Secure Coding.
Practical steps to defend your web apps.

WEB APPLICATION VULNERABILITY MANAGMENT

GIAC (GSEC) Gold Certification

Author: Jason Pubal, jpubal@mastersprogram.sans.edu
Advisor: Barbara Filkins

Accepted: July 23, 2014

Abstract

For years, attackers have assailed network and system level vulnerabilities, fueling demand for products like firewalls and network vulnerability scanners. As these products mature and IT security teams learn to better handle network security, the industry is seeing a visible increase in attacks moving up the stack to target application-level vulnerabilities.

Mature application security programs perform security testing against web applications during development but tend to be reactive to security defects once an application has been deployed. Web applications that are being used in a production environment are not typically subject to recurring security tests. There is fear of causing an impact to functionality and a notion that as long the codebase has not changed it is impossible to find new security vulnerabilities in an application that was thoroughly tested prior to being deployed. Any vulnerability not found during this initial testing is only known about and addressed after an incident has occurred. Meanwhile, infrastructure vulnerability management performs recurring vulnerability scanning against production network and servers as the web applications hosted on them are ignored.

As threats evolve and new attack vectors are discovered, applications need to be tested to see how they are affected. Application vulnerability management needs the same rigor infrastructure vulnerability management has; web application vulnerability assessments need to be continuous. The web application vulnerability management framework presented in this paper is the next step in application security. This framework introduces a methodology, processes, and activities to achieve that goal.

1. Introduction

For years, attackers have assailed network and system level vulnerabilities, fueling demand for products like firewalls and network vulnerability scanners. As these products mature and IT security teams learn to better handle network security, the industry is seeing a visible increase in attacks moving up the stack to target application-level vulnerabilities.

Many companies conduct continuous or frequently recurring vulnerability scans of their infrastructure, but ignore applications once they are deployed to production. Those applications make up a large part of the attack surface, and are where attackers are focusing their attention. According to Verizon's Data Breach Investigations Report (2014), 35% of breaches were caused by web application attacks, making it the most prevalent attack pattern.

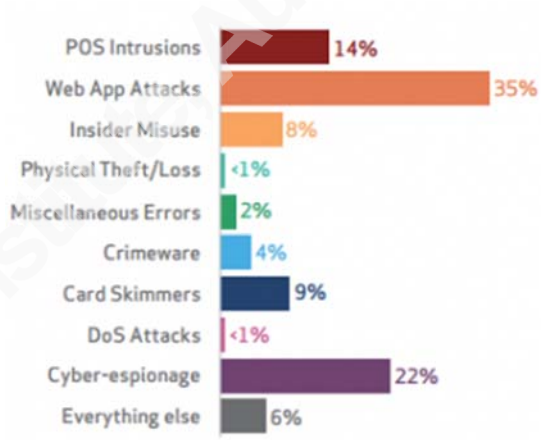


Figure 1. DBIR Frequency of Incidents. From "2014 Verizon Data Breach Investigations Report," 2014.

Our state-of-the-art models for application security fall short when it comes to vulnerability management; they are reactive and only take action after damage has already been done. To establish an effective application security program, organizations need to implement application-level vulnerability management as an ongoing process. The web application vulnerability management framework detailed in this paper lays out a methodology that can be followed to help secure web applications. I developed it over the last couple of years of my career, and have successfully implemented it to conduct

Jason Pubal jpubal@mastersprogram.sans.edu

vulnerability management against hundreds of web applications at a large financial services company.

1.1. Risk Management

Vulnerability is a component of risk. Vulnerability management plays a crucial role in finding a variety of technical vulnerabilities in an environment, prioritizing the resulting risk, and improving the overall security posture by addressing those likely to lead to incidents. Harris (2012) defines risk as "the likelihood of a threat agent exploiting a vulnerability and the corresponding business impact" (p. 57). Expressed as a formula, that is:

$$Risk = \frac{Threat \times Vulnerability}{Countermeasures} \times Value$$

Breaking down the components of risk, threats are someone or something that can do harm. Vulnerabilities are the weaknesses of an asset that allow the threat to cause harm. Countermeasures are precautions that have been taken. Value is the monetary worth of the asset representing the potential loss of an incident.

Information security is about managing risks to sensitive data and critical resources. There will always be some amount of risk present, the role of information security is to bring it in-line with organizational policies. The tolerance level for risk will vary between organizations. Information security professionals have to gauge the executive team's risk appetite and help them make well-informed decisions that are in the best interest of the company.

These decisions can include accepting, transferring, avoiding, or mitigating the risk (Wheeler, 2011, p. 54). Sometimes it is in a company's best interest to accept a risk. If the cost of risk mitigation is more than the value of the asset being protected, then making a formal decision to accept the risk makes sense. For example, it would be unwise to spend $5,000 on a fence in your backyard only to protect a $500 grill. Transferring risk shifts liability for the risk to another party, and is often done through purchasing insurance. Avoiding risk is done when the risk is so high that the best the best

course of action is simply to cease the activity that presented the risk. Risk mitigation is taking steps to reduce risk exposure. This can be done by changing any of the variables in the risk formula. However, it is difficult to control threats, and it might not make sense to reduce the value of assets. Most commonly, applying countermeasures and reducing vulnerabilities are used to reduce risk. This is where vulnerability management comes into play.

It is important to put vulnerability management in the larger context of risk management. The goal is not to eliminate every vulnerability, but finding the right balance. "The security program should be filling a governance and oversight role to help identify the risks that have the greatest chance of harming the organization with the most severe impact. If you have properly educated the organization about the likely risk exposures, then you have fulfilled your obligations even if the business chooses not to address the risks" (Wheeler, 2011, p. 24). Security should be a trusted partner that helps the business make well informed risk decisions.

1.2. Vulnerability Management

Vulnerability management is the "cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities" (Foreman, 2010, p. 1). Nicolett (2005) defines it as an ongoing several step process as shown in figure 2: policy, discovery and baseline, prioritization, shielding and mitigation, eliminate root cause, and monitoring. (p. 2)

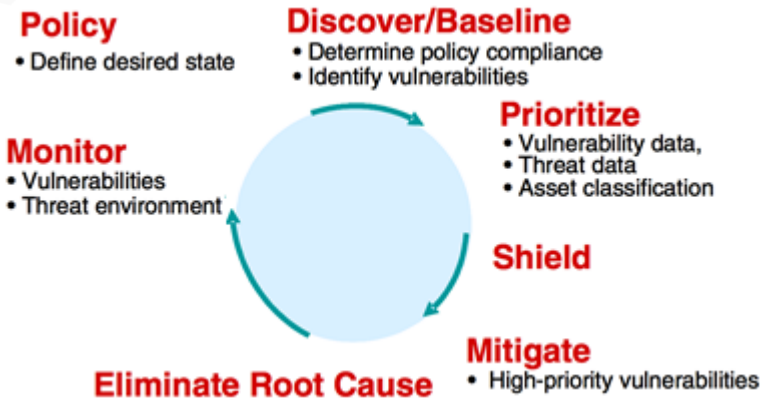


Figure 2. Vulnerability Management Process. From "How to develop an effective vulnerability management process" by Nicolett, M, 2005. Gartner.

