly, assuming that there are no issues found in the pre-production environment, the artifact is delivered to a production environment and is now live for customer interaction.

At each stage of the pipeline the processes and tools ensure that all checks are complete before it is moved to the next stage. For instance, if the build breaks in the developer's environment, then unit tests are not triggered. Likewise, if integration tests fail, then the code is not delivered to a pre-production environment. This model moves code rapidly through the system with each component tied to the next through automation. Although there are many steps in figure 4.1, some organizations can accomplish most or all of these steps through one platform and can do so rapidly.

NOTE. GitLab is a common platform that can perform most of these components allowing organizations to build, test, deploy, and monitor their application.

This process of code deployment can occur many times a week, a day, or an hour. With each release bringing new features to customers at a rapid pace.

4.2 DevOps isn't the only game in town

There are many different processes and models to release software. Similar to choosing the right security controls that align to the actual risk it is attempting to protect, the right release methodology needs to be in line with the organization's goals. If you are an organization that needs to release software multiple times a day like Facebook, Amazon, Netflix, or Google do then a DevOps model works well. If you are an organization that requires more strict control over your code releases like a healthcare, or critical infrastructure organization a Lean or Waterfall methodology may work better. The important takeaway with these different methods is that every organization approaches their release cycle differently. In fact, in many organizations, they will not follow just one approach across the organization. There may be some teams following a DevOps model while others are Lean. I will describe these methodologies next.

4.2.1 Waterfall

The Waterfall methodology is a methodology that depends on each part of the process acting as a checkpoint where formal signoff must be achieved before the next part of the process can begin. This, of course, leads to a long deployment process and can have the impact of the organization releases features that are no longer considered cutting edge.

In a typical Waterfall model, there are several stages. Much of this will sound familiar to what has been described in previous chapters. This is largely due to the fact that the many organizations built their application security programs while their engineering teams still worked in a Waterfall methodology.

- Requirements gathering where product requirements from the client or internally within the organization are defined and documented.
- The Analysis phase is where the scaffolding of the application is defined including the database schema and business rules that will govern the way the application works. For instance, that the application will require authorization from an administrator for certain critical functions.
- The Design phase begins to build the architecture of the application and makes key decisions on technology.
- Coding is where the real fun begins. Everything to this stage has been to clearly define what needs to be created while coding.
- The Testing phase is when code has been considered complete and begins to go through the process of testing which is likely to uncover defects that will need to be resolved before the final product reaches production.
- The Operations phase is where the operations team takes over an ensures the uptime and patching of the application as it's running.

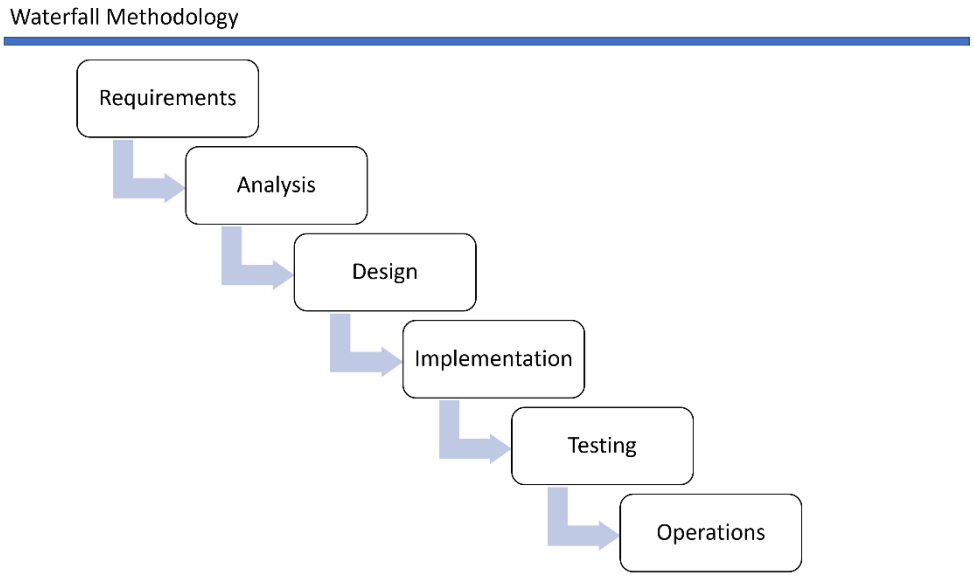


Figure 4.2 Waterfall methodology

Once these phases complete and the code is running in production, the process starts again with a new set of requirements. The release could have been a monthly, quarterly, or even a bi-annual release. These releases tend to be large with many features, hence the heavy process to ensure that everything is accounted for and production ready. However, in this process, there is typically a part of the development team that is needed to manage the incoming production defects. They process these defects and push fixes to production in something like a fix pack, but still with the proper oversight of a *change control* entity that ensures that changes are properly considered by an audience of stakeholders.

Security in Waterfall is handled primarily through two avenues. One is through the methodology itself during the requirements gathering, the design phase, as well as the testing phase where the security team or tools within the organization will impose security. Some of the requirements that are pressed may be part of regulatory, compliance, or contractual needs. New security technology will also be required by the security team if it provides better protection for the organization. The security tools that are either integrated with the development environment, or during the testing phase, will uncover different security issues that will be treated as defects where they will be triaged, assigned, and resolved.

The second way that security makes it into the Waterfall methodology is through the support path. If a vulnerability is found in production the support team will typically be alerted and required to triage and resolve it in a fix release. This is true for not only issues that may be reported by external parties, like clients, but also for issues that are discovered when components that are used in the application become vulnerable. This can happen when a weakness is discovered in something like the web server, or a parsing library that has a

newly discovered vulnerability that needs to be resolved quickly. Waiting for the next release cycle for a critical or high vulnerability to be resolved is unlikely to win over any fans.

Although Waterfall can create latency in the process of releasing software, the lines for injecting security in the process are pretty clear. Need to get security requirements in? Do that during the early phases. Found a vulnerability after the code is in production? Open a ticket with the support team. What happens when the process is kicked up another notch?

4.2.2 Agile

Agile is another methodology that allows for development teams to deliver software to clients. However, in this case the process is continuous and has the intent of getting software to the client often and quick. Agile methodology is typically associated with Scrum and Kanban frameworks where there are short iterations and clear work items outlined for a given timeframe.

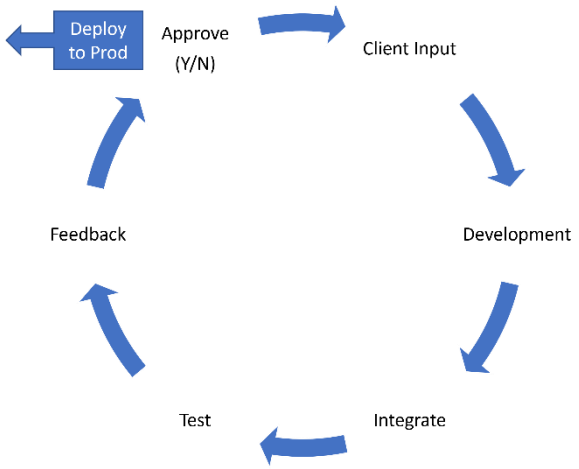


Figure 4.3 Agile process diagram

Agile methodology focuses on delivering software quickly through collaboration within the organization as well as with the client. Agile is also able to adapt to changes without impacting the delivery of software, hence the name. Unlike Waterfall that requires a more structured approach to intake of new requirements, Agile allows for changes to be integrated in a short period of time. If the development organization is using something like Scrum in their Agile methodology, the development team would work in sprints that are a short burst of work over two, three, or four weeks. The goal in a sprint is to complete some minimal viable product by the end of that sprint that can be delivered to a client. In most cases, a two-week time period for code release will not provide enough time to release a large change, but a small viable proof of concept on a feature is possible. This can then be delivered to an environment for clients to "touch and feel" and provide their feedback.

The development team that works in an Agile methodology is typically small, and tend to be cross functional, or full-stack, where the team has ownership of all components of the application development and deployment. This means that the team will own the user interface, the business logic development, the database schema and development, the configuration management, the deployment, and the operational aspects of the application. A team that strictly follows Agile will follow the twelve principles in the Manifesto of Agile Software Development.

- Customer satisfaction by early and continuous delivery of valuable software.
- Welcome changing requirements, even in late development.
- Deliver working software frequently (weeks rather than months)
- Close, daily cooperation between businesspeople and developers
- Projects are built around motivated individuals, who should be trusted
- Face-to-face conversation is the best form of communication (co-location)
- Working software is the primary measure of progress
- Sustainable development, able to maintain a constant pace
- Continuous attention to technical excellence and good design
- Simplicity—the art of maximizing the amount of work not done—is essential
- Best architectures, requirements, and designs emerge from self-organizing teams
- Regularly, the team reflects on how to become more effective, and adjusts accordingly

Agile focuses more on individuals contributing to the greater good, than relying on technology and processes to bring software to a reality. This leads to quicker, and sometimes more impactful, decisions being made at the lowest level within the develop team. The teams are empowered but also reap the consequences should something go wrong. For instance, a poor design decision made by one developer, could impact the rest of the team when it comes to support.

How does security integrate in the Agile methodology? Similar to the DevOps methodology, Agile allows for quick development and deployment of fixes for security vulnerabilities. These found vulnerabilities can be triaged by the application security team, presented to the Scrum team for prioritization, and worked in a future sprint. This means that, in theory, a fix for a security issue could be in production within a few weeks. Shorter if the issue is severe enough. Where there are some potential issues is getting security vulnerabilities prioritized and understood by the development team and the product owners. There will almost always be pushback especially when timeframes are short and client features need to go out. However, similar to Waterfall, the Agile team may have a separate Scrum team that is responsible for resolving defects found in production and potentially picking up security issues as well. When there is a separate work stream like this it allows the development to continue unimpeded while support and security work would be merged into the development branch or released as its own fix pack.

4.2.3 Lean

Lean has been in practice for over decades going back to the early automobile assembly lines in order to streamline processes, optimize people and resources, and more importantly

eliminate waste. When the Lean concept was introduced to the world of software development its purpose was to focus on reducing waste and maximizing value to deliver quality products quicker. It was not surprising to see development practices, thereafter, leveraging both Lean and Agile to optimize software development practices in what is commonly called Lean Agile. There are 5 key principles that lay the foundations of Lean.

- Defining value – This is what the customer is looking for and is willing to part with their money to get.
- Map the value stream – Identifying the tasks and components that make that value a reality. Those that do not add value are wasted motions.
- Creating flow – This is where impediments are removed, and the process is defined well enough to ensure that the value stream is smooth.
- Establish pull – This is where the work in progress is reduced while ensuring that just enough is being done for a smooth flow of work. The goal here is to eliminate context switching which is ineffective.
- Pursue perfection – This is reducing overall waste and always looking for process improvement by asking questions around the efficiency of the current process.

These key principles can be applied in various fields and practices including software development. It may not be surprising that many people adopt Lean practices even in their daily lives for non-technical practices.

Since the introduction of Lean Agile, the software development practice has been able to reduce rework and waste by replacing Waterfall development in many organizations. This would increase the ability to deliver marketable features quicker and continue to improve the development practice along the way. The Lean culture not only allows teams to continue to improve but creates a culture where discovering ways to become more valuable for the collective team becomes an acceptable practice.

Today most mature software development teams are practicing Lean Agile as an inherent byproduct of the two methodologies combined. Lean software development principles incorporate the original Lean and Agile principles by:

- Identifying and eliminating wasteful steps and practices.
- Build in quality through paired programming and test automation.
- Encourage knowledge sharing practices with other team members.
- Defer commitments and plans without having full knowledge of scope/plan/work.
- Deliver faster with targeted scope and features.
- Embracing and respecting people's opinions/input/feedback for continuous improvement.
- Optimizing the whole by seeing the bigger reward and not focus solely on one function or one practice.

In software development, reducing waste means focusing on the most needed requirements that are the most marketable and consumable for the users. In addition, leveraging Lean also supports the concept of having a *full stack team* where the team includes not only the core developers, testers and project managers but environments, network, and security resources as well. These specialized roles are also included to ensure delivery of the product at the highest value.

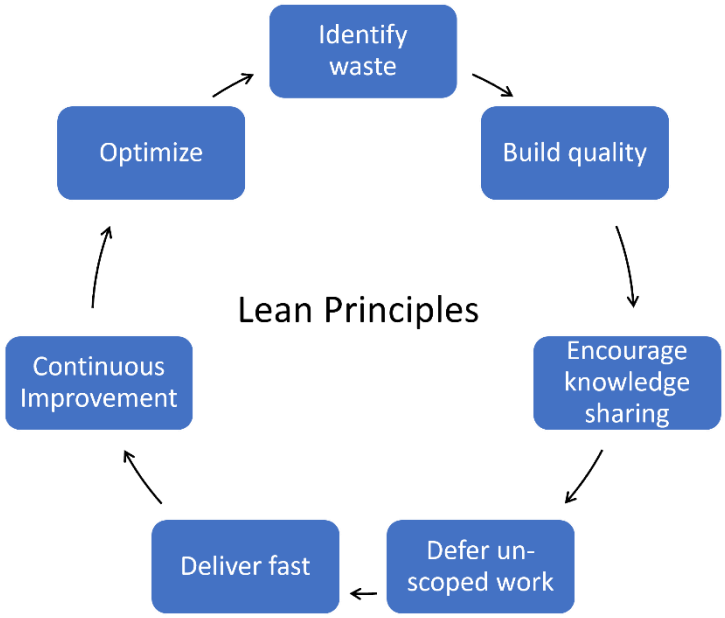


Figure 4.4 Lean process diagram

How does application security work inside of a Lean methodology? Security involves many different aspects including application, infrastructure, and business risk. Because these aspects are so vast, identifying and prioritizing the security risks has become more complex and manual. In the Lean world, this may amount to potential wastes and rework. Lean in application security should be used to unblock security processes and procedures and insert security into the software development life cycle. This is accomplished by aligning security practices into software development that will increase participation and reduce rework by incorporating security requirements and practices during requirements, analysis and design phases. In addition, Lean in security includes establishing security monitoring and continuous improvements as the product evolves and mature through the software development life cycle.

One important callout related to Lean in application security is that it works well with other software development methodologies such as DevOps and Agile considering that Lean is an approach that looks for continuous process improvement. As mentioned with other methodologies, the tools that are used elsewhere to provide security are still applicable here. Threat modeling, code reviews, test automation, monitoring are all still needed to provide assurance that security is baked into the product when it is delivered to the customer. The major difference is the focus on a "test and learn" approach that attempts to continuously review the security processes in each stage in order to understand where improvements should be made. An example of this would be in the case of threat modeling. When performing a threat model, regardless of the methodology, it's important to validate that the model is accurate. After a recently completed threat model for an application, it may be

found that through validation a control that was assumed to be present was not. In this case, the security team and the development team will review their threat modeling process and develop better process documentation as well as ensure that the appropriate stakeholders are included in the threat modeling process.