- **Policy**: Define the desired state of the network, system, and application resources in policies and standards. Create policies that define vulnerability and patch management processes along with asset secure configuration standards.
- **Discovery and Baseline**: A prerequisite to vulnerability management is knowing what assets exist. Discovery is the process of enumerating those resources. An initial vulnerability assessment provides a security baseline that contains the list of vulnerabilities present.
- **Prioritization**: With a vulnerability assessment in hand, use that data as part of the risk formula to prioritize the findings. Easy to exploit vulnerabilities on high value assets that are likely to be attacked get a higher priority.
- **Shielding and Mitigation**: Mitigation can be the most challenging part of vulnerability management as it requires changes implemented across the organization that will affect different IT functional areas. Network devices, servers, and applications can be managed by different teams, each of which information security has to work with to fix vulnerabilities. As a mitigation tactic, shielding can block attacks while a patch is being developed. For example, an intrusion prevention system or web application firewall rule might be able to block SQL injection against a web application until developers fix the code.
- **Eliminate the Root Cause**: Evaluate patterns of vulnerabilities to identify and eliminate root causes. Improvements in policy and process can proactively eliminate certain classes of vulnerabilities from systematically reappearing in the environment.
- **Monitoring**: Researchers discover new vulnerabilities frequently, attack vectors evolve over time, and IT environments are always in flux. All of this has to be monitored and taken into consideration through iterations of the process.

Vulnerability management is not something that is done once and completed. Having a vulnerability assessment or penetration test done, and then fixing every issue on the resulting report, is not vulnerability management. "The discovery and baseline steps need to be continuous, and all subsequent vulnerability management steps should be repeated as part of an ongoing process" (Nicolett, 2005, p. 5).

1.3. Software Security Maturity Models

A maturity model is a framework that can be used as a benchmark for comparison when looking at an organization's processes. It is a set of structured levels that describe how well the behaviors, practices, and processes of an organization can reliably and sustainably produce required results. There are two excellent and freely available software security maturity models: OWASP's Software Assurance Maturity Model (OpenSAMM) and the Building Security In Maturity Model (BSIMM).

1.3.1. OpenSAMM

The Software Assurance Maturity Model is an open framework designed to help organizations formulate and implement a strategy for software security. It walks one through conducting an assessment to measure an organization against defined security practices. With the assessment in hand, one then builds a roadmap that outlines an iterative plan where various practices are improved over several phases based on business drivers and risk tolerance.

OpenSAMM defines four business functions: governance, construction, verification, and deployment. Each of these has three associated security practices. For each security practice, SAMM defines three maturity levels. It provides the following visualization.

Figure 3. OpenSAMM Overview. From "Software Assurance Maturity Model" by Pravir Chandra, 2009, OWASP.

1.3.2. BSIMM

The Building Security In Maturity Model is the result of a study of 67 software security initiatives. In its fifth version, it describes activities companies are currently doing around software security so they can be compared with what an organization is doing. With organizational goals and objectives in mind, the BSIMM can be used to determine which additional activities might make sense. It gives an overview of how software security teams tend to be organized, and how large those teams typically are.

The BSIMM is organized similarly to OpenSAMM into what it calls the Software Security Framework (SSF). It has four domains: governance, intelligence, secure software development lifecycle (SSDL) touchpoints, and deployment. Each domain has three associated security practices. Each security practice has a number of activities that were observed at a company as part of a security initiative for a total of 112 different activities. BSIMM provides the following visualization of the SSF.

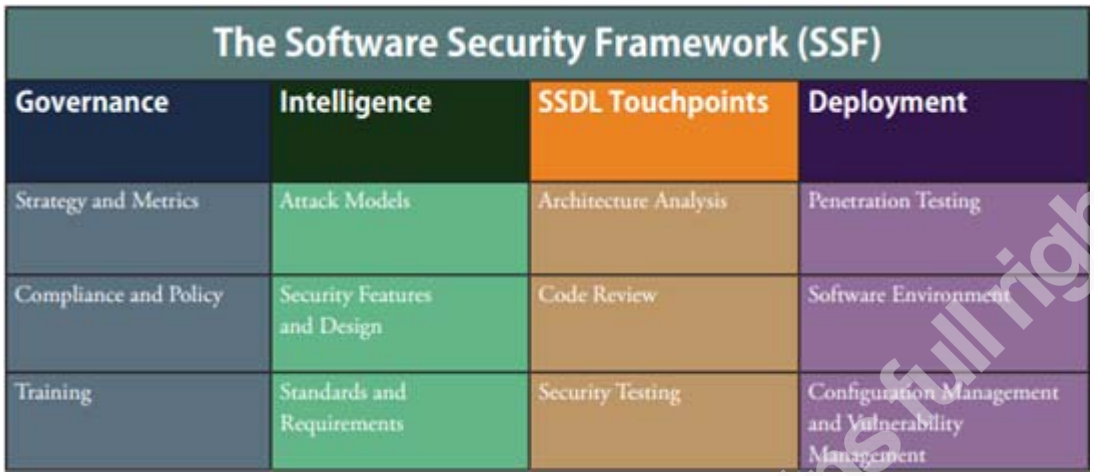


Figure 4. The Software Security Framework. From "Building Security In Maturity Model" by McGraw, G., Migues, S., & West, J. 2013.

1.3.3. Inadequate Vulnerability Management in Current Maturity Models

According to McGraw, Migues, & West. (2013), vulnerability management is "quality control performed during the development cycle" (p. 39). It is a practice that attempts to find and fix security vulnerabilities while software is being written. According to Chandra (2009), vulnerability management is focused on handing external "vulnerability reports and operational incidents" (p. 16). That is, once an application is live in production, vulnerability management is the practice that addresses a vulnerability that was either exploited to cause an incident or reported by a third party.

Both of the above software maturity models are reactive to security flaws once an application is deployed. Neither model covers a "cyclical practice of identifying, classifying, remediating, and mitigating vulnerabilities (Foreman, 2010, p. 1)." In each model, security testing stops once an application is finished being developed and moves to deployment. Vulnerability management as defined in these maturity models only takes place when triggered by an event out of our control such as a security incident.

As new attack vectors are discovered and current threats evolve there is need to integrate these techniques into security testing to identify and address potential vulnerabilities in our existing software. This can only be done with recurring vulnerability assessments. Application vulnerability management needs to have the same kind of rigor that infrastructure vulnerability management does. Continuous application

security assessments against deployed, production web applications need to be conducted as part of an overall vulnerability management program.

2. Web Application Vulnerability Management

For a holistic view of a web application vulnerability management program, it will be covered as a framework. This framework has three domains: governance, verification, and construction. Each of these has a number of related practices that will be covered. First, the prerequisites of vulnerability management. Next, enrolling applications in the program. Then, the remediation process used when a vulnerability is found. Finally, measuring the program with example metrics.

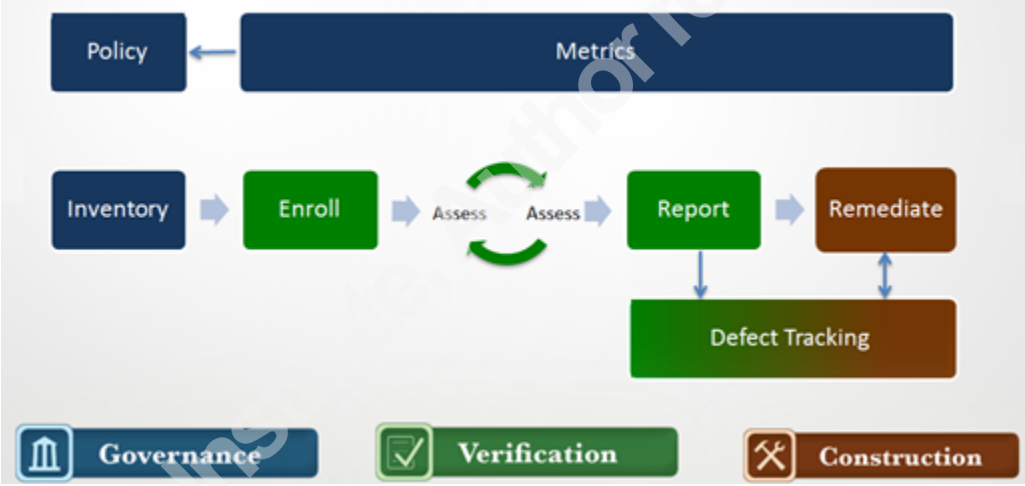


Figure 5. Web Application Vulnerability Management Framework.

2.1. Preparation

Before conducting security assessments of applications, there are a few things that need be in place: a security policy that defines what is assessed against, an inventory of applications to test, and tools with which to perform the assessments.

2.1.1. Policy

Everything in information security should be rooted in corporate policies and standards. Policy is agreed upon by executive leadership, and represents the culture that exists in the organization. “It describes the way in which people behave when doing their

work" and "is a message to the workforce from management to tell them what is expected of them" (Sherwood, Clark, & Lynas, 2005, p. 410).

Policy impacts application security in three crucial ways. First, security policy communicates the intentions of management for managing risk and enforcing security in the organization. It is the document that empowers the security team and gives them responsibility to perform application security assessments in a way that is sanctioned by management. Second, it outlines secure coding practices developers use to create software and the security requirements for commercial off the shelf (COTS) applications. These are the standards to be assessed against. When we find a vulnerability, the documented secure coding practices are referenced to give guidance to teams that need to fix the issue. Finally, policy should contain remediation timelines. This tells various teams how long they have to get security vulnerabilities fixed.

2.1.2. Inventory

Asset management is a prerequisite for vulnerability management. Before one can secure stuff, one needs to know what stuff one has. It is helpful to know how many applications there are, where they are located both physically and logically on the network, who manages them, and what their value is. These attributes should be stored in an inventory.

It is likely, especially for larger companies, to have applications hosted both internally and in the cloud. It is also likely that there are rogue applications that have been acquired outside of official IT processes. Discovery, one of the first steps of vulnerability management, helps find all of these. Discovery is "done through a combination of technical and manual processes. Automation helps inventory what is out there, but asset owners must identify the business function and relative value of each target" (Foreman, 2010, p. 184). To derive the value, approach it both from a data sensitivity and business criticality perspective. That is, knowing the data classification or type of data and number of records can help determine the legal and brand reputation impact, along with the potential amount of costs associated with losing those records. Knowing the impact on business operations helps determine what the potential loss of revenue could be.

Another item to consider keeping in the inventory is software components. The Open Web Application Security Program (OWASP) maintains a list of the top ten vulnerabilities found in web applications. A new addition in the last iteration of this list is "Using Components with Known Vulnerabilities." To prevent this, Williams and Wichers (2013) recommend you "identify all components and the versions you are using, including all dependencies" (p. 15). Identifying and tracking the components being used makes it possible to take action when a security issue is found with one of them.

When building or updating an inventory, using the following reconnaissance tools and techniques will be useful.

- **Google**: Using Google for reconnaissance can be as simple as searching for a company and noting what websites are returned. Google's PageRank, the algorithm used by Google Search to rank websites in their search results, is one way to prioritize what applications to look at first.
- **NMAP**: Nmap, or Network Mapper, is a security scanning tool used to discover hosts and services on a network. Given a company's external IP address range, it will port scan those IP addresses and enumerate which IP addresses and ports are running web servers.
- **Domain Name Service (DNS)**: If your company is utilizing its own DNS services, work with the DNS administrator to run a query to show what domain names the company has registered and are in use.
- **Reverse DNS**: There are services, like ewhois.com, that collect identifiable information on websites. Ewhois.com can search for the email address that was used to register a DNS name, along with other things like the Google Analytics ID being used. Often, a company will use the same registration email address or Google Analytics account across all of their websites.
- **Recon-ng**: Recon-ng is a modular reconnaissance framework. It is designed to take advantage of several third party web based resources from one tool. Through various modules it conducts host discovery, server

enumeration, searches through publicly available databases like PunkSPIDER for known web application vulnerabilities, searches through social networks like LinkedIn for contacts, and leverages various DNS registrars to gather information. Due to its modular design, anyone can write plugins for it allowing its functionality to grow over time and become more useful as a discovery tool.