4.2.4 DevOps supports security better

DevSecOps, SecDevOps, DevOps with Security, it's been called many things. I've mostly heard it referred to as DevSecOps, so that is the term I will use here. Like the parent term of DevOps, the DevSecOps definition can depend on the process and technology.

> **DEFINITION** IBM defines DevSecOps as automatically baking in security at every phase of the software development lifecycle, enabling development of secure software at the speed of Agile and DevOps.

The methodologies I covered previously each provide a means for security to be integrated, however the DevOps methodology provides a unique opportunity for security to be more rapidly combined with other features in the development pipeline. As described, DevOps, and by extension DevSecOps, allows for development teams to have security fixes deployed to a production environment as quickly as the CI/CD pipeline can support.

In today's environments, where your applications are running, security issues are ever present. In early 2021, Redscan released a report that reviewed the trends in vulnerabilities that were added to the National Vulnerability Database (NVD) from 1989 to 2020. This report highlighted how much has changed over that time and specifically how 2020 was a banner year for new vulnerabilities. The report found the following:

- More security vulnerabilities were disclosed in 2020 (18,103) than in any other year to date – at an average rate of 50 CVEs per day
- 57% of vulnerabilities in 2020 were classified as being 'critical' or 'high severity' (10,342)
- There were more high and critical severity vulnerabilities in 2020 than the total number of all vulnerabilities recorded in 2010 (4,639 including low, medium, high, and critical)
- Nearly 4,000 vulnerabilities disclosed in 2020 can be described as 'worst of the worst' – meeting the worst criteria in all NVD filter categories
- Low complexity CVEs are on the rise, representing 63% of vulnerabilities disclosed in 2020

What does this tell us about what we are up against in application security? Vulnerabilities are being released at a rapid pace and they are becoming easier to exploit. This means that in order stay ahead of significant issues, application security needs to be able to move swiftly as well. With this in mind, the methodology that the engineering organization has decided to use matters. For instance, in the Waterfall methodology, waiting possibly weeks or even months to release a resolution to a found vulnerability would leave the organization exposed much longer than necessary. Even in an Agile organization, a fix might be weeks away. Pretty fast, but not fast enough for an attacker that might already

have code that exploits a CVE that was just publicly released. Attackers today can take information from a CVE and turn it into an exploit within hours.

This is where a methodology like DevSecOps can support the rapid deployment of security fixes. A well-tuned pipeline should allow, depending on the vulnerability, for a fix to a security issue to be deployed within hours assuming that the application security team has a well-defined process for triaging and assigning found vulnerabilities, and the DevSecOps team has the ability to pull in code changes, test, and deploy in a rapid manner.

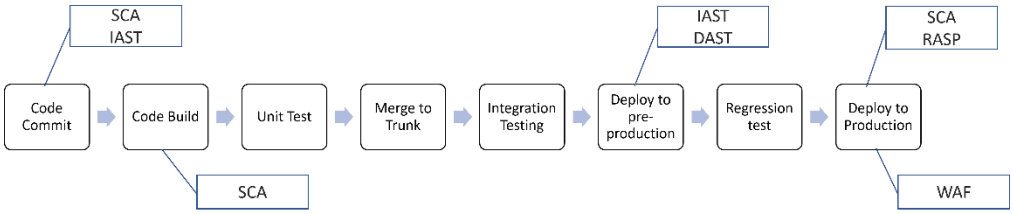


Figure 4.5 DevSecOps process diagram

There are several key practices to keep in mind when building security into a DevOps pipeline.

- Security issues are no different than any other software defects. In fact, they should be indistinguishable from software defects considering that vulnerabilities are essentially a software defect that has security implications. Consider a defect like a memory leak in a software program. This can be leveraged by an attacker to consume the resources on the server and make the application unavailable. However, even without the security considerations, this is still an application issue that impacts the performance.
- Automation of security should be a priority. Just like there are automated tasks within the CI/CD that run tests, perform deployment, and run checks. Security should be no different. In this case, the running of SAST, DAST, SCA, infrastructure code configuration checks, and the like should be made part of the CI/CD with feedback to the development team.
- Once the previous step is integrated in order to ensure that security is an automated part of the CI/CD, the findings in those tools need to be presented to the development team as early as possible and enabled to block the build from completing successfully and deploying vulnerable code. This last point is critical. Code should not be allowed to continue to a production environment with known vulnerabilities. Just like the failure of a regression test would stop the build from progressing.
- Once the application is deployed to a production environment, it is important to continue to monitor it for security related issues. As I mentioned previously, things like insecure third-party libraries are of a constant concern and can be introduced after the code is deployed to production. If everything goes well in the CI/CD and there were no found vulnerabilities with the developed code, that still doesn't mean that a third-party library that is used by the application would always be secure.

Additionally, as infrastructure is today being setup and configured as code, called Infrastructure as Code (IaC), there is a unique opportunity to apply a consistent approach to secure infrastructure, but in order to do this the organization needs to have a method of monitoring for configuration drift. This is where a single change is made in a production environment, often through manual means, which varies from the IaC templates that the organization normally uses.

Considering these practices when developing the pipeline will give the organization the better protection against the deluge of vulnerabilities. Practices aren't always enough. Having the right tools integrated at the right time matters as well.

4.2.5 DevSecOps Example

Taking the example of Superior Products, they have been using the DevOps methodology for some time and their development teams consist of developers, testers, operational people and, of course, Dashing Danielle. She has worked tirelessly over the past few months building in the following tools and processes:

- SAST (static application security testing) has been integrated into the developers IDE (integrated development environment) where the developer has the opportunity to scan their code for security issues early.
- SCA (software composition analysis) is integrated into the IDE as well as in each developer's browser as a plugin. The IDE integration allows for the developer to identify issues in the third-party libraries that they are using in the building of their software. The browser plugin allows the developer to research new libraries with confidence that there are no known security vulnerabilities impacting the library they are reviewing
- When the developer issues a pull request to submit code, they code repository tool requires the developer to submit the results of their SCA and SAST in the pull request.
- The Continuous Integration tools have SAST and SCA tools enabled that incrementally scan the code changed when the developers code is integrated with additional code changes from other developers.
- DAST (dynamic analysis security testing) is integrated into the testing environment to ensure that once the code is integrated and deployed to the testing environment, security test used to scan while the application is running are triggered. This also includes the use of automated penetration testing tools to augment the DAST scan.
- Dashing Danielle integrated the ticketing system that is used in Superior Products so that any tools that identifies a security issue will generate a defect, assign it to the appropriate development team, and alert them of a new finding on their collaboration channels. The defect is opened with recommendations from the scanning tools on how to resolve the found issue.

This setup allows Superior Products to rapidly identify security issues, get them to the right team, provide resources on how to resolve, a criticality and timeline for resolution based on the organizations resolution policy. This well-oiled machine is capable of finding and resolving security issues rapidly before the code is deployed to production.

What about Acme Services? They're not as up to speed as Superior Products and although they are using an Agile Methodology their releases are only deployed to production once every four weeks. They are still relying on a SAST tool that only runs once the code has been checked into their code repository and the continuous integration engine picks up the changes. Furthermore, the development team has decided not to trigger a failed build when vulnerabilities are found due to the timeline commitments to get code deployed at the end of the four-week sprint. This means that code is built with vulnerabilities identified and is then deployed to a testing environment with the known vulnerabilities. Acme Services has decided to partner with their internal penetration testing team to run a penetration test prior to the final build package being completed. This ultimately means that vulnerabilities are discovered late with the intention of queuing up the vulnerabilities that are found into the next sprint and subsequent release. This leaves them potentially exposed for weeks and shows the power of having tools integrated throughout the process.

4.3 Application security tooling in the pipeline

As I mentioned most of the tools that have been discussed previously in this book are still valid in a DevSecOps pipeline with subtle differences. There are also platforms and tools that are built specifically for the rapid release cycle of DevSecOps that can aid in the development and delivery of code in a secure manner.

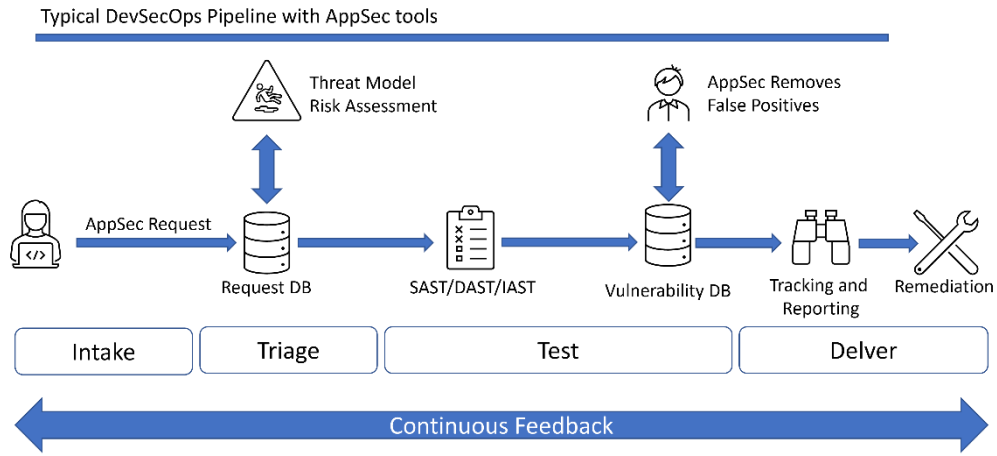


Figure 4.6 DevSecOps according to OWASP

In the Rugged DevOps pipeline your initial entry is through intake where there are still manual, and people led activities such as threat modeling and a secure design assessment. From here a well-tuned pipeline lets the tools do the work of identifying and coordinating the vulnerabilities that may have been uncovered. I'll cover these next.

4.3.1 Threat modeling in DevSecOps

I covered threat modeling in earlier chapters where I described the process of identifying threats that impact a given design or architecture with the goal of identifying threats that can impact an architecture as early as possible. One key difference with threat modeling in the DevSecOps methodology is the need to address operational threats early as well. This means finding threats that impact the code once it is deployed to a running environment. Some example threats that impact the operational environment are:

- Lateral movement in an operational environment where a compromise allows the attacker to pivot to another system.
- Changes to configuration after the application has been deployed that open the environment up to potential attacks. For instance, turning off security controls during a troubleshooting session.
- Improper segmentation between production and non-production environments that lead to the ability of an attacker to compromise a non-production environment in order to pivot to a production one.
- An attack on the any of the key resources for application security such as hardware security modules (HSMs), and secrets used by the application which potentially lead to an availability issue.

Threat modeling is still a valid input into the development of requirements in DevSecOps by simply asking "what can go wrong" and "what can we do about it?" One of the key benefits of having operational resources assist with the exercise of threat modeling is that they are in the unique position of seeing first-hand how the application can be abused. If you ever listen in on a conversation between a developer of and application and a person responsible for the deployment and operation of that application, you will hear two different voices on how the application actually works. The operational resource will have a much different perspective on how the application is actually used by end users. Chances are, they will see the application being used in creative ways that were not considered by the developer or even the application security team. This is the power of threat modeling in DevSecOps. Identifying potential flaws with those that understand how the product works, and can be broken, will improve the quality and security of the application when the design decisions are being made.

One of the hurdles with threat modeling in the DevSecOps methodology is that it does not react well to slow processes. As I described in previous chapters, threat modeling can be done through a manual process with a whiteboard and subject matter experts or through a tool such as Microsoft Threat Modeling Tool. Neither of these can feasibly be used in an environment where releases are happening multiple times a day. This means that threat modeling has to occur at a higher-level during design and architecture decisions when requirements can be outlined and integrated.

To complete a threat model in the DevSecOps methodology, the team will engage in the same threat modeling process as in any other methodology. This means that during the design phase, while requirements are being determined, the subject matter experts will gather to perform a more formal threat model that takes into consideration the design choices being made and the impacts to the application based on those choices. The threat

model should be well documented and included in the requirements tracking tool or the code repository tool that the development team is using so that it is available for review as well as updates as new alterations are made.

However, in the DevSecOps model, some critical decisions are being made at impromptu meetings in the team room or in virtual meetings. In this case, threat modeling is less of a formal activity and instead relies on security minded resources being able to think on their feet about the different issues that may impact the design choice that they are making. This requires security to be tightly integrated with the DevSecOps team. A successful approach to this is to have application security resources embedded in the actual DevSecOps team that can raise questions about the various threats that may impact a given decision. This security minded team should understand the following:

- What are the current threats to the application based on the design choices and architecture?
- How do these threats impact the application and are there known weak points?
- What current counter measures are in place, or need to be implemented to eliminate the threat or reduce the risk?

Even taking a moment to stop and ask these questions will help the team determine what the basic risk is on the decisions that they may be making during discussions that fall outside of the more formal threat modeling process. Getting to this point will require a culture change within the organization since it requires people rather than tools like the ones we'll cover next.

4.3.2 SAST in DevSecOps

Static application security testing (SAST) is not known for its blinding speed. In fact, as security tools go SAST typically gets a bad rap. I've done my fair share of complaining about SAST tools, their speed, and their abundance of false positives. I've seen others liken it to the shotgun approach. Not very precise, but effective if you are looking for results. This can become exacerbated if the organization has not taken the time to properly tune the SAST tool. This can be a recipe for disaster. It produces a lot of results, that then need to be triaged and processed. What's more, it adds a lot of time to the build process further upsetting the development team.

As I mentioned, DevSecOps only works well when there is actionable information that can be utilized by the development team in a timely manner. Although SAST tools can be noisy with potential for a lot of false positives, it's actually an extremely useful tool for detecting issues early in the development pipeline. Especially when the application security team and the products they support use SAST in their IDEs and tune the SAST tool per-application with specific invariant enforcement, anti-pattern detection, and specific issue detection.

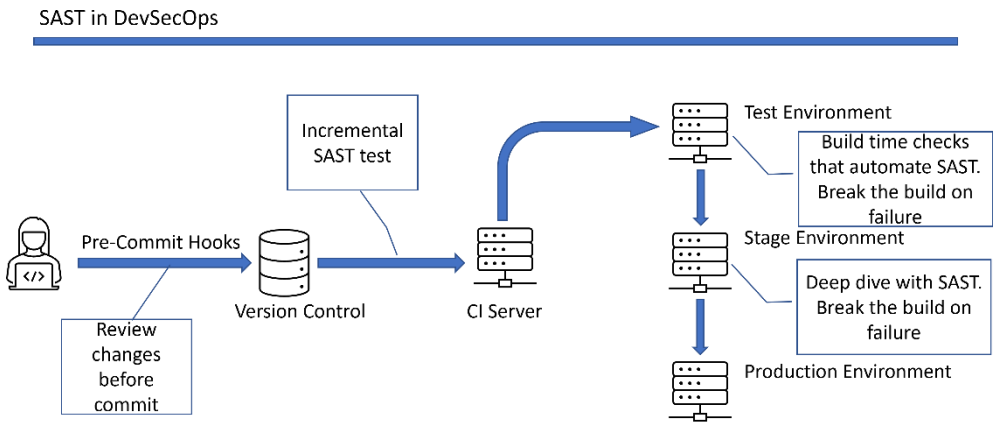


Figure 4.7 SAST in DevSecOps

As you can see, integrating SAST with the intention of getting feedback to the developer as soon as possible provides the ability for the development team to respond and resolve issues quickly. As an example, one of the developers at Superior Products is working on a new feature and is ready to check in their code. Prior to doing so, the developer runs the SAST tool that is integrated into their IDE and scans the code for potential vulnerabilities. During the scan it is discovered that the developer has a potential cross site scripting vulnerability that was coded since the code takes input from the user through a form but does not validate that the input does not include script. The integrated SAST tool provides recommendations on how to resolve the issue and the developer confirms with Dashing Danielle that the recommended remediation will resolve the issue. The developer codes the fix, rebuilds, and rescans with a clean output from the SAST tool.