2.1.3. Dynamic Application Security Testing Tools

Dynamic application security testing (DAST) technologies are designed to detect conditions indicative of a security vulnerability in an application in its running state. Web application vulnerability scanners are tools that scan web applications to look for security vulnerabilities such as cross-site scripting, SQL injection, command execution, directory traversal, and insecure server configuration. They communicate with an application through the web front-end in order to identify potential security vulnerabilities and architectural weaknesses. First they spider the application, starting at the application's front page and recursively following every link to find each page and input. Then they fuzz those inputs looking for responses that indicate security issues.

Choosing a DAST tool will rely on the environment and vulnerability management requirements. Gartner has an Application Security Testing Magic Quadrant that highlights the commercial space.

Figure 6. DAST Magic Quadrant. From "Magic Quadrant for Application Security Testing" by Neil MacDonald & Joseph Feiman, 2013, Gartner.

2.2. Enrolling Applications

The groundwork has been laid, and it is time to start enrolling application into the program. Most often, these applications will come from the inventory created earlier. That inventory is not static; additional applications could be found during future iterations through the discovery and reconnaissance process. New applications may be purchased or developed. Work with the project management and development teams so these new applications can be enrolled before they are deployed. If there is a security incident involving an application that is not included in the vulnerability management program, it may need quickly assessed to figure out what the risk profile of that application is.

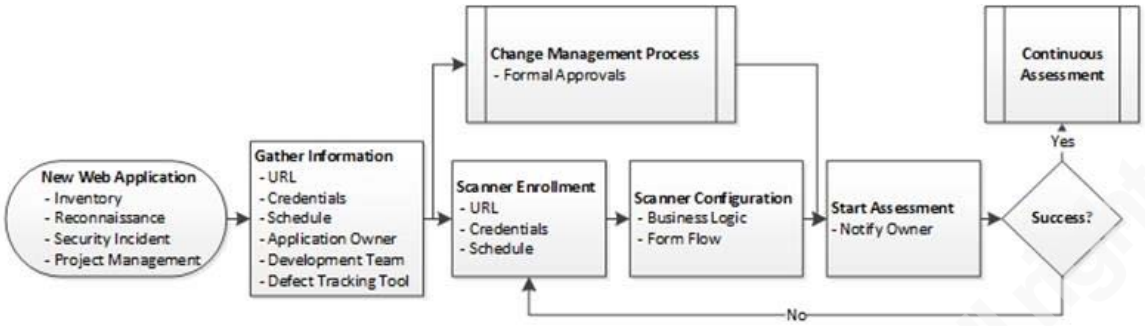


Figure 7. Enrollment Process.

2.2.1. Information Gathering

After identifying an application for enrollment, there might be some basic information required if it is not already a part of the inventory. This will be used both to configure our tools, and later when we start fixing vulnerabilities.

- **URL**: This is the single most important piece of required information. The URL is the front door of a web application, and will be used to configure the DAST scanner.
- **Credentials**: Most web application will have some kind of access control and authentication mechanism. Authentication is the process of verification that an individual or an entity is who it claims to be. Authentication is commonly performed by submitting a user name or ID and one or more items of private information that only a given user should know such as a password. Since a DAST scanner cannot assess what it is unable to access, a set of credentials may be required if the application requires authentication.
- **Schedule**: Do everything possible to prevent scanning from interfering with the business functionality of the application. If peak hours of operation take the application to maximum capacity, then configure the scanner to perform assessments during off peak time. Work with the application's business owner to define an appropriate scanning schedule.
- **Application Owner and Development Team**: Application vulnerability management requires constant communication with the people responsible

for the application. Reach out to them early in the process, and keep them informed about progress. Before an assessment it may be helpful to talk to the development team to get technical details about an application.

- **Defect Tracking Tool**: If the application was developed internally, the development team may already have processes and tools for fixing application bugs. A security vulnerability is essentially a kind of application defect. Using the tools that are already in place will allow for the greatest chance of successfully remediating vulnerabilities with the least disruption to other things the development team is already doing.

2.2.2. DAST Tool Configuration

With all of this information in hand, create an entry for the application in our scanning tool and configure it appropriately. How this works will be specific to the tool being used.

To maximize scanner coverage, it is important to take the application's business logic into account. There might be parts of the application that are only accessible under certain circumstances. For example, an ecommerce checkout page might only be available after adding items to a shopping cart. The scanner has to do things in a particular order to test the checkout functionality. Be sure to know how the DAST tool accounts for business logic, and configure it accordingly.

2.2.3. Approval and Notification

The vulnerability management program will suffer without getting buy off from application owners, and letting them know when their application is going to be scanned. Being in the situation of causing an impact to an application that the business owner did not know was being scanned can cause unneeded political problems. Make absolutely sure that application stakeholders are aware of the vulnerability management program's purpose, their role in it, and are notified when their application is being assessed.

Most mature IT organizations should have some sort of change management process. If one exists, try to leverage this process to get formal, documented approval for application assessments.

2.2.4. Continuous Assessment

Acquiring approvals and getting the DAST tool configured can be done in parallel. After those are complete, it is time to start the application assessment. During the first iteration, monitor the application closely. If there is a negative impact to the operation of the application, stop the scan. It may be necessary to reconfigure the scanner to ignore application functionality it does not work well with, or to alter the scanning schedule. After continuous assessments are under way, they are part of normal, business-as-usual security operations.

2.3. Reporting and Remediation

Once continuous assessments are running, the DAST scanner will find vulnerabilities. Depending on the accuracy of tool, there will be some amount of false positives. That is, the DAST tool will say that a vulnerability exists some place that it does not. The security team needs to conduct analysis of the findings to ensure they are not reporting false positives to application teams. This is especially important early in the program when there is a need to establish credibility.

The biggest difference between infrastructure and application vulnerability management is the level of effort required for remediation. For a vulnerability on a Windows server, the fix is usually to apply a patch. However, for a vulnerability in a web application, the development team has to create a patch. The development team may have to take time away from building new functionality that customers want. This needs to be taken into consideration when doing the risk analysis described below. Two developers spending a week fixing code is likely more expensive than the ten minutes a server administrator might spend downloading and installing a Windows update. Further, what if the vulnerability is in a legacy application that is not actively being developed? What if no one in the company writes code in the language in which the application was written? There might be need to pull developers off other projects, get outside help, or get creative with a new countermeasure.

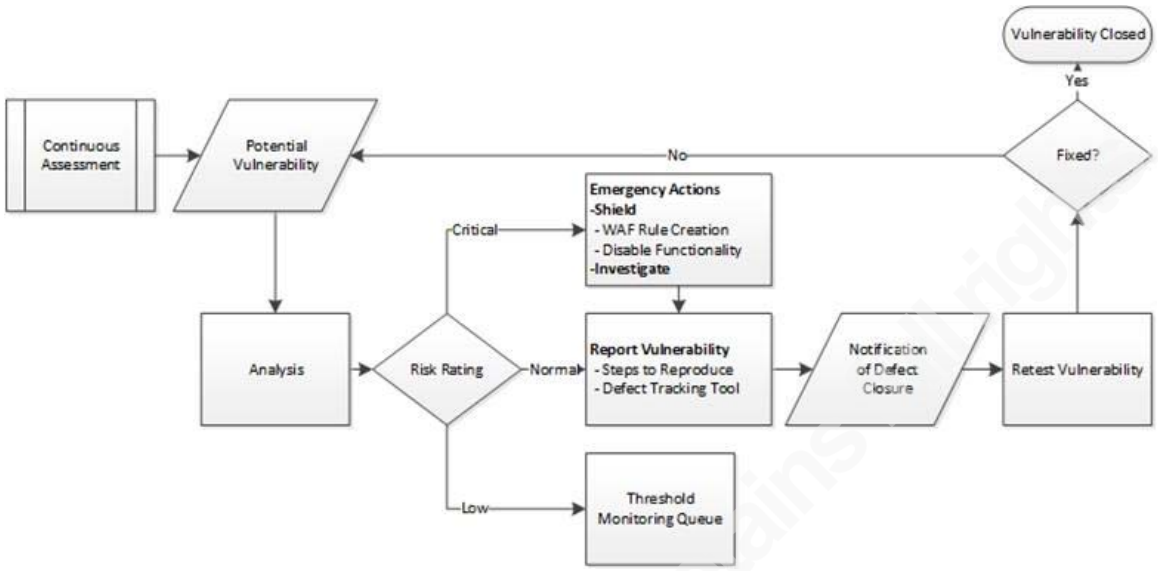


Figure 8. Remediation Process.

2.3.1. Risk Analysis

Determining the risk resulting from a vulnerability to prioritize and plan a course of action is one of the most important, but most overlooked, parts of the process. DAST tools typically provide some kind of criticality score, and most vulnerability management programs are happy to use it to prioritize remediation. However, the tool by itself cannot take the value of an asset into account, and will not know what countermeasures are in place. All of this information is used to determine the risk, and should be weighed against the level of effort that will be required for remediation. It might not make sense to spend $5,000 in development effort to protect $500 worth of assets.

$$Risk = \frac{Threat \times Vulnerability}{Countermeasures} \times Value$$

There are several mature risk analysis methodologies, and a vulnerability management program should use whichever one best fits its requirements. To streamline remediation efforts, map risk ratings to a couple different workflows that fit into your current security operations and software defect processes. After determining the risk resulting from a vulnerability, these categories might be useful for planning remediation

efforts: normal, critical, and low. A normal risk finding follows a standard remediation process. For a critical risk, remediation actions are expedited and might require a temporary countermeasure such as creating a web application firewall rule or disabling the vulnerable functionality. Low risk findings are deemed not worth remediating at this time. However, these should be tracked, monitored, and revisited in case the threat vector evolves changing the level of risk.

2.3.2. Reporting

Developer's knowledge of security and secure coding practices can vary. Every vulnerability report is an educational opportunity. Most DAST tools provide decent verbiage describing the issue and general ways to fix it, but they do not do a very good job of taking the specific application into account. The vulnerability report should have general verbiage describing the vulnerability, include steps required to recreate the issue, screenshots of the affected functionality of the application, HTML or code snippets, and programming language specific remediation advice. If applicable, the report should link to applicable corporate secure coding standings and web based training. Providing this level of detail allows a developer to zero in on the problem, and review the related material if necessary.

While infrastructure support teams have spent years doing patch management, developers have spent years fixing application defects. This is where knowing what defect tracking system an application's development team uses comes into play. Frame the vulnerability as a software defect, log it into the application's defect tracking system, and assign it to a team or individual developer. This helps eliminate bystander apathy. Rather than exporting a PDF report from the DAST tool and emailing it to the entire team, assigning a vulnerability to a specific developer assigns responsibility for remediation. An email to everyone on the team can be ignored by everyone on the team, but assigning a specific person holds someone accountable and will have a higher chance of success.

2.3.3. Validate Remediation

After the vulnerability has been remediated, retest it to make sure it was properly fixed. Only after the vulnerability remediation is validated by the security team should

the issue be considered closed. If it has not been correctly fixed, it may require additional analysis of the finding and of the attempted remediation. Perhaps it was fixed enough to reduce the risk to an acceptable level. If it was not fixed, go back to the beginning of the remediation process and start again. If retesting proves the software no longer vulnerable, mark it closed in the applicable tools.

2.4. Metrics

To know how effective the vulnerability management program is and determine if it is having an impact on software security, it needs to be measured. Metrics allow an organization to understand the performance of their security organization and weigh the costs of security safeguards against their effectiveness.

According to Jaquith (2007) good metrics have 5 characteristics: they are consistently measured, cheap to gather, expressed as a number or percentage and using at least one unit of measure, and are contextually specific.

- **Consistently measured**. Anyone should be able to look at the data and come up with the same metric using a specific formula or method. Metrics that rely on subjective judgment are bad.
- **Cheap to Gather**. Metrics should to be computed at a frequency commensurate with the process's rate of change; it is ideal to analyze security effectiveness on a day-to-day or week-by-week basis. Automating metric generation is key.
- **Expressed as a cardinal number or percentage**, not with qualitative labels like high, medium, or low.
- **Expressed using at least one unit of measure**, such as defects, hours, or dollars. Adding a second dimension such as defects per applications or defects over time adds value.
- **Contextually specific**. The metric needs to be relevant enough to decision makers that they can take action. If no one cares, it is not worth gathering.

I have found these metrics useful, and employ the following examples for decision making, executive dashboards, and to show the success of my program.