Once the developer commits their code to the code branch, additional static analysis scans are performed where the entire application is taken into consideration as opposed to the targeted and incremental scan at the IDE level. The branch scan detects a possible buffer overflow that was introduced when the developers code was merged with another developer's code. The buffer overflow is reviewed by Dashing Danielle and the team, and a remediation plan is devised. With the code resolution in place, the local and branch builds both successfully complete with no further findings from the static analysis scans.

4.3.3 DAST and IAST in DevSecOps

Once the code has proceeded through the pipeline, from the developer's environment through the build process and is deployed to a test environment, it is ready for further security testing. One of the methods I spoke of previously is dynamic application security testing (DAST). This provides a look inward at the application and attempt to discover security issues as it runs in an environment. Likewise, interactive application security testing (IAST) can be used run inside the application and see attacks as they happen allowing the tool to observe and report on vulnerabilities. As you can assume, you may not want to run

DAST or IAST in a production environment as it has the potential of disrupting normal, legitimate traffic and create a production outage. However, there are some DevSecOps models that organizations use that allow for at least DAST to be run in both the pre-production and production environments.

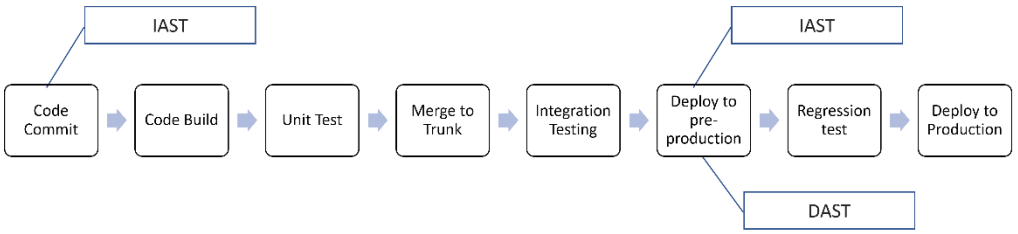


Figure 4.8 IAST and DAST in DevSecOps

Ideally DAST and IAST would be run in the same testing environments that are being used for other testing like regression and integration testing as these environments should already be setup and configured for testing. One of the benefits of this setup is that IAST tools need to see activity in order to report on a finding. In other words, to uncover a vulnerability in the code, the application is instrumented with an agent that continuously monitors the application for incoming attacks and reports on the findings. Having a suite of regression tests will help the IAST instrumentation detect potential attacks.

Fortunately, many IAST tools can be run even earlier in the process. Developers will have running application environments that are local on their machine or will at least have an environment where early testing can be done prior to pushing code to a branch. If the organization has decided to make IAST available to the developers in their local environment, the developer will be able run the chosen IAST tool in that initial environment to uncover issues prior to pushing code to a branch. If the organization has made the commitment to integrate IAST in the developer's environment, then the developer will be able to run unit tests that will intentionally execute workflows that should trigger alerts in the IAST tool. When something is found, the developer will be able to react quickly to resolve the issue.

In the pre-production environment is where more functionality has been integrated with the application and the IAST tool will be able to discover broader issues that can only be located once the code has been integrated in an environment that moves beyond the unit tests and begins to run integration and system tests. It will take additional time for the developer to resolve the issue since it would be found later in the process, but it is still preferable to finding an issue in the production environment.

Complimentary to IAST, DAST can be pointed to an environment and run as a security test against that environment. Although it can be used in the development environment, same as IAST, it is more effective as a point-and-shoot tool in pre-production environments where all the code and features are integrated. Ideally, the DAST tool will only test what has changed in an application, perhaps a new feature, UI, or API. This can be accomplished using

parameters for a targeted DAST. This type of incremental scan fits well into the DevSecOps model.

As I mentioned, DAST can be run in a production environment, but in most cases the risk of hindering production traffic is too high. Additionally, if the tools have been properly integrated in the lower environments and the developer's environment, then the need for production environment testing is less critical. I prefer to avoid running testing style tools in the production environments and instead focus on the run-time protection tools in production.

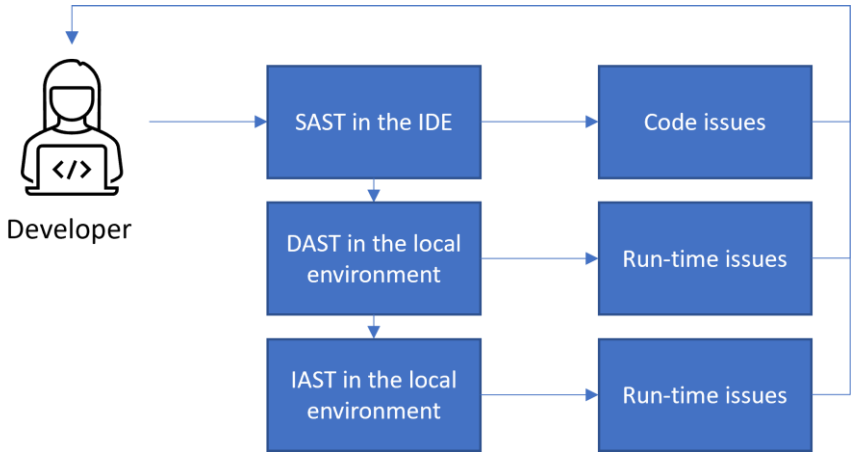


Figure 4.9 IAST and DAST in developer's environment

One additional point of DAST and IAST in the DevSecOps model is that it is necessary to have a feedback loop back into the developer's collaboration and defect tracking tool. When this is working well the findings in either of these tools will be able to alert the developer of a finding. This is more critical the further right in the pipeline that the issue is found, like the pre-production or even the production environment.

At Superior Products, the developers have the appropriate static analysis tools running in their environment as mentioned previously. To take their scanning tools and techniques a step further they integrate both DAST and IAST to ensure they are able to get additional security issues identified and resolved in a timely manner. Dashing Danielle has helped them devise a process to leverage the IAST tool in combination with unit tests, and other testing suites. This has been integrated into the developer's local environment which provides them the advantage of only seeing the vulnerabilities that impact the code changes that they made, before the code is checked into a code branch. The tool can identify issues during the running of the tests and point to specific lines of code so that the developer has the ability to make the needed changes.

Prior to code check-in, one of the developers at Superior Products leverages their IAST tool in combination with their unit test to determine whether any vulnerabilities are present with the new code that has been written. The IAST tool discovers that there are a few SQL

injection flaws in the new UI. After consulting with Dashing Danielle, it's discovered that the developer has not used the scrubbing function that has been built to take user input and detect whether there is a possibly injection attack being attempted. Once the developer adds this additional function to reject possible injection attacks, they rebuild the code and re-run the tests. This time the IAST tool determines that there has been no issue found and the team is satisfied with continuing to integrate the new code and deploying it to a lower test environment. Here the system and integration tests are run, and additional results can be found from the integrated IAST tool. During this additional testing, it is found that a XSS attack is possible in code from a developer that skipped the IAST scan. A defect is opened, the team is alerted on their collaboration tool and the developer is visited by Dashing Danielle who spends time outlining the purpose of the IAST tool and the benefits to the organization who is not only more secure but has saved potential costs related to detecting and resolving the vulnerability in production.

In this pre-production environment, Superior Products has integrated their DAST tool as well to uncover additional issues. The DAST tool will complement the IAST tool nicely where findings from IAST rely on indirect testing of the application to provoke a finding, the DAST tool will crawl the entire application and send attack patterns to weak areas of the site to test whether an attack would be successful. The results from the DAST tool will be less precise than the IAST tool leading the development team to work with Dashing Danielle to triage and discover duplicates from the IAST tool. In this instance, the DAST tool identified a potential buffer overflow issue when the DAST tool overwhelmed one of the inputs through fuzzing. The development team worked with Dashing Danielle to identify whether this issue was discovered in the IAST or other tools in the DevSecOps pipeline. However, once the duplication review and triaging of the issue is complete, the development team has determined that the finding is true and will be able to resolve it prior to it getting released to a production environment.

Tool	Pro's	Con's
SAST (Static Application Security Testing)	• Can find line number where issue exists • Applied early in the lifecycle • Can make remediation recommendations	• Tool may not support all languages in the organization. • Can produce many false positives. • Does not see the code while it's running.
DAST (Dynamic Application Security Testing)	• Not language or technology dependent • Not as many false positives as SAST • Can test software you don't own	• Can't always locate the impacted line of code • Can't find code specific issues • Found later in the SDLC
IAST (Interactive Application Security Testing)	• Agents or library that is continuously monitoring • Lower cases of false positives • Works well in DevSecOps	• More difficult to deploy and maintain. • Potential development work in order to integrate it • Can usually only see something as it's tested

As I previously mentioned, DAST can be run in a production environment, but this not recommended as it can have unforeseen impacts such as availability issues or corruption of data. There are options to have DAST run less destructive testing, but the organization needs to weigh the cost and benefit. Another alternative reason for running DAST in production would be for testing the security tooling and alerting for the security organization. Perhaps the organization would like to learn whether the security operations center (SOC) is able to detect incoming attacks. Otherwise, so long as the organization has confidence that its pre-production environment mirrors its production environment (and it really should) then running DAST in the pre-production environment should suffice.

> **NOTE** In reality, most organizations will only choose one lower environment to run these types of tools. License cost can quickly explode if you are being charged "per environment". It depends on the vendor and their license structure.

4.3.4 SCA in DevSecOps

As I covered in previous chapters, software composition analysis (SCA) is used to detect issues with libraries used to build the application that are from a source outside of the organization. These are typically libraries that are from third-party sources or within the frameworks used to build and run the application. An example would be log4j used by Apache as a logging framework. Frequently, these third-party libraries will be found to have vulnerabilities in their code that require patching. Nothing different than what any organization would do if they found a security issue within their own code that required a

patch. In order for an organization to resolve this issue that was found in the third-party library, they will need to package the latest, non-vulnerable version of the library in their own application. SCA tools are used to match the third-party libraries that are used in the application against a list of known vulnerable libraries that are publicly disclosed. Usually on a well-managed repository like the National Vulnerability Database. It should be no surprise that the earlier in the process that this can be completed, the better for the organization as a change in a library could mean a redesign of the application depending on the library.

What this means in the DevSecOps pipeline is that the developer needs access to information regarding the safety of the libraries that they are considering and packaging within their application. This can be accomplished in a few ways. Most vendors today have IDE plugins or even browser plugins that will let the developer know that the library they are reviewing or about to leverage in their application has a known vulnerability associated with it. Keep in mind that with SCA, the library must have a known weakness, usually in the form of a published CVE that is reported in the National Vulnerability Database (NVD). Some SCA vendors will have their own process for detecting weak libraries, but all of them will leverage a public repository like the NVD to locate known CVE's that are associated with the library.

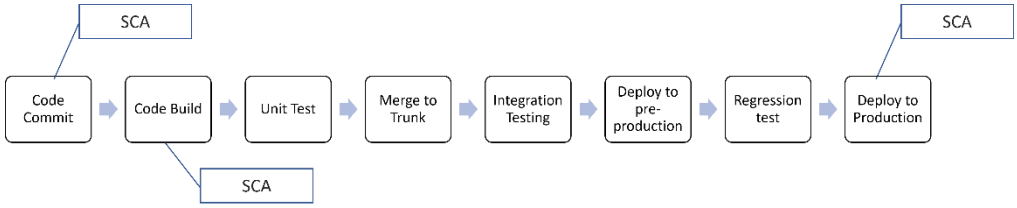


Figure 4.10 SCA in DevSecOps

With this early access to CVE information, the developer can make the decision to use a non-vulnerable version of the software prior to even writing the first line of code that integrates that library into the application. However, new CVEs are being released daily. This constant flood means that the day after a developer has chosen a secure library to use, it could become vulnerable to a newly discovered issue. Or worse, this can and usually does happen when the library is already packaged and deployed to an environment. This is where the rest of the DevSecOps pipeline needs to pick up the burden. As with the other tools in the DevSecOps pipeline, the earlier in the process, the better. Once the code has been committed to a branch the integration server that performs the build and test tasks needs to perform the task of calling the integrated SCA tool, usually through an API, to determine if any of the libraries in the software bill of material (SBOM) are considered vulnerable. During this build process, any finding can be resolved by breaking the build and requiring a change to the library.

This sounds great, but you're probably already thinking about the implications of finding a vulnerability at this stage in the development pipeline. If you checked in the code and you have a library that has been identified as vulnerable, it will be difficult to package and test a new library in a short period of time. Additionally, there may not be an immediate path to a secure library. In other words, there may be a library with no secure version released yet.

This is where a robust risk management process needs to be in place. Knowing what the application risk appetite is and the organizations overall risk appetite means that the team can prioritize a resolution that matches their risk. For instance, consider a found vulnerability in a library that is used by one of the applications at Superior Products. The library is used to provide a graphing function in the UI to display charts. The particular vulnerability is related to how the graph is rendered. After the build is broken due to this finding by the SCA tool, Dashing Danielle reviews it and recognizes that this particular rendering function is not actually being used in the application, as they are using the library for a different purpose. With this in mind the risk is identified as being a low risk and the build is allowed to proceed after the application security provides what's called a waiver, meaning that the build can proceed for a set period of time. When the waiver expires the build will be broken again but this buys the development team time to provide a secure library once it is available.

4.3.5 Run-time protection in DevSecOps

Run-time application self-protection (RASP) and web application firewalls (WAF) play an important part in the protection of the application once it is running in a production environment. Throughout the DevSecOps pipeline, the goal is minimizing the risk to the organization as code is deployed at a rapid pace. When the organization either cannot catch an issue before it is released to production, or an issue is found after the code has been released, the organization then needs to rely on run-time protection to limit the risk.

As mentioned, the best approach in the DevSecOps pipeline is to perform incremental scans using the security tools as early in the process as possible in order to locate and resolve issues as close to when the code is being developed as possible. As the code progresses, the security scanning becomes more aggressive to locate any issues that require more overall application visibility.

However, at some point the code needs to make its way to a production environment where it will face the test of real-world attacks. This is where protection mechanisms are required to provide defenses against attacks that were not found earlier in the development lifecycle. Not all of these attacks could even be found during the earlier stages of the lifecycle. Novel means of attack are constantly being developed by attackers which means that the scanning of the application is simply not enough. A prime example is the fact that many attackers will leverage multiple vulnerabilities chained together to compromise a system. Meaning that it's more than just a single critical, but instead it could be a few lows that turn out to be the culprit in an organization's compromise.