2.4.1. Security Testing Coverage

Security testing coverage tracks the percentage of applications in the organization that have been subjected to security testing. That is, out of all of the web applications the organization uses, how many of them are enrolled in the vulnerability management program? This is expressed as a percentage.

$$STC = \frac{\text{count (of enrolled applications)}}{\text{count (deployed applications)}} * 100$$

2.4.2. Mean-Time to Mitigate Vulnerabilities

Mean-time to mitigate vulnerabilities measures the average amount of time required to remediate an identified vulnerability. This is a measure of the organization's or development team's performance. It also captures how long, on average, the window is in which a vulnerability can be exploited. This is expressed in a number of days.

$$MTTMV = \frac{\sum \quad (\text{Date of Mitigation} - \text{Date of Detection})}{\text{Count (Mitigated Vulnerabilities)}}$$

2.4.3. Top 10 Vulnerabilities

OWASP maintains a list of top ten vulnerabilities to raise awareness about application security. While this could be a starting point for application security training, it would be more helpful to know what an organization's specific pain points are. With this, the security team can create custom training modules based on what the organization's development teams are having trouble coding. This can be visually represented with a pie chart to easily see how much of an issue the vulnerability is compared to others. Below is a Top 10 Vulnerabilities chart.

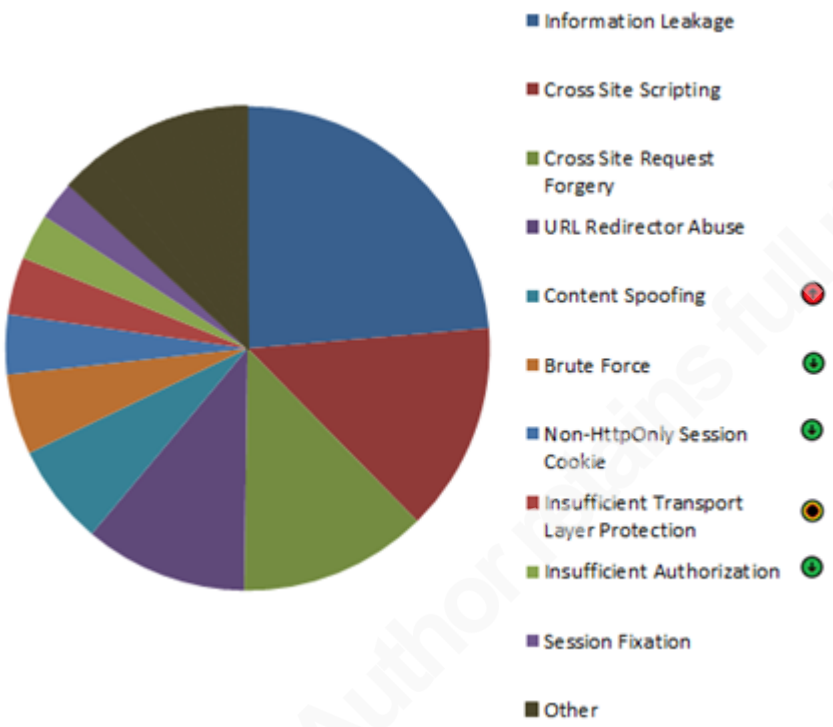


Figure 9. Top 10 Vulnerabilities.

It is a good idea to keep an eye on changes to the top ten vulnerabilities within one's organization. For example, encryption configuration issues appeared on this top ten list as new. Seeing "Insufficient Transport Layer Protection" on the list for the first time is an indication that something might be going on that warrants further investigation. SSL/TLS industry best practices were updated the month prior (Ristić, 2013). Earlier that year, weaknesses were discovered in RC4 cipher suites. Because of the Snowden leaks, it is suspected that the NSA can crack RC4 (Leyden, 2013). When there are major changes in the top ten list of vulnerabilities, it may be good time to revisit security policies and see if anything needs to change.

2.4.4. Vulnerabilities per Application

Vulnerabilities per application is a count of the number of vulnerabilities found in a specific application. This is expressed as a cardinal number. This can be visually represented with a bar chart to easily see how different applications compare against each other. Below is a Top 10 Vulnerable Applications chart.

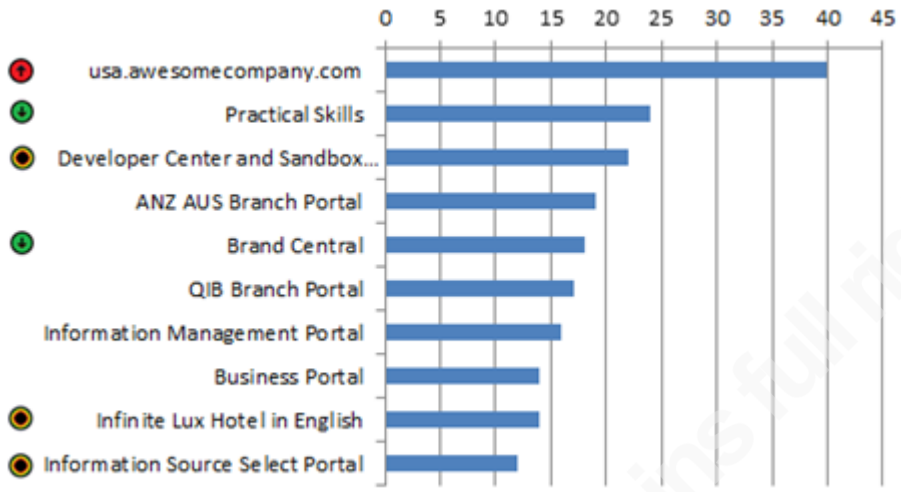


Figure 10. Top 10 Vulnerable Applications.

Having a top ten vulnerable applications list makes it apparent what applications on which to focus remediation efforts. It points security to the development teams responsible for these applications targeting them with the training developed from the top ten vulnerabilities list.

This is another place keeping track of changes is useful. If an application drops in rank, the application team is likely doing a good job fixing issues. If an application goes up on the list, reprioritize and act accordingly. If a new application shows up on the list, it may be an application that is being tested for the first time and presents a high risk to the organization. Report the vulnerabilities to the appropriate team and start working with them on remediation.

2.4.5. Risk Ratings – Defect Counts & Trending

Another important metric is a defect count by risk rating. This can be represented as a point in time with a bar chart, and trended over time with a line graph. Look at the trending of these defect counts along with the total number of applications being assessed to see how the trend is related to the amount of applications enrolled in the program. With this, it is easy to see how much risk exists in the environment, and get a feeling for how effective the program is.

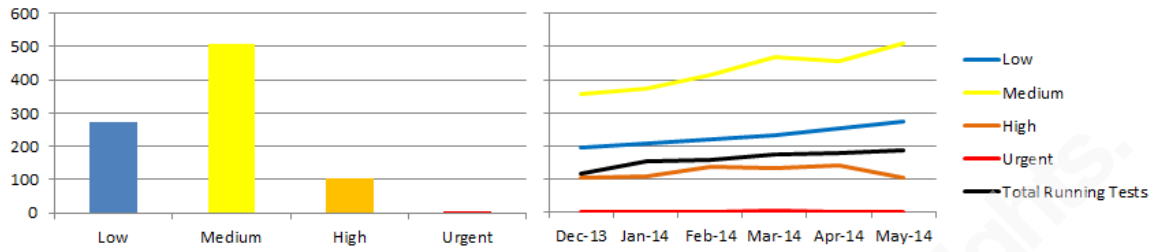


Figure 11. Vulnerability Risk Ratings.

3. Conclusion

The web application vulnerability management framework represents the next logical step in application security. In the era of the Internet, Web applications are critical to conducting business. Attackers know this, and are focused on exploiting vulnerabilities for their own gain. Information security professional must apply the rigor of a vulnerability management process to discovering and remediating those vulnerabilities before they are taken advantage of and their companies suffer loss.

As threats evolve and new attack vectors are discovered, it is essential to test applications to see how they are affected. Web application vulnerability assessments need to be continuous. This framework introduces a methodology, processes, and activities to achieve that goal.

The framework provides a holistic view of a web application vulnerability management program. Along three domains; governance, verification, and construction; it covers the prerequisites of web application vulnerability management, enrolling applications in the program, the remediation process used when a vulnerability is found, and how to measure the program using data visualization and metrics to determine whether or not it is successful.

Jason Pubal jpubal@mastersprogram.sans.edu

4. References

Chandra, P. (2009). *Software assurance maturity model*. Retrieved April 29, 2014, http://www.opensamm.org/downloads/SAMM-1.0-en_US.pdf

Foreman, P. (2010). *Vulnerability management*. Boca Raton: CRC Press.

Harris, S. (2012). *CISSP all-in-one exam guide*. Berkeley, Calif: Osborne

Jaquith, A. (2007). *Security metrics: Replacing fear, uncertainty, and doubt*. Upper Saddle River, NJ: Addison-Wesley.

Leyden, J. (2013, September). *That earth-shattering NSA crypto-cracking: Have spooks smashed RC4*? Retrieved November 2, 2013 from http://www.theregister.co.uk/2013/09/06/nsa_cryptobreaking_bullrun_analysis/

MacDonald, N., & Feiman J. (2013, July). *Magic quadrant for application security testing*. Retrieved March 25, 2014, from http://www.veracode.com/sites/default/files/Resources/AnalystReports/gartner-ast-magic-quadrant-report-2013.pdf

McGraw, G., Migues, S., & West, J. (2013). *Building security in maturity model*. Retrieved April 29, 2014, from http://bsimm.com/download/dl.php

Nicolett, M. (2005, March). *How to develop an effective vulnerability management process*. Retrieved March 23, 2014, from http://www85.homepage.villanova.edu/timothy.ay/DIT2160/IdMgt/how_to_develop_.pdf

Ristic, I. (2013, September) *SSL/TLS deployment best practices*. Retrieved November 2, 2013, from

Jason Pubal jpubal@mastersprogram.sans.edu

https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices_1.3.pdf

Sherwood, J., Clark, A., & Lynas, D. (2005). *Enterprise security architecture: A business-driven approach*. San Francisco: CMP Books.

Wheeler, E. (2011). *Security risk management: Building an information security risk management program from the ground up*. Waltham, MA: Syngress.

Williams, J., & Wichers, D. (2013). *OWASP Top 10*. Retrieved March 21, 2014, from http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf

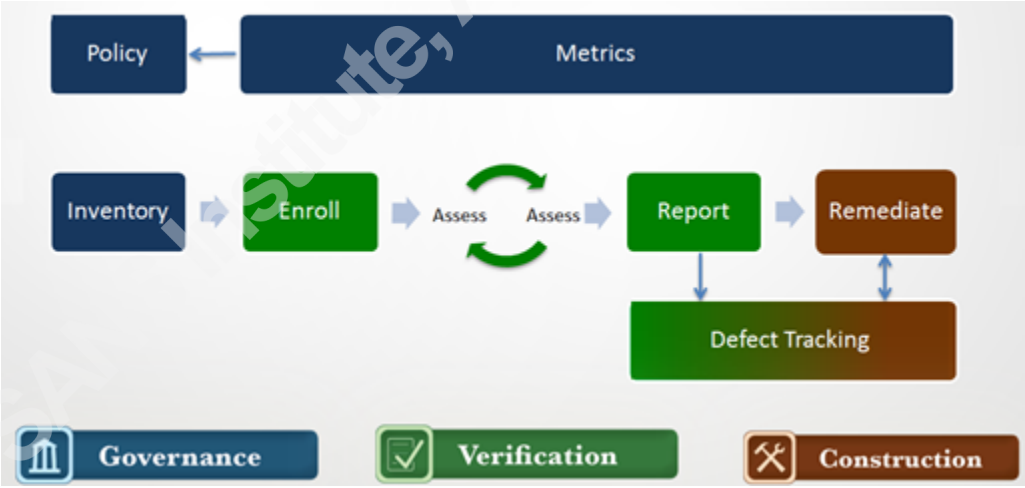
Verizon. (2014) *2014 Data breach investigations report*. Retrieved May 20, 2104, from http://www.verizonenterprise.com/DBIR/2014/reports/rp_Verizon-DBIR-2014_en_xg.pdf

5. Appendix I: The Web Application Vulnerability Management Framework

The web application vulnerability management framework represents the next logical step in application security. Mature application security programs perform security testing against web application as they are being developed, but tend to be more reactive to security defects once an application has been deployed. Infrastructure vulnerability management performs recurring vulnerability scanning against production network and servers, but the web applications hosted on them are ignored.

As threats evolve and new attack vectors are discovered, applications need to be tested to see how they are affected. Application vulnerability management needs the same rigor infrastructure vulnerability management has; web application vulnerability assessments need to be continuous. This framework introduces a methodology, processes, and activities to achieve that goal.

The web application vulnerability management framework has three domains: governance, verification, and construction. Each of these has a number of related practices, defined below.



5.1. Governance

Governance focuses on the processes and activities related to how an organization manages software development and application security.