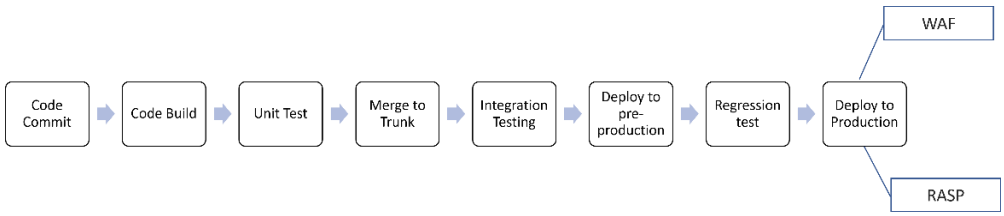


Figure 4.11 Run-time protection in DevSecOps

One of the key features of run-time protection tools like a WAF is that many are offered as software as a service (SaaS). This means that the management of the tool is transferred to a third-party who hosts the software. The organization can also choose to host their own WAF within their network, but there is an increased cost of managing the internal WAF. This SaaS based WAF has its benefits and drawbacks. When a new attack method or vulnerability is identified, the third-party is generally responsible for delivering a ruleset in the WAF to protect the application going forward. If the WAF is internally maintained, then the security or the operations team will be responsible for delivering the new ruleset from the vendor. One way is no better or worse than the other in the DevSecOps model, it's just important to know that the management and the timing of when the rulesets are delivered to the WAF need to be considered.

With RASP, by its nature, it is integrated with the application and runs alongside of it. This means that there is no SaaS offering and the DevSecOps team is required to ensure that it is functioning and up to date. New rulesets may come in from the vendor of the RASP tool, but similar to the WAF, the team will need to ensure that the rulesets are deployed and do not create a performance or availability issue. Additionally, as new attacks are discovered, it is up to the organization to integrate those signatures into the RASP or WAF. For example, the organization may see a specific type of attack that is not related to the generic rules that are applied by the vendor of either a RASP or WAF. This specific attack might only impact the organization or the industry they are in, as in the case of a coordinated attack against something like a sports betting application that shares a common technology stack with other sprots betting applications. In this case, if there is a coordinated attack against sports betting with a specific exploit that is only being leveraged against this industry, then the organizations will want the ability to deploy custom rules that can block these incoming attacks quickly. This assumes that the vendor does not have a generic rule to block the exploit.

For both WAF and RASP, one of the key considerations is the testing of any new rulesets that are deployed. As I mentioned previously, most organizations will not use multiple licenses in their protection tools to test multiple lower environments. Especially for run-time protection tools, you are only getting the benefit from them when you are running them in an environment that has the potential to see and block active attacks. These lower environments are not likely to see external attacks unless they are open to internet to allow customers to test in them, or they are used in penetration testing. One recommendation is to have these runtime tools in a pre-production environment where broad testing is done to ensure that a new ruleset that is enabled in the tools does not create a regression issue. The most common issues are where the new rules block legitimate traffic or otherwise case the application to no longer function as expected.

With these tools deployed and providing protection for the application, the DevSecOps team will have more confidence in delivering new features to production

4.3.6 Security orchestration

Application security orchestration and correlation (ASOC) was introduced as a concept in 2019 by Gartner.

DEFINITION Gartner defines ASOC tools as those that "streamline software vulnerability testing and remediation by automating workflows. They automate security testing by ingesting data from multiple sources (static, dynamic, and interactive [SAST/ DAST/IAST]; software composition analysis [SCA]; vulnerability assessments; and others) into a database. ASOC tools correlate and analyze findings to centralize and prioritize remediation efforts. They act as a management layer between application development and security testing tools."

ASOC is a combination of two different tools that both assist in the DevSecOps pipeline. One is the application vulnerability correlation (AVC) and the other is the application security testing orchestrations (ASTO). AVC tools ingest vulnerability information from multiple sources so that the vulnerabilities can be de-duplicated automatically therefore reducing the amount of time and effort that the team has to spend in doing this work manually. ASTO tools can orchestrate and automate multiple commercial security tools in the DevOps pipeline in order to ensure that security testing is not only happening but that it is integrated with the continuous integration platform that the organization is using.

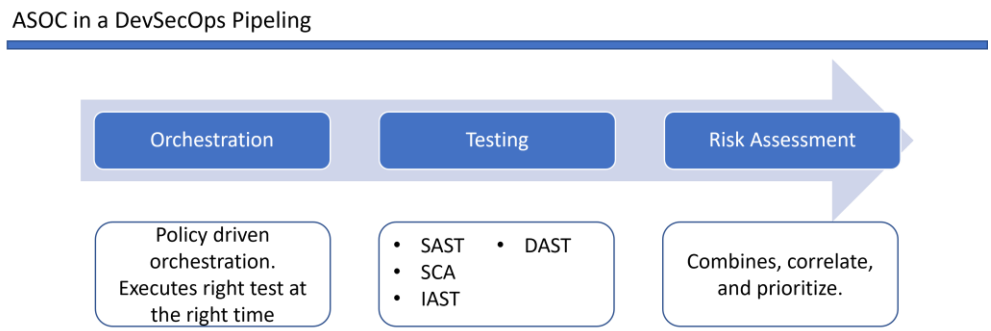


Figure 4.12 ASOC in the DevSecOps Pipeline

ASOC tools are becoming more prevalent as vendors are recognizing the need to provide this capability to the organizations that are looking to integrate the following capabilities in the DevSecOps pipeline:

- Correlation and analysis of vulnerability information multiple application security testing tools like SAST, DAST, IAST and others.
- Integration with defect-tracking tools that are used by the development organization to address defects and vulnerabilities.
- Integration with CI/CD tools and platforms used by the development organization to build and deploy their applications.
- Rapid speed and accuracy of results to reduce the noise. Ideally, the vendor will leverage some level of artificial intelligence or machine learning to help reduce false positives and provide more actionable results.
- Reporting that can be used to measure an organizations success with reducing new vulnerabilities.

ASOC can simplify the prioritization process by discovering whether the vulnerability is applicable and then assigning a criticality to it so that it can be prioritized. If the ASOC tool is using machine learning (ML) it may look to see how past responses to vulnerabilities may influence future behavior. Additionally, when the ASOC is integrated with the defect tracking tool used by the development team, they can be sure that the results they receive are actionable. This also allows the application security team to see vulnerabilities across the organization and build reports that are valuable to leadership. As described an ASOC fits well in the DevSecOps methodology given it ability to automate, deliver quality results, and provide a single pane of glass across the organization.

4.3.7 Security education

Security education is near and dear to my heart. It's the underdog when it comes to security process, tools, and technology. Why is this important for DevSecOps? Simple. Having security minded developers and operational people when decisions are being made rapidly reduces the burden on the application security team and helps to deliver more secure code from the start.

When most people hear security education they think about the video or slide material with a quiz at the end. This is still widely used across probably all organizations at some level. Although there is a time and place for this type of training, it's not effective for raising the security IQ of the organization in a sustained manner. This only occurs when the organization has invested in a training platform that can deliver training on-demand, or even better, at the time when an issue is discovered. Some technology solutions can integrate with defect tracking tools in order to provide a link to application security training modules for the specific vulnerability that was found. For instance, perhaps a SQL injection vulnerability was found in a penetration test. When the application security team opens the vulnerability in the defect tracking tool the application security training module can detect that the ticket was opened and add a link to the training platform so that the developer has access to information on how to resolve the found SQL injection.

One of the key conditions of training is it must be quick and to the point. Most of us get pretty aggravated when we have to sit through not just boring, but also long, training. The most effective training is timely, brief, and often if necessary. As we've been talking about DevSecOps it's important to point out that many SAST tools have IDE plugins that allow for coordination of vulnerability discovery with some quick hit training module. This allows the developer to immediately see why the line of code may lead to a vulnerability. These don't need to be long either otherwise what keeps the developer from simply going to the internet for help? These quick hit modules should be just a few minutes long and in the language of the developer. No corny videos please.

To augment this quick hit training, it's important to ensure that more traditional training is still available. However, not all security education needs to be death by PowerPoint. Many of training platforms today are engaging and approachable by a potentially hostile audience. However, the organization has other options for delivering security training. Some application security training platforms offer the option to host tournaments that will provide a training ground for a large group of developers where they can test their skills in something like a hack-a-thon. Other platforms offer gamified training that intends on providing training where

the learner has the ability to test their skills right there and then through training that provides real world, hands on, scenarios.

Some large organizations will host internal developer conferences that include a security component. This provides a platform for the security organization to showcase some of the new technology and processes that are working around the organization. This is even more effective when development teams bring their experiences with security and show how it has better enabled them while making them more secure. It's worth mentioning that the organization doesn't have to draw on their internal resources exclusively. Brining in security speakers from the outside is also helpful in showing with some of the organization's peers are doing. This is a great example of where you can generally get free, or near free, security awareness.

The biggest payoff with developer training is that it reduces the burden on the rest of the organization and application security team. Anyone that has run an application security function will tell you that there is no way your team can scale to cover all the various applications and areas of concern. The team must rely on the organization to build security in when they are developing code and the tools that are layered into the development lifecycle can only do so much. Secure code starts with security minded developers.

4.4 Feedback Loop

One of the unique features of DevSecOps is that it is a constantly moving pipeline. This means that in a mature DevSecOps pipeline, there will always be code moving through the various parts of the assembly line until it is deployed to production. The security gates are gone from the process. So how do we make sure that developers are getting the feedback, the code is secure, and the organization is safe from increased risk?

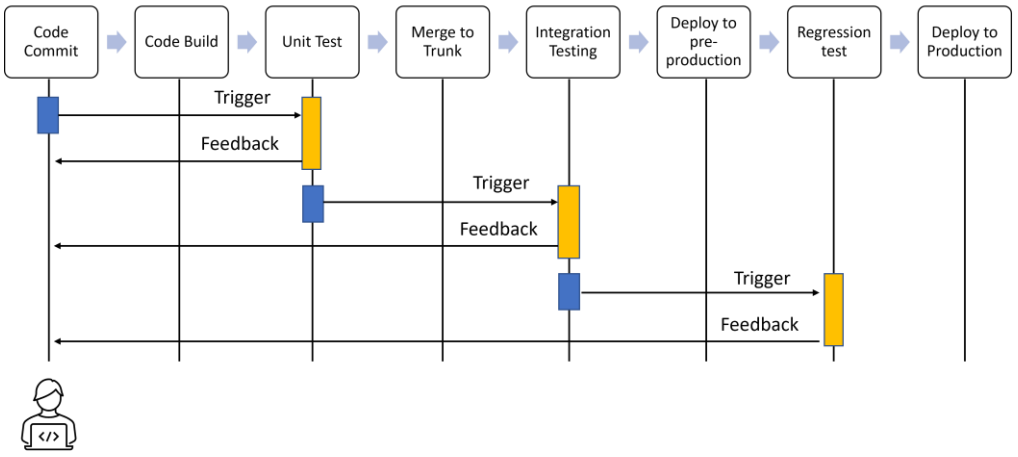


Figure 4.13 DevSecOps feedback loop

To do this, the pipeline needs to have hooks throughout that provide constant feedback to the developer, regardless of where in the process the issue is found. The SAST tool running during code check-in may append the output of that analysis to the code check-in. The integration server will run tasks that execute security scans such as IAST and DAST once the code has been deployed to a workable environment. The output from those tools may trigger a message in the development team's collaboration tool as well as automatically open a ticket in the defect tracking tool. It's also possible that the organization has a policy to block a build on vulnerabilities of a certain type or of a certain criticality level, like a criticality of High or above or if it's a SQL Injection vulnerability. In which case the build would break, and the developer would be notified through email or their collaboration tools that the build has been broken due to a change that was made during their check-in.

This type of feedback loop allows for the pipeline to continue moving while performing an event-driven approach to the security feedback loop. This means that it is occurring asynchronously and does not create the more state-machine style pipeline where each stage is waiting on the next to complete cleanly before proceeding. This still needs to rely on a "go/no-go" decision before deploying to production. The organization may take the position that they will allow certain vulnerabilities of a particular level into production as long as there is a path to resolution in a short period of time. Say a few days. However, the more mature organization will not allow any discovered vulnerabilities to be delivered to a production environment and will require a clean production build in order for deployment to occur.

As you can see, the DevSecOps pipeline serves the same purpose for two different work streams. On the one hand, being able to deliver value to the customer in a matter of hours is immensely desirable. Likewise, every security person should want that same capability to deliver security resolutions to the organization in a rapid manner. It's a win-win.

4.5 Summary

- DevSecOps plays a critical role in enabling the development team to deliver software with confidence that security is built into their pipeline.
- There are other development methodologies that can still deliver security but struggle to do so at the speed that is required in the modern development lifecycle.
- DevSecOps breaks down the segmentation between the security team and the development team and integrates security throughout the pipeline.
- The common tools used in application security still apply (threat modeling, SAST, DAST, IAST, RASP, and WAF). In fact, they are able to be applied earlier with a faster feedback loop to the development team.
- Feedback loops are critical to the success of resolving found vulnerabilities by getting the information to the development team where they collaborate the most.

5

Security belongs to everyone

This chapter covers:

- Expanding application security through standards, requirements, and reference architecture
- Building a culture of security that includes education
- The maturity models that can be used in an application security program
- Decentralized AppSec in software development

Stop me if you heard this before, but security is everyone's problem. We've all heard this many times, but what does it really mean? In my experience, the ability to scale an application security team to meet the need of a large organization is difficult if not impossible. Many of the organizations that I have worked with have had hundreds or even thousands of developers. In these organizations, even what I would consider a large application security team was no match for the sheer volume of work in the organization. This means that organizations must find other, more creative ways to bring security to the overall development of software.

As I mentioned in previous chapters, the best defense against attackers is to ensure that all members of the organization are able to understand the security risks that the organization faces. This does not mean that everyone needs to be a security specialist but having at least basic security knowledge goes a long way. Nothing makes me happier than someone reaching out to me asking about the security impacts to a design choice. This means that they are at least thinking about security and how their choices can affect it. And I'll admit, I don't always have the answers, but being able to work through an issue with someone else teaches both of us. I can't imagine that I'm the only security person who feels that way. Not because we're actually wanted, but because when more people think this way, the security folks have the right advocates in the organization.

Getting to this little bit of heaven takes an organization that has put the effort into building a culture that not only takes security seriously but thinks about it as much as the security teams do.

5.1 Security is everyone's problem

I try to draw parallels when I am thinking about security, and one that comes to mind here is your common household. One person in the house is not generally the security person who goes around the house everyday making sure that the doors and windows are locked, that the alarm is on, the stove is off, that the cars are locked, and that everything in the house is secure. I suppose that might happen in some households, but the point is that this is not an efficient way to approach the security of the house. In reality, it is the responsibility of everyone in the house to ensure that they are considering the security and safety of the house and those inside. If the smoke detector is out of batteries and begins to beep, yes maybe the tallest person in the house has to change the battery, but others in the house can hear it beeping and can let the tall person know so they can change the battery. This is a silly example, but the point should be clear.

In an organization, the security team should not be the only ones aware of or looking out for security issues. Yes, they are the ones that typically manage the tools and processes that manage the security risk of the organization, but they don't always have 100% visibility into all the areas of the organization. More importantly, organizations are releasing new features, and functionality all the time. It is unlikely that the application security team has insight into those new features until they are about to be released. That is often too late for the application security team to react, and it becomes more of an exercise in reducing or accepting the identified risk than building security into the design early in the process.

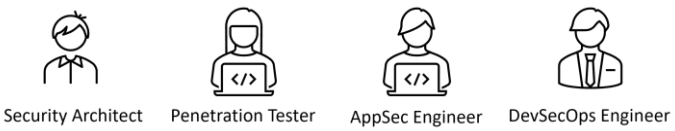
When the organization takes the steps to build a security culture, they will be able to augment the application security team and create more secure code. However, this requires an application security team that has built the frameworks, processes, tools, and education that is needed to enable the development organization to achieve that level of security.

5.1.1 Structure of an application security team

It is extremely difficult to hire enough resources in the application security team that will fit all the needs of the organization. Although every application security team is structured differently and has different needs, most application security teams provide the following core functions:

- Threat modeling and risk assessment
- Penetration testing
- Implementing scanning tools
- Triaging discovered vulnerabilities
- Code and architecture review
- DevSecOps and automation support
- Secure education and evangelism
- Documentation, whitepapers, standards, and requirements development

On a daily basis the application security team members perform these tasks as their core function, but there is other ad hoc work that happens such as general consulting on design and strategy across the organization. It is often hard to find the "unicorn" application security person that can do all or even multiple items from the core functions. This requires the application security team to hire multiple people with more focused skill in the core areas. For instance, in order to meet the needs of the functions the team may hire the following roles:



> **NOTE** This is a very unpopular opinion, but one that a lot of my application security peers share: Penetration testers are not application security engineers. Many organizations work hard to build a team that can perform penetration tests against applications. Don't get me wrong, this is needed. However, this is not the primary function of a mature application security team.

The organization will also have to sprinkle in the obligatory support around the team like leadership and project management. As you can see, this team can focus on delivering security to the organization through the tools, processes, and people. However, scaling the team to meet the organization's needs is next to impossible. The recommended size of the application security team varies and has been proposed as 10:1 of software engineer to application security engineer. In large organizations, this ratio is difficult to meet and depends on how seriously it takes building security into their applications. However, in most cases, the application security team is roughly 1% of the development headcount. Again, this is not a recommendation, but rather what is typically seen in the industry. I have been a part of teams that are oversized and undersized based on that measurement. This is why creating a culture that supports the security of the applications that are being developed is critical to reducing the overall risk of organization.

> **EXERCISE** Take a look at your own organization's application security team and understand what functions they perform. If your organization doesn't have one, research some popular organizations and their application security team structure. Each organization I have worked for has had a different set of goals and staffing model to support application security. You will never find the same model in every organization.

5.1.2 Just hire more application security people

In most engineering organizations, when you have a new project or push to meet a deadline, the organization will just higher more engineers, testers, architects, and others to meet the need. This is not only impractical for application security but is also often cost-prohibitive. Security teams in an organization are often working on a smaller budget compared with the overall IT spend. Although the spend varies by organization and industry—and there was a significant increase in security spend during the global pandemic in 2020—most

organizations spend 5-15% of their IT budget on security. The bulk of this spend goes to security operations and perimeter defense tools like a security information and event management (SIEM), network detection and protection, firewalls, and others. The application security team typically gets a small portion of that budget that is mostly dedicated to personnel.

Despite spending most of their budget on personnel the application security team can never scale to the size needed. However, this is not the biggest problem facing the application security team. It's actually very difficult to hire the people you need. There has been a term that has been used a lot in the industry called "negative unemployment." This means that there are more jobs open than qualified people to fill them. To be clear, this is not just application security, but across the board in security like analyst, network security, information security specialist and others. This means there are millions of jobs left unfilled in security. Those of you reading this book that work in security know exactly what I'm talking about.

The reality is that application security resources are even more difficult to locate than other personnel in their peer groups. Generally, you are looking for someone that has not only security expertise but also development experience. Your best bet is to find someone that has been a developer and gained an interest in security. The importance of finding someone that has development experience cannot be overstated. Someone with this background brings the following to the team when they join:

- Understands the development pipeline
- Can interpret code and perform code reviews
- Understands the overall development process
- Can speak the language of the development team

Having these qualities as well as an understanding of security will be an asset to any application security team. This allows them to take vulnerabilities that impact an application and be able to interpret this issue in a way that is understood by the development team. For instance, a prototype pollution issue may have been discovered through a DAST tool and assigned to be triaged by the application security team. These issues can be complex to describe and walk through with the engineering team if the application security member cannot describe how prototypes works in JavaScript. Having a background in development means that the application security person can at least describe the issue, how it impacts the application and help develop mitigations. More importantly, when the development team makes the claim that the vulnerability does not impact them, which they will do, then an application security person who knows development will be able to determine whether the impact is real.

You can see how these types of hires can be hard to find. With a constrained budget, and a small pool of resources to hire from, the application security team will struggle to find the right people that can deliver security to a development organization.

Organizations can also make it difficult on themselves when they don't have a good job description or even job title for what they are looking for. An application security architect is different than a solution architect. A security engineer is different than a security analyst. Even worse, some organizations will have their own internal titles that may map to industry

standard job titles, but only muddy the waters. This means that well-qualified individuals may pass on a role due to a misunderstanding of the title.

Definition

Each organization has a different description for common roles. Here are a few of the common application security roles and a brief description:

- Application Security Architect – Represent the application security team on strategy in the organization and develop secure architecture, standards, and documentation.
- Application Security Engineer – Work closely with the application development team to ensure that security is integrated in the development process for the development team.
- DevSecOps Engineer – Take the responsibility of the application security engineer with the added responsibility of automating the processes in the development pipeline.
- Penetration Tester – Provide testing of applications for security related issues and develop summary reports on findings.

One last point I will make regarding hiring in application security. Sometimes the right person for the team has no clue about security. The application security team is usually operating parallel to the engineering teams they work with, which means that having members of the team that understand the engineering process is as important and, depending on the role, sometimes more important than understanding security. The team will need people that can integrate tools, develop and understand code, speak intelligently about the software development process, and operate and tune tools. To achieve this, the team does not always need to hire the best and brightest security people as these gaps need to be filled with a team that is diverse in thought and background.

5.1.3 How to close the gap

So, what does this mean for an organization that wants to raise their security posture without having to hire a massive army of penetration testers, security engineers, and security architects? When you can't afford a car, but you need to drive to the store, find a friend that has one and borrow theirs.

> *'Look for the helpers. You will always find people who are helping.'*
>
> **Fred Rogers**

Getting anything done in application security usually means that you will be borrowing time from the resources in engineering or at least working with the product and engineering teams to jostle for space in their releases. In most cases you will want to make this as transparent and open as possible. Nothing drives a manager crazy, me included, more than their team getting pulled into tasks that are unrelated to the work that they should be doing. However, security is everyone's responsibility, right? It's also the responsibility of the

application security team to make sure that the engineering team understands what needs to be done and provides the most amount of support they can.

One approach to making this work is by engaging with the engineering leadership on a regular basis to ensure that priorities are aligned. This isn't just application security priorities, but also the priorities of the product. You will not advance security if you are bringing only your problems and needs to the table. Listening to the product team and the engineering team about their concerns with security and their pain points when it comes to balancing security with feature releases will go a long way. To effectively spread security in the engineering organization you need to build a relationship that is developed with mutual trust and support in mind.

5.2 Security education

I admit that I believe in the power of training and education. I have taught undergraduate and graduate computer science students and have led training programs at organizations on application security. Training is a cheap and effective way to raise the security IQ of the development organization. But this goes beyond the annual training and handful of courses throughout the year that is usually obligatory and ineffective. In fact, although the vast majority of organizations have your standard security education in the form of phishing and social engineering training, most organizations do not focus on engineering security training. This is a huge gap.

There are a multitude of approaches to bringing security education to the engineering teams. Each approach has its benefits and are delivered in different manners in order to be most effective. On-line training, just in time training, and microlearning can be delivered pretty quickly and efficiently to the engineer to address an immediate need. Alternatively, conferences, lunch and learn, and in-person training can be long but much more in depth.

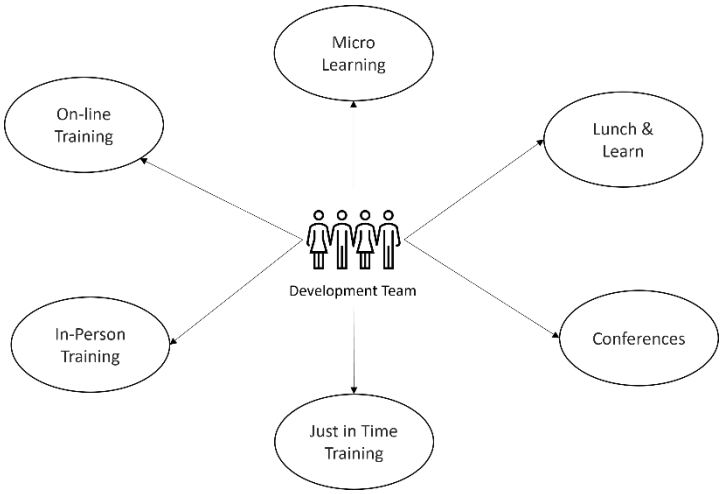


Figure 5.1 Security training options for a development team

With a well-rounded approach to security education, the organization begins to build the ecosystem that supports a more security minded engineer. This is a cost effective, and scalable approach to reducing the burden on an often-small application security team and ultimately raising the awareness of security in those that are responsible for writing code.

5.2.1 Raising the security IQ

Creating a culture of secure engineering is a multipronged effort, but it starts with effective training. Some examples of training that can be done on a periodic basis in the organization include the following:

Self-paced training online	There are a lot of options for online training platforms. Some are more general training platforms that offer courses in various disciplines. Others are specific for security like awareness or secure engineering.
Instructor led training	Depending on the organization, there may be a learning and development team that can run training. However, much of this will be training that is specific to the organization, and most of the times will not be security related. If the organization is looking specifically for security training, they will need to leverage a third-party company that can come on-site to perform the training.
Application Security led training	The Application Security team can also provide training to the engineering organization both online and in person. One of the benefits of this is that the Application Security team can focus on known issues within the organization to focus on. However, this requires the Application Security team to maintain the training and keep it relevant.

Each of these are effective means of delivering security education to the engineering organization. There are varying costs in terms of budget as well as time. For example, online training through a third-party platform can be expensive, especially if you are paying per user. But you're able to quickly roll it out to a large audience simply by assigning it to engineers. In-person training led by a third-party can be expensive per person and doesn't scale well. When you compare it to the costs of an online platform, in-person training is highly focused and the opportunity to ask questions and follow up with the instructor adds a benefit that is often not there for online training. Many organizations will take the approach of having training where the targeted in-person training is done infrequently and specific to a core group, and the online training is open and available to the wider organization to take advantage of on-demand.

NOTE Many training platforms offer integration into the development environment that will allow for the developer to take a short training module that explains the issue with assistance on how to resolve it. Often these can be integrated into the analysis tools that the organization is using, specifically with SAST tools that are part of the IDE.

One additional method of in-person training that can be effective, and more cost-effective is when it is led by the application security team. Organizations that take this approach will designate, or even certify, several members of the application security team to be trainers. This allows the organization to hold several training sessions per year at a reduced cost. There is also the added benefit that the application security team can provide specific use cases that apply to the organization and also present recommendations that align with the organization's requirements, goals, and standards.

The application security team has a unique opportunity to augment the training approach of the organization. This is the team that is seeing the daily activity across the organization and knows where the organization is most vulnerable. This gives the application security team the ability to take the information they have to formulate more specific training for the engineering organization. Take for example if the organization is seeing SQL injection vulnerabilities consistently appearing in scans and penetration testing. The Application Security team can provide training for the engineering organization that focuses on SQL injection. Taking it a step further, if the vulnerabilities are frequently coming from one development team, they can provide specific training just for that team. This doesn't have to be a significant investment of effort. An hour or two with the development team led by one of the application security team members would suffice. The benefit of working with a team that has a history of releasing the same vulnerabilities is that the application security team can provide examples of the issue and remediation using the code base that the engineering team is working in.

5.2.2 Microlearning & just-in-time training

Microlearning is by no means a new concept but it has been taking off in the past few years. The goal is to break down larger training into quick bite-sized training that is easily consumed by the intended audience. Although there is no real measuring stick on time and size of the microlearning it is commonly less than five minutes long. This allows the learner to understand a distilled concept and take away the key points in a short period of time.

This type of learning works well in the DevSecOps model that I covered previously. The fundamental goal of microlearning is to provide quick learning while the learner is performing their job. A great example would be if a developer has coded a SQL injection vulnerability identified in through their SAST tool that is integrated into their IDE. In this case they can take a quick microlearning module that describes the impact and remediation of SQL injection vulnerabilities. This provides enough information to the developer to understand the issue and provide a remediation that resolves the vulnerability.

Microlearning doesn't have to be video based, it can be delivered in other mediums like blogs, e-learning, games, podcasts, and infographics. How this works in your organization really depends on the culture and the learning habits of the engineering teams. It also depends on the content that is being taught. Something that is complex in nature may

require a hands-on activity like a game or interactive learning, whereas something a bit simpler might be well received with just a short read or infographic.

Microlearning also works well when it is used as a refresher or a reinforcement of concepts that were taught in longer form training. The organization may have more formal annual training that is used to teach core concepts, but microlearning that is targeted to specific topics that were covered in the formal training can be sprinkled throughout the year to help reinforce those topics. This is especially helpful when it ties in resources that the organization maintains like documentation.

When considering an approach to microlearning the organization should consider the following:

- Find the optimum length and delivery mechanism. As mentioned, microlearning can be delivered in more than just a video format.
- Ensure that the approach has support from the organization and the leadership. This will ensure that it is not considered as an optional training but is instead considered part of the overall education of the audience.
- Make sure that the training has clear objectives and that there is an approach to measuring success. This could be as setting an objective to reduce a certain type of vulnerability in a particular development team. This can be measured by tracking net new vulnerabilities of that type.
- Considering mixing up the delivery methods so that the training doesn’t become repetitive and stale. This means that not every microlearning on a topic like cross-site scripting needs to be an infographic. Have several options including a hands-on option.
- Keep metrics on the microlearning like frequency, most visited, and who is accessing. This will help with making improvements over time.

One of the most critical aspects of microlearning is that it is delivered and available at the right time. As I mentioned previously, you want to be able to reinforce good habits when the bad habits are being identified. If you cannot integrate with your developers' IDE to link out to a microlearning module, then look to your organization's learning management system (LMS) or collaboration environment to host the content. The important thing is to make sure that when an issue is found, the developer has access to resources that will provide them guidance.

As an example, I’ll turn back to Superior Products. The organization has decided to incorporate microlearning to target two specific vulnerabilities that have been showing up frequently in their scanning tools and penetration test. One is SQL injection and the other is cross-site scripting. In order to take advantage of microlearning to combat these two issues Superior Products has incorporated two methods of delivery:

- Two infographics that reinforces the basic security methods to combat both SQL injection and cross site scripting (XSS), namely:
- Scrubbing and sandboxing user input
- Use an allow-list of known good values to pattern match
- Using parameterized queries instead of concatenated SQL statements
- Two separate videos for each vulnerability that are 5-6 minutes long. They briefly

show how SQL injection and XSS are used by attacker followed by more specific code examples that show how to resolve the vulnerabilities.

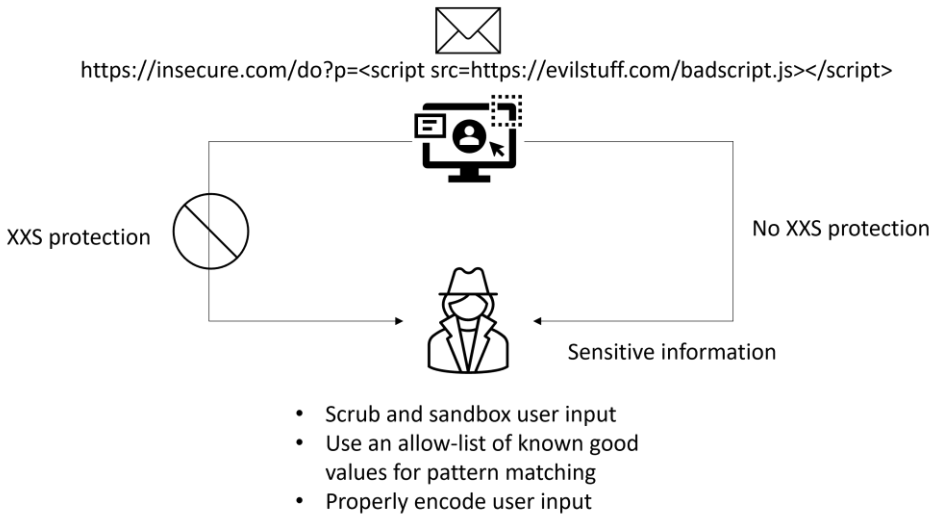


Figure 5.2 Simple example of a XSS Infographic used in microlearning

Dashing Danielle has discovered that she can use the SAST tool in the IDE to link to content that the organization hosts. In this case, she has configured any found SQL injection or XSS issues to include the stock remediation recommendations from the SAST vendor with a link to the internal collaboration tool that hosts the infographics. This is a simple solution, and one that Dashing Danielle is able to use to add content to the infographics over time.

> **NOTE** Jira is one of the more popular issue tracking tools on the market. It allows for the tracking of defects and agile project management. Since it is an issue tracking tool, it is often used by security teams to track security vulnerabilities as well.

Additionally, Dashing Danielle was able to build a task in the continuous integration tools that opens an issue in the issue tracking tool for the organization, like Jira ticket, and assigns it as a task to the team member that submitted the code when a SQL injection or XSS vulnerability is discovered. The task can do the following:

- Leverage the scan results from the SAST and DAST tools that are used in the CI.
- Pull out the SQL injection and XSS issues from the scans.
- Determine the pull request or code check-in that introduced the vulnerability.
- Open a Jira in the team's project and assign it to the submitting developer.
- Add a link to the opened Jira ticket to the microlearning on the specific vulnerability that was introduced.

In this case the microlearning is hosted in the organization's LMS and is accessible to all developers at Superior Products. This allows access on demand and outside the workflow described. More importantly, the microlearning is assigned to the developer that needs to make the change and the details of the change are bundled with the opened ticket.

Dashing Danielle can also use this process to track how well the team is doing in resolving both the SQL injection and XSS issues. Using Jira, Dashing Danielle is able to see the opened and closed issues related to each vulnerability. Over time, if the training and other process in place are working, these should be reduced over time.

5.2.3 It's more than just training

Training doesn't have to be boring. Something as simple as an hour presentation on a security topic once a month goes a long way. Some organizations call this a lunch-and-learn or a brown-bag session. These can provide a quick and interactive method of discussing security topics that are relevant to the organization. What is most effective with these types of sessions is when they are led by people from the engineering organization that are working on something that has security implications like the following:

- New authentication or session management framework being developed in an application
- A technology change that requires a different security model
- Applying encryption using a new database technology
- Secure messaging for clients

The engineering team can be nudged by the application security team, but it's an even bigger win when the engineers bring topics to one of these talks. The goal is to bring topics that the rest of the organization can learn from. What did they encounter when developing the new feature that might cut down on the churn for others? What makes the new feature a better security model than previously? When it comes from a peer team in engineering the other teams are more likely to welcome the conversation and take advantage of the retrospective, especially if the technology stack is the same or similar.

When such an event cannot be coordinated with the engineering organization, the application security team should take advantage of the time to bring in new topics that are important to the security of the organization. For example, a change in the industry that requires support from engineering, like the retirement of insecure standards, is a prime topic to be discussed in this forum. The application security team can also use this time to present what other similar organizations are doing the provide better security. For instance, if the organization is in the healthcare space the application security team can bring in topics related to new regulations that impact the security and privacy of the data that the organization collects.

The application security team should also look to bring in speakers that are external to the organization if appropriate. It's one thing to have an engineering team or the application security team discuss certain security topics, but an entirely different thing if a speaker from the industry presents a topic that is relevant to the organization.

Conferences, meetups, and other forums are also a good source of security information for the engineering organization. As security leaders, we should be encouraging members of

the organization that are outside of security to attend security conferences. However, many engineering-oriented conferences usually run a security track. These tracks are great considering that they are normally on topic for the specific conference. For example, if the conference is specific to deployment technology and methodologies, you'll see the security track cover the security of infrastructure code and the running environment.

5.3 Standards, requirements and reference architecture

Tools and training can provide some reasonably good security controls by alerting, defending and teaching. However, building a house first requires a good blueprint. That blueprint needs to be built on solid guidance, historical evidence, heuristics, and best practices. Translated to the software development business this means having a solid foundation built on standards, requirements, and architectures. These should come from:

- Input from the industry that the organization is in like healthcare or finance
- Input from the broader security industry that apply to the organizations services
- Input from within the organization based on previous discoveries and learnings

However, writing standards, requirements, and architecture is only the first step. Getting buy-in requires structure.

5.3.1 Creating and driving standards

In many organizations you will find a group that is dedicated to ensuring that the organization is following a body of standards. This often is driven from an enterprise function, like the enterprise architecture team if it exists, but can also be from a Center of Excellence (CoE) or a Community of Practice (CoP). Regardless of where the direction comes from, having a central group that defines standards for the organization helps establish the foundations needed to build security in early. Often organizations will take the approach of running what's called an architecture review board (ARB) to drive standards and best practices across the organization. This team is commonly decentralized and spans several domains with experts in each domain. Every organization is structured differently so the domains may vary by name and practice, however, the guiding principle here is to ensure that representatives that have a vested interest in ensuring that the organization has a common approach to building solutions and provide the appropriate leadership and oversight at the domain level while using domain expertise to create and validate architecture.

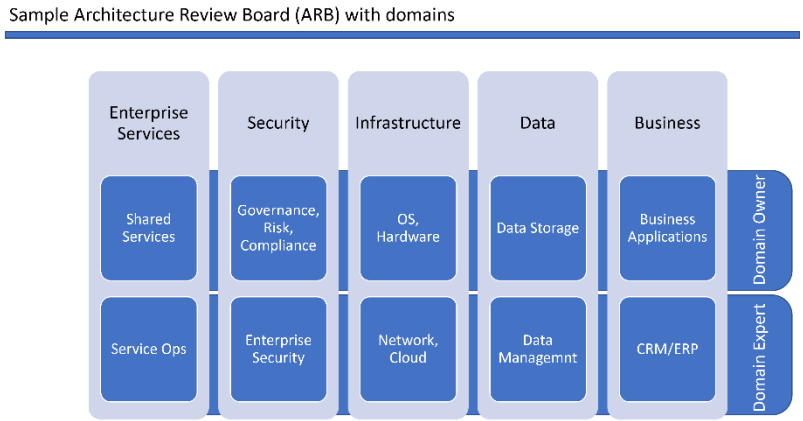


Figure 5.3 Overview of an architecture review board

The ARB will be tasked with creating a consistent method to how architecture problems are solved and setting guidelines on technology that is to be used in architecture. This is often a consensus board meaning that representatives from the various stakeholders in the organization will participate and have a voting interest in the decisions and the setting of standards. There are a few things that are gained by an ARB in an organization.

- Better visibility of the solutions across the organization
- More consistency in an architectural approach
- Reduce the complexity of systems by potentially reusing patterns
- Potential consolidation through visibility
- Awareness and education through shared knowledge

So, what is security's role in the ARB? It's clear that if you want to drive good security standards and patterns, this is the place to be. A prime example of levering the ARB to drive a security standard is to standardize on the version of transport layer security (TLS) that is used in the organization. It is not uncommon for organizations that have several business units or multiple products with different customers aligned to the products, to have varying versions of TLS running in their environments. The security representative in the ARB, who is usually a senior architect within the security organization or the security architect in the enterprise architecture team, will be able to bring forward a requirement to standardize the organization on a particular TLS version.

To illustrate this, I'll turn to Superior Products again. Dashing Danielle, due to her exceptional security leadership and trust among her peers has been brought in as the formal security representative in the ARB after the previous security representative on the ARB left to pursue a career in asparagus farming. For her first task on the ARB, Dashing Danielle is determined to move the organization to a more secure version of TLS. As of this writing, the most common and secure TLS version is TLS 1.2. Although, TLS 1.3 is out and available, the technical hurdles for client migration to using 1.3 are substantial. With this in mind Dashing Danielle crafts a short presentation that describes why the need to standardize on the

version of TLS including the security implications, but also the increase in request from customers regarding their Superior Products stance on the supported TLS version. Along with the presentation Dashing Danielle creates a brief written standard on TLS that looks something like this:

> **TLS is a protocol created to provide authentication, confidentiality, and data integrity protection between two communicating applications.**
> **Applications in Superior Products shall support TLS 1.2 by January 1, 2022. After this date, servers shall support TLS 1.2 for client-facing applications. Lower versions of TLS such as 1.1 and 1.0 shall be disabled on servers that support TLS 1.2 if it has been determined that the lower versions are not needed for interoperability.**

This is a simple statement that addresses the what, when, and who the standard applies to. Armed with the standard and the presentation, Dashing Danielle joins the next ARB meeting that occurs on a bi-weekly basis where she presents the case for standardizing across the organization on TLS version 1.2. She receives backing from some of the members of the ARB, but others are skeptical of the customer impact. Dashing Danielle promises to come back to the next meeting with more concrete evidence of the client impact so that the standard can be put to a vote and integrated into the approved standards.

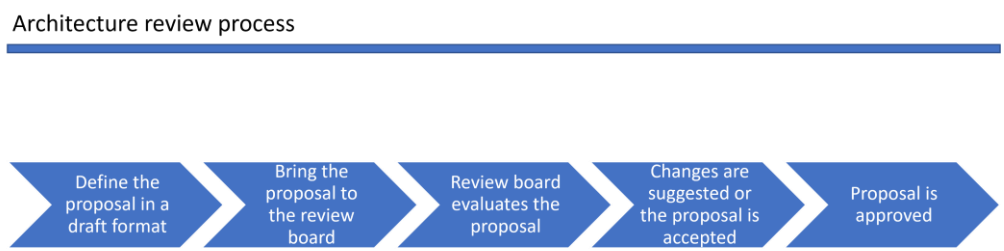


Figure 5.4 Approval of an architecture in a review board

Dashing Danielle works with several stakeholders in the operational organization as well as client services to test a non-production environment in order to gather metrics on the impact of the change. Additionally, she gathers metrics from the network team to understand what customers may still be connecting on the lower versions of TLS so that the team understands the size of the change needed. With this information, Dashing Danielle is able to put the ARB at ease on the required change and the impact to customers. With this hurdle overcome, Dashing Danielle brings the standard to review and approval in the ARB where it becomes a formal standard that is now required to be adhered to by January 1, 2022, by the products within the organization.

However, Dashing Danielle is not able to rest on her laurels at this point, it's now time to do the hard work of driving the change across the organization. Although the ARB has approved this standard, it's up to Dashing Danielle to help drive the change across the organization. With the data that she gathered as part of the development of the standard, she begins to work with the customer services teams to develop a communication plan to the customers. She works with the operational teams to establish timelines and processes to initiate the enablement of TLS 1.2 and the disablement of the lower versions. Throughout the rollout in the pre-production environments, she works with the operational and development teams to resolve any discovered issues as part of the change. Additionally, she works with other members of the ARB and leadership in the organization to make sure that it remains a priority in the organization.

Although this is a simple illustration, the reality is that this would most likely be a much larger project in most organizations led by a project manager with all the stakeholders involved. The purpose of this illustration is to show how external pressure from the industry can lead to the creation of a standard, and the process to get that standard approved and eventually realized by the organization.

> **EXERCISE** Take a look at your organization and think about an area that lacks standardization. Some of the easy ones are around encryption, technical debt, versions of protocols, or even use of certain software. Draft a standard that would address the gap. If your organization is ready for it, take it to the ARB or equivalent. If your organization does not have an ARB, bring it to your engineering leadership for review.

5.3.2 Creating reference architecture

The ARB described in the previous section has another function as well. This is to review and approve architecture that should be implemented across the organization. This doesn't mean that all architecture designs need to go through and be approved by the ARB, it means that architectures that have broad impact or can bring savings to the organization should be reviewed and built with consensus. A well-devised ARB addresses architecture decisions to provide the following benefits to the organization:

- Scalable education to stakeholders and architects on new standards, architecture, and strategic direction.
- Communication channel to deliver documentation including the boards decisions, metrics, and current projects
- Review of high impact projects that are business critical, and impact architectural quality and alignment

With these benefits in mind the organization leverages the ARB to approve of cross-business critical architecture that has a business impact. This is often called *reference architecture*.

DEFINITION Reference architecture is used to provide a template and common taxonomy for a solution in a given domain. It aims to create commonality for developing a solution. Additionally, the purpose of creating reference architecture is to align the organization's strategy around a specific set of tools, standards, guidance, and implementation. This helps drive lower costs in not just licensing but also in development time and effort.

Similar to creating standards, reference architecture is a prime opportunity for security to be injected into a process whose goal it is to develop and design a common approach to solving a problem. There are several opportunities to inject security into reference architecture. Some examples are:

- Federated identity management
- Secrets management when encrypting client data
- Authorization model using OAuth

These examples provide an opportunity to bring together architecture that meets the organization's need to solve a problem, like an authorization model, but does so with the intention of building security in. Once this reference architecture is developed and approved by the ARB, it is ready to be used by the organization when various applications are looking to develop a solution. Leaning on the example of the authorization model, building this reference architecture means that each application will have the same approach to designing authorization into their application. When the reference architecture is built with security in mind, and it is reused, this cuts down on the security concerns of that application going forward. It also means that if issues are discovered or changes need to be made to the reference architecture, that those changes can be easily communicated to the impacted applications through the ARB.

It's clear to see why this is important from a security perspective. Being able to design securely once and reuse everywhere is powerful. However, in reality there will be nuance to the adoption of that reference architecture. Not every application will be able to leverage the reference architecture verbatim and it is impractical to build a reference architecture that covers all possible scenarios. This is where local architects will need to interpret the architecture in a way that meets the spirit of the design, especially the security aspects, while applying it in the context of the application.

I'll use our favorite organization Superior Products to illustrate this further. Superior Products has several applications that are accessed through an API gateway. The user logs in through a mobile device or other user agent to the API gateway which then provides the access to the services through an API provider. This is a less than ideal scenario since Superior Products has to manage the user credentials in a datastore. The organization would like to move to a solution that allows for decentralized authentication through a third-party identity provider.

DEFINITION An identity provider (IdP) is used to create, maintain, and manage identity information through a decentralized network. This distributed network provides authentication information and services.

Dashing Danielle has heard the pain points from several different applications on the management of credentials and ensuring that users are able to gain access to the API services they offer. With this in mind she sets off on some research to determine what opportunities there are to develop a pattern and reference architecture that can be used by the applications in Superior Products. She comes across several different options but lands on a solution that leverages a third party IdP solution that is popular in the industry.

> **NOTE** There are many IdP solutions on the market. One of the leaders is Okta, but there are others like Ping Identity, RSA, Microsoft, and Oracle.

Dashing Danielle works with the third party to develop a basic architecture that can be used to bring to the ARB to get buy in. The architecture consists of a simple diagram that includes an unnamed IdP, the API gateway, and the backend services.

Simple reference architecture for API access

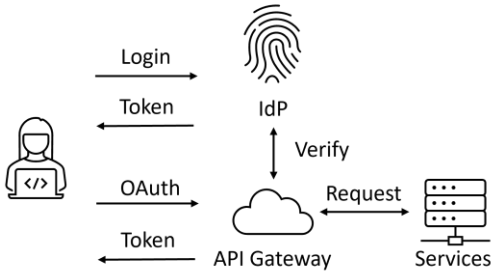


Figure 5.5 API architecture to be used as reference architecture

Dashing Danielle documents the following workflow to help establish the basic understanding of how the IdP will be used in accessing Superior Products APIs:

- A user will login through the mobile or web application where the IdP will then be used to check the credentials and issue a token.
- Another token is created for that user and app through the API gateway's OAuth token generation service. This is used to access the application going forward.
- The API gateway and the IdP ensure that the request is within the boundaries of the user's permissions and returns a token to the end-user application that is used to call the actual API.
- The API gateway verifies the token before serving up the requested API.

Dashing Danielle brings this architecture pattern to the ARB where, after several iterations and clarifications, it is approved as a reference architecture in the ARB. Dashing Danielle is then able to take this approved architecture to the various applications that are currently experiencing issues with managing users of their API's. She coordinates with the chosen third-party IdP to bring the solution across the board at Superior Products.

NOTE To learn more about APIs and security, check out API Security in Action by Neil Madden.

At Acme Services, things are not going as well. There is no ARB which means that there is little opportunity to coalesce around a common architecture or standard. Each product finds themselves often developing a design that resembles their peer applications but using different solutions. This drives up the cost to the organization. Because there is no ARB, two of the teams came to the same conclusion that they wanted to leverage a third-party to manage identities for them but ended up pursuing two different third parties leading to additional complexities in the organization's architecture. Eventually, their API economy is accessed in a disjointed way that increases cost, reduces supportability, and becomes a nightmare to manage.

5.3.3 Bringing requirements into the organization

One of the benefits of creating the reference architecture and standards is that these will lead to the development of requirements that can be easily consumed by development teams. Where standards and reference architecture focus primarily on the strategic level, requirements bring it down to the level where the code is actually being developed. As a reminder, requirements are part of the initial steps of software development. These are used to describe how the software should behave and the various goals it is being designed to achieve.

As I have been laying out in this section, the ARB is used to approve and manage the various architecture and standards, but it does not deal with requirements as that is in the hands of the local development team, including the product owners and architects. However, the reference architecture, standards, and industry practices are used as guidelines for developing those requirements. For instance, in the case of the reference architecture described previously, there would be several requirements around the access of the API gateway, the token generation, the IdP, and others. From a security standpoint, these requirements would focus more on ensuring that the architecture is applied in a way that does not compromise the confidentiality, integrity, and availability of the application and the data. That means that, as an example, the requirements will be written to state that tokens will be send over a secure channel (for example TLS), that the tokens will be short-lived which reduces the potential exposure, and that the design is able to handle potential disruptions with the IdP.

EXERCISE Using the reference architecture earlier, write a requirement that can be used to implement a portion of that architecture. Even better, if your organization already has reference architecture in place, access it and write a requirement that meets a portion of that architecture. If the team you work in is required to meet that architecture, introduce the requirement in your team.

When creating requirements, it's important to ensure that they provide sufficient guidance on what is being asked and that, especially when they are tied to an architecture or industry standards, they meet the spirit of the overall guidance. Being able to create requirements that meet the overall framework means that the architecture can be met by

the development team through prescriptive statements. There are other guiding means that can assist in generating requirements. I'll talk about those next.

5.4 Maturity models

Where reference architecture and standards are primarily developed within the organization to give specific guidance on what the organization believes is best practices for developing software, frameworks and *maturity models* are used to help organizations measure themselves against the industry.

> **DEFINITION** A maturity model looks at the people, processes, and technology in an organization and measures that against a numbers-based maturity level. It is a measure of the organizations current position in a given disciple and provides steps on how to raise the maturity to a higher level. Although in our case we are curious to know the security maturity of an organization, maturity models are used broadly in other technology domains.

There are two well-known maturity models that are used for developing security. One is OWASP's software assurance maturity model (SAMM) and the other is building security in maturity model (BSIMM).

- OWASP SAMM: https://owaspsamm.org/
- BSIMM: https://www.bsimm.com/

Both are useful ways for an organization to understand where they need to put their focus when it comes to developing security practices. However, as you will see, each of their approaches are different.

5.4.1 OWASP SAMM

OWASPs SAMM is a maturity model that follows a more traditional approach to raising the security maturity of an organization. SAMM focuses on the secure software development lifecycle while being designed to be technology agnostic. There are three levels of maturity in SAMM with one being the lowest and three being the highest. It is not important for every organization to put together a roadmap that gets them to level three, as every organization has a different risk appetite. If the organization does not process or manage any sensitive data, and is not have any critical products, level one or two might suffice. The current version of SAMM contains five functions:

- Governance
- Design
- Implementation
- Verification
- Operations

If this looks familiar, it's because this tracks well with the SDLC and enables the organization to assess and then build a plan to address gaps in each stage of the SLDC. Within the five business functions there are fifteen security practices that align to those functions. Each practice dives into the specific items that are used to meet that practice and

have a set of activities that align to three maturity levels. With each level becoming increasingly more difficult to achieve.

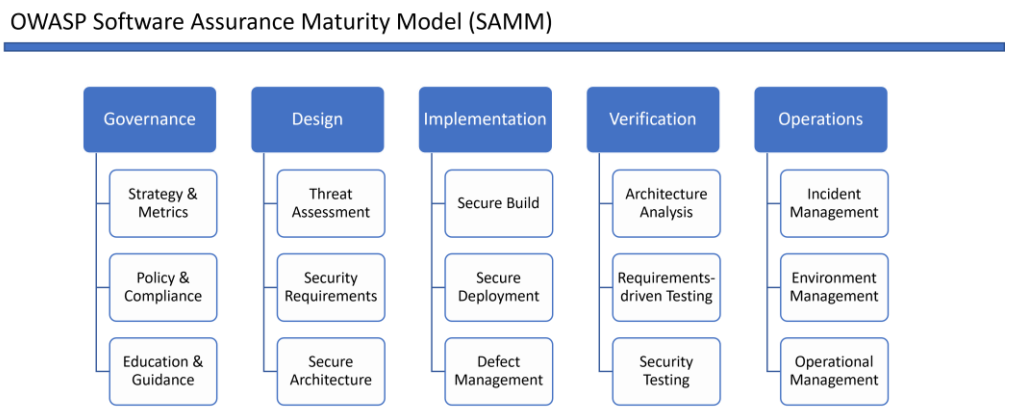


Figure 5.6 Overview of the SAMM version 2

For example, table 5.1 shows the security practice of Threat Assessment under the Design function in SAMM. The focus is around application risk profile and threat modeling so each of the questions are related to the organizations practices around risk profiling and threat modeling to identify the current maturity level.

Table 5.1 Example questions in SAMM related to the Threat Assessment practice

Maturity Level	Question
1	Do you identify and manage architectural design flaws with threat modeling?
	You perform threat modeling for high-risk applications You use simple threat checklists, such as STRIDE You persist the outcome of a threat model for later use
2	Do you use a standard methodology, aligned on your application risk levels?
	You train your architects, security champions, and other stakeholders on how to do practical threat modeling Your threat modeling methodology includes at least diagramming, threat identification, design flaw mitigations, and how to validate your threat model artifacts Changes in the application or business context trigger a review of the relevant threat models You capture the threat modeling artifacts with tools that are used by your application teams
3	Do you regularly review and update the threat modeling methodology for your applications?
	The threat model methodology considers historical feedback for improvement You regularly (e.g., yearly) review the existing threat models to verify that no new threats are relevant for your applications You automate parts of your threat modeling process with threat modeling tools

Each of these questions is designed to highlight processes that the organization should follow to provide better overall security in their SDLC, with each process becoming more difficult to obtain. In this case, simply performing a threat model is a foundational step. Having a process that revisits that threat model and methodology on a regular basis is a more mature approach to identifying application business risk.

With the assess and build process the organization can leverage SAMM on existing products and lifecycles. Should the organization purchase another company or product, SAMM can be used to measure the security of newly acquired software. There are four basic steps to SAMM that assist the organization into assessing their current level and implementing a roadmap to a more secure level.

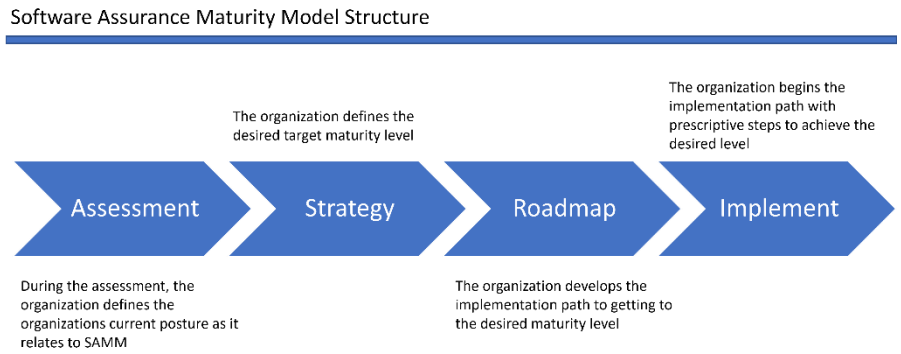


Figure 5.7 Structure of the SAMM

Similar to other maturity models the first step is to obtain the organizations current maturity. In this stage the organization will evaluate its current posture and practices by identifying and interviewing stakeholders. OWASP has developed a handy toolbox that provides interview questions that should be used during the assessment: https://owaspsamm.org/assessment/

Here are some example questions:

- Do you understand the enterprise-wide risk appetite for your applications?
- Do you regularly review and update the strategic plan for application security?
- Do you publish the organization's policies as test scripts or run books for easy interpretation by development teams?
- Do you require employees involved with application development to take SDLC training?

The goal of these questions and the responses is to build a rating that is used to ultimately provide the maturity level of the organization. Although not all questions are yes/no, the SAMM toolbox allows the responder to provide some nuance to the answers such as yes with a caveat. In other words, "Yes, we review x annually" or "Yes, we review x before significant decisions". Once the interviews have been completed, a score will be associated with each of the functions and security practices that shows how the organization meets the maturity in that given function. This score is then combined with the other areas to create an overall maturity score for the organization. In order to successfully perform an interview, you'll want to ensure consistency with the interview process with the interviewer and be flexible with the process. Look at different formats for performing the interview like anonymous questionnaires, workshops, or in-person interviews. The organizations culture should be taken into consideration during this activity to ensure the organization gets the most benefit.

Now that the organization understands it current posture, it's time to set a target and define a plan. As was mentioned before the target does not necessarily need to be level three across the board. In some cases, the organization may take the approach that they will accept a certain level of risk due to not being at the highest level, or the application is not

sensitive enough to warrant the overhead of the higher maturity level. Although each effort varies, when setting the target, it's critical to understand what the effort might be for the application team. This also means that the team must include the dependencies that might be on each of the activities with other teams and external sources.

With the assessment complete, and the targets defined, the team can follow a general roadmap planning exercise that outlines how the organization will gradually builds the maturity to the desired level over time. Although there is no prescriptive timeframe related to SAMM the organization may be under internal or external pressure to obtain a particular level by a certain time. For instance, the organization may be asked by clients to adopt a formal process around something like threat modeling by the year end. If the organization does not currently have a threat model process, they would want to prioritize the threat modeling activities in the roadmap and work backwards from the year end date in order to define the milestones.

Lastly, the fun part: implementing the path to a more mature secure SDLC. With the roadmap in hand the application security team will work with the organization to help adopt the new processes. It is important for the stakeholders to be aware of the changes, the purpose, and the milestones all throughout this phase. The application security team will need to collect metrics and track progress along the way to ensure that the organization is on target for the desired level.

The SAMM provides an organization a well-defined approach to discovering their current security posture as it relates to software security. It is a prescriptive approach to building maturity. But wouldn't it be nice to see what other organizations are doing to build security into their development process?

5.4.2 Building Security in Maturity Model

Where SAMM allows the organization to look at their current security posture in the secure software development process and build a roadmap that reflects OWASP's guidance maturing, BSIMM looks at the industry peers of the organization to help build that roadmap. BSIMM is built on interviews with 128 organizations over nine industries. Like SAMM, BSIMM is organized into four domains with twelve software security practices and 122 activities.

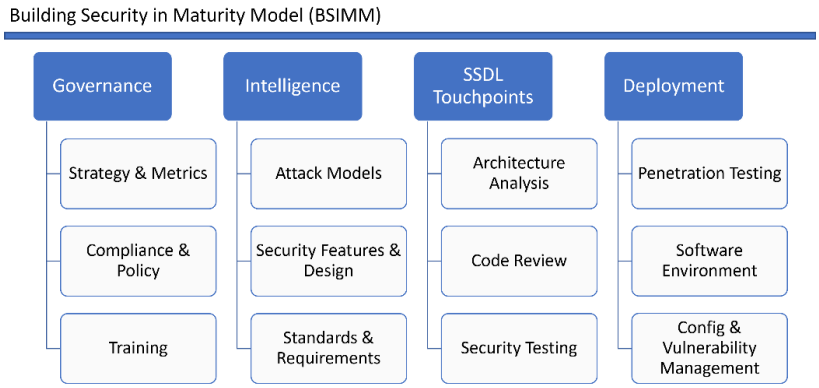


Figure 5.8 BSIMM structure and organization

As of this writing, the current version of BSIMM is version 12 which included 9,000 software security members in both the security groups as well as security champions in the organization. The industries that were included in BSIMM 12 were:

- Financial
- FinTech
- Independent software vendors
- Tech
- Retail
- Insurance
- Healthcare
- Cloud
- IoT

BSIMM includes three levels of maturity that relate to the observed frequency of the given *activity*, with level 1 being infrequently observed and level 3 being most frequently observed.

> **DEFINITION** **Activity are actions carried out or facilitated by a software security group (SSG) as part of a practice. Activities are divided into three levels in the BSIMM based on observation rates. Frequently observed activities are designated level 1, with less frequent and infrequently observed activities designated as levels 2 and 3, respectively.**

For each of these activities the organization can measure their posture against others in their particular industry. Although this is not the prescriptive approach that OWASP takes with SAMM, it helps the measuring organization know how it measures up compared to its peers. One of considerations with BSIMM is that the best practices that are outlined may not be a great fit for your organization, but it is hard to argue if you come across activities that are being completed by many organizations in your industry. One example of this is identifying your personally identifiable information (PII) obligations which means knowing the

requirements related to the capture and retention of PII in your system as well as building an inventory of PII across the organization. When the vast majority of the organizations that participate in BSIMM are doing this activity, it's a safe bet that your organization should be as well.

To this point the BSIMM provides five of the top activities that most organizations should pursue regardless of their industry. These are activities that are being integrated in almost all of the organizations that are part of the BSIMM. Once again, peer pressure is a strong motivator and these activities have consistently ranked at the top five for several years.

Table 5.2 Example questions in SAMM related to the Threat Assessment practice

Activity Name	Domain and Practice	Description
Implement life cycle instrumentation and use to define governance	Governance: Strategy and metrics	This can be best defined as applying application security tools into the SDLC process in order to gather metrics as it relates to the security policies in the organization.
Ensure host and network security basics are in place	Deployment: Software environment	Before running your developed software, ensuring that the environment it will be deployed to is secure. That means having the systems and network that it's deployed to properly secured.
Identify PII obligations	Governance: Compliance and policy	No surprise, but this is related to the data that an organization accumulates. More importantly, it's about knowing what type of data is being stored and where it is being stored.
Perform security feature review	SSDL touchpoints: Architecture analysis	When architecture changes that impacts the security level or model, an architecture review should occur to understand the risk.
Use external penetration testers to find problems	Deployment: Penetration testing	Your internal testers and tools are great at finding certain issues, but nothing brings a security issue to the forefront quicker than an issue uncovered by an external tester.

While these activities are the most common across the organizations that are part of BSIMM, it's important to see the trends of where organizations are showing the most growth over the past versions. For instance, the top three activities that have seen the biggest growth in the past several versions are:

- Use orchestration for containers and virtualized environments
- Ensure cloud security basics
- Use application containers to support security goals

Each of these have seen several hundred percent increases from the past versions, meaning that more organizations are beginning to adopt these activities over the past few years. This is no surprise given the increase in adoption of cloud and container technology. As these technologies expand, the need to increase the security around them grows.

What do these metrics show about an organization's software security initiative? It shows not only how an organization measures up to its peers, but how the industry in general is moving and how the organizations are addressing the new concerns. Armed with this knowledge, it's time to build a plan around how to attack any known deficiencies in the organization with a Maturity Action Plan (MAP). This helps the organization address the open challenges and objectives by building an action plan that raises the maturity. Currently Synopsis offers services that help the organization determine what steps need to be taken to build the MAP. This includes first looking at the organizations current posture with an assessment of the security program. From there a future state is defined and the gaps are highlighted to build the roadmap. The last step is to develop the action plan to achieve the desired state.

As I mentioned, BSIMM is a great measure against peers in your respective industries. One of the key aspects to understand with BSIMM is that if most of your peers are tackling software security with the same or similar activities, then this is a good indicator that you should be doing it as well. This is the power of BSIMM.

5.4.3 Addressing your security immaturity

Both BSIMM and SAMM take two different approaches to building secure software. So, which makes the most sense? Of course, it depends. If you absolutely need to see how you measure up against the other organizations in and out of your organization, then conducting a BSIMM engagement makes sense. However, if you are looking to do a basic self-assessment that shows your current software security posture then SAMM is the better option as it is open, and assessments are conducted by the organization.

The most important factor in following a maturity model is to understand what your organization can bear. Getting a BSIMM assessment and a MAP that shows where you should be in the next 12-24 months will be worthless if your organization is not prepared to do the work of increasing the maturity. This means that the leadership must be bought in to the strategy, and the engineering organization must be aligned with the application security function. More importantly, the right resources must be dedicated to the initiative. Take for example that in both the SAMM and BSIMM that architecture analysis is an activity. This activity requires knowledgeable resources both in the engineering team and the application security team to be able to review the architecture for security control effectiveness, and

mitigation of threats. Not all of the organization's resources will be able to perform this level of activity effectively.

The last comparison points to make about the BSIMM and SAMM is to know whether you require or need an external assessment. Since BSIMM is proprietary to Synopsis, the BSIMM assessment is by default external. The SAMM allows you to perform a self-assessment, however, like I stated previously you will need expertise in your organization to conduct that assessment. You can have a third party perform a SAMM assessments if that is the preference though.

There is no right answer with a maturity model, but it is an important factor in what I will talk about next. With a maturity model, especially BSIMM, it makes it easier to sell an overall software security initiative to the stakeholders in an organization and build a better path towards a democratized and decentralized security program that builds security into software early.

5.5 Decentralized application security

As I have mentioned numerous times at this point, securing software requires a village of security minded individuals and teams. Application security resources cannot be in every coding decision, every design review, and every assessment. The current approach of a centralized team that applies software security tools which create an output of security vulnerabilities that need to be resolved in a certain time period creates friction, frustration, and drag on the development of software. Moreover, this centralized application security function becomes a team of ankle biters that are attempting to provide security by telling the development teams that they are awful at writing secure code.

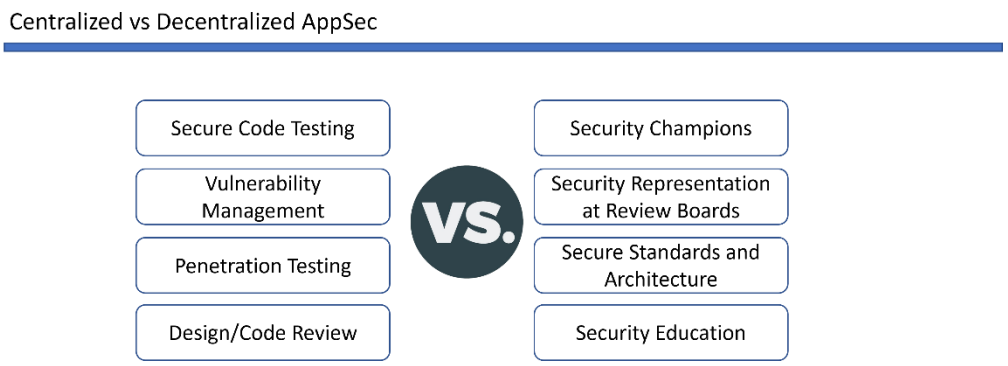


Figure 5.9 Application security centralized approach vs decentralized

Taking the decentralized approach means that the application security team focuses their efforts on building the structure of secure software development as opposed to the daily operational aspects of managing tool and driving closure of vulnerabilities. These are still important and needed in all organizations, however, application security teams cannot scale to the level needed in order to be the ever present.

Building the structure to support the secure development of software, the application security team will create the reference architecture, the standards, and requirements to be used by the engineering organization. Additionally, it is critical that they become a part of the fabric of developing software. This means that they should sit on review boards and other architecture or leadership groups to ensure that security is made part of the process. Lastly, the application security team should focus on building an education structure that helps create champions or, at the very least, raises the security education of the develop organization. I will cover this more in depth in the next chapter, but the critical takeaway here is that security needs to become part of the culture of the organization in order for the software to be developed in a secure manner. One popular method of building or expanding the security culture is with a security champions program.

5.5.1 Security champions program

I've talked about them a bit in previous chapters. They're called many things, mostly champions or coaches, but the goal of security champions is to spread the wealth of security across the organization. Starting a program that build security champions will rely on a change of culture in the organization and includes buy-in from the engineering leadership and organization as a whole.

The general definition of a security champion is a non-security employee that is part of the organization usually in some type of development or development support role. They can be from one of the following roles:

- Developer
- Quality assurance
- Architect
- Designer
- DevOps

To start the security champions program the application security team will first need to identify what teams and applications are part of the program. Not every team or application may take part especially if they are a small team or low risk to the organization. Obviously, the focus will be on the highest risk applications in the organization to begin with. This list should be well vetted and approved by the appropriate leadership in both security and engineering. This will help ensure that no team was missed and that the ones that are intentionally left off the list are agreed upon.

Once the list of teams is agreed upon, the role of the security champion should be defined. The most critical qualification is that they should want to be there and have a passion for security. Forcing employees into the champion program will lead to a failed program. These champions will be required to spend a certain amount of their time on driving security. In some cases, this can be as low as 5% of their time dedicated to driving security in the product area that they are a part of. However, the organization needs to set the time effort expectation early so that there is no confusion around priority and activity. The expectations of a security champion vary from organization to organization, and it depends on their goals.

Table 5.3 Expectations of a Security Champion

Sharing security knowledge	Help in making key security decisions	Create and drive security best practices
Build threat models	Perform security reviews	Participate in research and development initiatives
Participate in bug bounties	Attend security conferences	Prioritize security work in the backlog
Monitor for vulnerabilities in the application	Write security test cases as part of the overall application testing	

Now that the expectations have been set and the teams identified, it's time to nominate champions. This process is as simple as working with the leadership in engineering to identify potential candidates. Once identified, a formal nomination is made of the candidates. This includes communication to leadership, the rest of the champions team, and the engineering team that they are a part of. The organization may also put the nominated security champions through formal training to achieve a level of security knowledge. This is not an uncommon approach in most organizations and can be achieved with either internal or external training. Training can be assigned with target levels to attain or certifications to complete. Some security training platforms include education paths designed specifically for training champions. The organization can also take the approach of leveraging internally developed education from the application security team if it exists.

Communication and the storage of information also need to be defined. Depending on the collaboration tools that are used in the organization, it could be as simple as setting up a channel just for champions in the collaboration tool like Microsoft Teams, Slack, or Discord. Additionally, the champions team will periodically develop content that needs to be stored for easy retrieval. Again, depending on the organization, this could be tools like Confluence, Hive, SharePoint or the like. This content should be used as the knowledge base for all activities for the champions team and the development teams that they are part of and include:

- The champions team charter
- The members of the team and the teams/applications they serve
- Activities and meeting minutes
- Processes for threat modeling, vulnerability management, and performing security reviews
- Training program and opportunities
- External resources

Although nobody likes more meetings, periodic touch points with the group of champions is a required activity. This can be just a simple bi-weekly or monthly that is used to discuss upcoming activities, ongoing projects, and current vulnerabilities. This also aids in keeping the participants engaged in the champions team and raising the visibility across the organization. One way to reduce the effectiveness of the champions team is to allow it to lapse or reduce the engagement of the team members.

With a team defined it's time to leverage it to take advantage of this increase in security footprint. Now the application security team has advocates for building secure software. And these advocates are embedded in the applications. But that's not the end of the decentralized model.

5.5.2 Leveraging the decentralized model

Our favorite security champion Dashing Danielle has been asked to join the application security team more formally as she has been demonstrating a keen ability to deliver security in Superior Product's flagship application. Before she can formally join the application security team, she needs to help find a successor to her role as security champion in the team. She announces her future move to application security to the team and asks whether anyone else would be interested in taking her champion role. Brilliant Brian has been working in the team for several years and has strong knowledge of the product. He has also been in many of the same meetings as Dashing Danielle where security has been the prime topic. He has become increasingly interested in the security space and gladly raises his hand when the opportunity came up to be part of the security functions in Superior Products.

Dashing Danielle first works with Brilliant Brian to develop a training path that will get him to the champion level. At Superior Products, they are using a leading provider of secure engineering training that includes several education modules that are built to create security champions. Brilliant Brian is onboarded in the training program with the first module slated to start immediately. This will get him to the first level of three levels in the platform. Once he completes all three and passes an assessment for each level, he will be considered a security champion. Based on the current pace, Brilliant Brian will complete the training in four months.

While training is underway, Dashing Danielle introduces Brilliant Brian to the various documentation and processes that each security champion must be familiar with. She invites him to the bi-weekly champions meeting where he is able to get a sense of the expectations for him and the other champions in the team. He acquaints himself with the ongoing projects and topics in the champions team.

Fast forward a few months after Brilliant Brian has completed the training and assessment, he is now the active participant in the security champions bi-weekly for the product as Dashing Danielle begins to pick up responsibility in the application security team. Brilliant Brian has been working diligently with the development team to help integrate and maintain the security scanning tools set forth by the application security team. He has joined in on code reviews to identify problematic code patterns that fall outside of the documented practices and reference architecture that has been developed by application security. He has raised the level of security education in team by encouraging other members to the security training offered in the organization. He even submitted a talk to a local conference on some of the techniques he has been using in the team to bring security to the developed code. With Dashing Danielle and the leadership team comfortable with Brilliant Brian's handling of the security of the team's code, Dashing Danielle is able to take the next step in her security career and become a formal member of application security. Brilliant Brian is able to leverage the work that Dashing Danielle has already done, as well as the continuous efforts by the application security team to build the development security ecosystem. From this effort the

development team has seen a reduction in new vulnerabilities being introduced in their code. A win-win for all parties involved.

5.6 Summary

- Security is not just an activity that security teams are responsible for. Each team has a responsibility to ensure that they are considering the security implications of the design choices they make.
- Security teams are not able to scale to meet the demands of security in any sufficiently large organization. This raises the need for more security minded individuals in the organization.
- Security education is not just for security teams, developers and those who support the development process can and should take advantage of security training. This raises the security IQ of the organization and helps spread the knowledge.
- Training comes in many different forms, and can be microlearning, traditional classroom training, conferences, brownbag sessions, and others. A mix of some or all of these will help the organization raise the security awareness.
- Standards, requirements, reference architecture are methods to create content that can be leveraged by the development teams which helps to decentralize the application security function.
- Maturity models can be used to determine where your organization is today and where you want to take it. For maturity models like BSIMM, you can measure your organization against similar organizations in the same industry. This is helpful when you need to understand where others are placing their efforts.
- A decentralized application security model depends on having well trained individuals that are prepared to address security concerns for their area. This is commonly solved through a security champion's style program that builds security minded individuals into the development teams.