Jason Pubal jpubal@mastersprogram.sans.edu

5.1.1. Policy

Security policy communicates the intentions of management for managing risk and enforcing security in the organization. It is the document that empowers the security team and gives them responsibility to perform application security.

It outlines secure coding practices developers use to create software and the security requirements for commercial off the shelf (COTS) applications. These are the standards to be assessed against. When we find a vulnerability, the documented secure coding practices are referenced to give guidance to teams that need to fix the issue.

Policy should contain remediation timelines. This tells various teams how long they have to get security vulnerabilities fixed.

5.1.2. Inventory

Asset management is a prerequisite for vulnerability management. Before you can secure stuff, you need to know what stuff you have. It is helpful to know how many applications there are, where they are located both physically and logically on the network, who manages them, and what their value is.

Discovery is one of the first steps of the vulnerability management process, and is done through a combination of automated and manual processes. Discovery needs be recurring, so an updated inventory is maintained.

5.1.3. Metrics

To know how effective the vulnerability management program is and determine if it is having an impact on software security, it needs to be measured. Metrics allow an organization to understand the performance of their security organization and weigh the costs of security safeguards against their effectiveness.

Metrics can inform policy. Data visualization, trending, and the high level view achieved through metrics allows management to make better decisions and shows us where our application security policy can be improved.

5.2. Verification

Verification is focused on the processes and activities related to how an organization checks and tests software. This is where our security assessments take place.

5.2.1. Enrollment

Enrollment has two activities. This is where we configure our dynamic application security testing (DAST) tool to assess an application. If the required information is not in our inventory, we may need to work with application owners to get it.

Jason Pubal jpubal@mastersprogram.sans.edu

Enrollment also includes notification and approval for security testing from stakeholders. Leverage your companies change management process if there is one already established.

5.2.2. Assessment

Continuous assessment is the core of the web application vulnerability management framework. This is where our DAST tool tests each web application on a continuous or frequently recurring basis as part of normal, business-as-usual operations.

5.2.3. Reporting

Reporting consists of two parts: a report detailing the vulnerability, and leveraging the development teams defect tracking system for assignment of responsibility and tracking.

The vulnerability report should have general verbiage describing the vulnerability, include steps required to recreate the issue, screenshots of the affected functionality of the application, HTML or code snippets, and programming language specific remediation advice. If applicable, the report should link to applicable corporate secure coding standings and web based training. Providing this level of detail allows a developer to zero in on the problem, and review the related material if necessary.

Frame the vulnerability as a software defect, log it into the application's defect tracking system, and assign it to a team or developer.

5.3. Construction

Remediation of an application vulnerability requires software development. While this is mostly done outside of the security organization, it is important to realize that each remediation effort can be seen as its own software development project. Construction concerns the processes and activities related to how an organization creates software.

5.3.1. Defect Tracking

Once reporting logs the vulnerability in the application team's defect tracking system, both the application team and security team can use it for updates, tracking, and reporting on progress. Depending how many application teams the security organization has to interface with, it is possible to need to keep track of several disparate defect tracking system. The security team may need to maintain its own vulnerability tracking database that links to each teams defect tracking system.

5.3.2. Remediation

The hands on coding portion is done by the application team. Depending on that team's level of security knowledge, they need advice or training. After the security fix has been completed, the security team verifies that it properly addresses the vulnerability finding. This step often requires retesting with the DAST tool, along with additional manual testing. In the event that it is not completely fixed, go back to

the analysis step of the remediation process to assess the best way to go forward. Only after this verification should the vulnerability be considered remediated and closed out in the appropriate defect tracking tools.

Jason Pubal jpubal@mastersprogram.sans.edu

6. Appendix II: Responsibility Assignment Matrix (RACI)

This RACI breaks down the roles and responsibilities involved in the web application vulnerability management framework. A description of the roles can be found below. Multiples roles may be filled by a single team or individual.

		ROLES					
Process	**Activity**	**Application Business Owner**	**Application Technical Contact**	**Application Development Group**	**Application Vulnerability Management Team**	**Secure SDLC Team**	**Governance, Risk, & Compliance Team**
Application Enrollment	Gather Required Application Information	C	C	C	R, A	C	C
	Configure DAST Tool				R, A		
	Approval	C	C	C	R, A		
	Start Testing	I	I	I	R, A	I	
Continuous Assessment	Vulnerability Scanning				R, A		
Vulnerability Reporting	Analyze Findings				R, A		
	Apply Risk Rating				A		R
	Report / Notify	I	I	I	R, A	I	I
Remediation	Prioritize	A			C, I	C	R
	Develop Fix	A		R	C, I	C	
	Apply Fix	A	R		C, I		
	Retest	A			R		
	Close Finding	A			R		

6.1. Application Business Owner

Applications serve some business function, and have a business owner who is primary accountable for the application. Information about the application, and formal approval for testing comes from its business owner. When a vulnerability is found, the business owner is accountable for remediation.

Jason Pubal jpubal@mastersprogram.sans.edu

6.2. Application Technical Contact

Another person may be more technically knowledgeable about the application. This person is informed and consulted during much of the process, and is responsible for deploying the application's fix after it is developed.

6.3. Application Development Team

If the application was developed in-house, its development team can be consulted for information along the way. When a vulnerability is found, they are responsible for creating the fix.

6.4. Web Application Vulnerability Management Team

This is the core security team involved in the framework. They are responsible and accountable for a large portion of the effort.

6.5. Secure SDLC Team

Security during the software development lifecycle (SDLC) is outside the scope of the vulnerability management framework. In the event that a different security team is responsible for security in the SDLC, this is that team.

6.6. Governance, Risk, & Compliance

The Governance, Risk, and Compliance (GRC) team is focused on the company's overall IT risk strategy, and responsible for performing risk assessments.

Last Updated: June 23rd, 2017

Upcoming SANS App Sec Training

SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
Community SANS Minneapolis DEV534	Minneapolis, MN	Aug 25, 2017 - Aug 28, 2017	Community SANS
Community SANS San Francisco DEV541	San Francisco, CA	Aug 28, 2017 - Aug 31, 2017	Community SANS
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Secure DevOps Summit & Training	Denver, CO	Oct 10, 2017 - Oct 17, 2017	Live Event
SANS Seattle 2017	Seattle, WA	Oct 30, 2017 - Nov 04, 2017	Live Event
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Cyber Defense Initiative 2017 - DEV522: Defending Web Applications Security Essentials	Washington, DC	Dec 14, 2017 - Dec 19, 2017	vLive
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced