Jumpstart a Web Application Secure Coding Program: A Five Step Process

SANS STI Application Security Written Assignment

Author: Jim Beechey, beechey@northwood.edu
Advisor: Johannes Ulrich

Abstract

Web application security has been top of mind for information security professionals for some time. Web application vulnerabilities are some of the most common ways for an organization to become compromised. In addition, many of these issues have been around for 10 or more years. Why then as an industry do we continue to struggle is this area? The challenge many face is translating the technical realities of these vulnerabilities into a secure coding program capable of educating developers on the issues, providing tools to make secure coding easier, measuring progress over time and flexibility to adapt as threats change. More pointedly, most struggle with where to even start. Secure coding programs require momentum to be successful and these five steps can help jump start any organization's program.

1. Introduction

Web application security continues to be one of the top areas of concern for information security professionals. “The IT practitioners in our study agree that it is important to reduce the risks caused by these threats. According to the findings, 74 percent of respondents believe Web application security is either more critical or equally critical to other security issues faced by their organizations... IT practitioners recognize attacks can be costly due to the potential for the loss of sensitive data, fines due to noncompliance with regulations and business disruption.” ("State of web," 2011). While this is not surprising to most, it certainly is cause for concern given the length of time some of the most basic web application security vulnerabilities have been around. SQL injection, for instance, has roots back to 1998. (Clarke, 2012) Yet, over 10 years later, the 2011 Verizon Data Breach Investigations Report still ranks SQL injection as one of the top Threat Action Types accounting for 54 breaches and 933,157 compromised records (Baker, Hutton, Hylender, Parmula, Porter & Spitler, 2011).

The question which must be asked is why the industry has not come further when it comes to web application security. While there are many debates, clearly information security professionals struggle to put into place an overarching program to combat these issues. Part of the challenge is information security teams cannot fix this issue alone. Web application security programs require collaboration and dedication across several areas of the organization. “Several departments should or maybe involved with owning components of the application security program, such as Risk Management & Compliance, Internal Audit, Project Management, Development Teams, Security Group, and Systems & Infrastructure to name a few.” (Carney, 2007) Another struggle is the politics involved in such an endeavor. In addition, there is no technology silver bullet to web application security. Finally, web application security issues cannot be resolved by tools and technology alone.

In order to have a successful web application security program, leaders must find a way to get the effort off to a strong start. Early momentum is critical in situations where many groups are involved and the political stakes are high. Therefore, having a plan to get the project off to a good start is paramount. Web applications security programs should focus on five keys areas when kicking off a program: Leading Change, Organizational Support, Inventory, Training, and Technical Tools.

2. Leading Change

Simply put, change is hard. Creating an application security program and truly making secure coding practices part of the daily lives of developers is a significant organizational change. Having an understanding of how to lead change will help significantly along the journey.

When thinking about change and how to influence or lead change, most research agrees that there are three general areas of human behavior to focus on. First, people tend to have a rational side to their personality in which they use reason and analysis to make decisions. Second, people have an emotional side which their emotions influence their decision regardless of what their rational side may think. Finally, the environmental factors surrounding various decisions play a major role in people decision making. “For things to change, somebody somewhere has to start acting differently. Maybe it’s you, maybe it’s your team. Picture that person (or people). Each has an emotional Elephant side and a rational Rider side. You’ve got to reach both. And you’ve also go to clear the way for them to succeed. In short, you must do three things: Direct the Rider, Motivate the Elephant and Shape the Path.” (Heath, 2010) The most important point to remember is that in order to lead or influence change, one’s plan should ideally include components of all three areas: rational, emotional and environmental. Different situations and more importantly, different people require different motivational tactics. If a plan for change can reach each of these three areas, then the plan is much more likely to be applicable to a wide range of people and situations.

For example, during the implementation of the web application security program there are several things that can be done to hit each of these three focus areas. Consider the impact of delivering news of a data breach or successful penetration test due to a web application vulnerability. This would certainly hit the emotional side of developers and management. These extreme examples can help motivate people to action. At the same time, having detailed plans, accessible training and re-usable code samples will help people's rational side to feel like they really can accomplish the task at hand. Finally, removing competing deadlines from other projects and budget concerns over training are examples of ways to impact change through environmental factors.

Ultimately, implementing a web application security program is a significant change and requires a plan to properly lead the change effort. Using the principles above the information security leader can ensure that change efforts are meeting the needs of the various parties involved in the project.

3. Organizational Support

3.1. Executive Sponsorship

Organizational support for a web application security program is critical. Having the proper executive sponsorship and key stakeholder buy-in is one of the most important first steps before officially kicking off the effort.

Certainly, any new project or initiative can be helped by the proper executive sponsor; however this is especially true for web application security efforts. "The role of the executive sponsor is to shepherd the project through the organization throughout its lifecycle. Most importantly, the executive sponsor must foster a culture of communication, direction, teamwork, and excitement among the many groups that must work together to ensure the project is a success." (Lynch & Roecker, 2007) Given the cross functional nature of web application security programs and political challenges the team may face, choose your sponsor wisely. Make sure he or she is prepared to meet the challenges and potential time commitment needed to make the project successful. "If you don't have executive level sponsorship, and work with -- not against -- the development team: your secure web application initiative lost the race before the starting gun was fired." (Hulme, 2009)

3.2. Subject Matter Expert Advocates

Another important piece in gaining organizational support is to have a few key subject matter experts behind the program prior to kick off. Certainly the information security team will be supportive, but who else can be an advocate for the effort? Try to find a developer or two in each area who is security minded and understands the importance of the effort. “Few people on a development staff will ever care about security. For those who show any sort of passion (but no more than one person per product; and one QA person for every four products), we think it is worth having that person take the lead in doing research and making recommendations when it comes to security features. If you pick good self-starters and give them responsibility, you may get a bunch of extra work out of them.” (Viega, 2009) Another avenue worth pursuing is the quality assurance group. While QA historically has focused on other areas of software quality, they likely will agree that minimizing security defects is also part of their charter.

3.3. Vendor Management

Secure coding practices are not limited to internal resources only. Proper vendor management is an additional component that must be considered. In fact, developing secure coding requirements for vendor partners can be a good quick win for a new program as there are several publicly available resources to help develop requirements and often there is less political pushback than putting requirements on internal teams.

When starting to write vendor requirements, work with the company’s contracts and governance groups to ensure that, once developed, the requirements can be easily included in all requests for proposal and vendor contracts. Before working on organizational requirements start with one of publicly available samples from other organizations. The SANS Application Security Procurement Language can be found at www.sans.org/appseccontract. The OWASP Secure Software Contract Annex can be found at www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex. These two resources are a great starting point to writing requirements for any organization.

4. Application Inventory

There is no stronger tenant to information security then “know thy system”. In order to properly secure something one must know what they have first. For instance, the SANS 20 Controls for Effective Cyber Defense includes two controls completely focused on inventory; Critical Control 1 – Inventory of Authorized and Unauthorized Devices and Critical Control 2 – Inventory of Authorized and Unauthorized Software. Web application security is no different and the first step in any program should be an inventory of an organizations environment. “The starting block of an application security initiative is to complete an inventory of all applications within the enterprise. Compile a spreadsheet of the number of applications, type of applications, middle-tier software, and database technologies that exist within all facets and business units of the organization. By understanding the business purposes and information/data that is flowing through these applications, you can start developing protection strategies and standards that will secure your organization’s most critical data”. (Carney, 2007) Once the inventory is obtained, applications should also be ranked based upon potential risk to the company. Factors such as the sensitivity of the data, the business impact of downtime and potential exposure of a data breach should be considered.

5. Training

Training is one of the more obviously needs of a secure coding program. While no one can argue some level of training is needed, finding the right balance and type of training is challenging. Most large organizations have many developers with varying experience and focus areas. Therefore, simply writing a big check to send all developers to a secure coding class is not going to work. Training must be planned, discussed and tailored to various groups.

Formal, classroom training attracts leaders because of the desire to outsource the problem and provide a quick fix even if the cost is high. The challenge is many developers have no interest in learning secure coding principles and the organizations is likely to gain limited value from this approach. “When you consider the direct cost and lost productivity, it typically costs about $1,000 a day to train a developer. It’s well established that classroom retention rates (without lab work) are generally around 50%, even immediately after the class. We have found that most developers don’t care much about security and consider even basic software security stuff “very complex” (which it can be). Therefore, retention rates are probably even lower than that. We’ve seen some data indicating that developers, on average, will have forgotten over 90% of secure programming training after six months. We think these numbers are about right. It’s not worth it. Just train the ones who are excited about learning – they’re the only ones likely to do a good job for you anyway!” (Viega, 2009) Therefore, organizations should focus their training dollars on a smaller subset of developers who have a desire to learn secure coding principles and make them subject matter experts.

However, training should not be limited to formal, classroom training. There are numerous other methods for creating at least some level of awareness around secure coding. Organizations should strive to make all developers aware of the most common web applications vulnerabilities such as SQL injection and cross site scripting. All developers may not be able to describe each in depth or fix broken code, but each should have the most basic understanding of what the issue is. This can be done in a variety of ways including the creation of short, video tutorials and quick 5 minute walk-through at staff meetings.

Another option is to highlight application security events in the news such as the recent SQL injection attack against Yahoo Voice. Plaintext passwords for 453,492 Yahoo Voices accounts were compromised due to a successful SQL injection attack. (Schwartz, 2012) Relating activities at work to recent news events can help greatly in re-enforcing the issue and aid in retention.

Finally, share the organization's penetration testing results and any incidents involving web application security. Some may consider these results confidential and have the natural inclination to not want to share negative news. However, providing developers with real-world examples from internally developed applications has a much greater chance of getting their attention than a theoretical vulnerability discussed in a classroom setting. Also, sharing the results helps get the most value out of a penetration test. "Perhaps the most common failure of the software penetration testing process is failure to identify lessons learned and propagate them back into the organization… Mitigation strategy is thus a critical aspect of any penetration test. Rather than simply fixing only those issues identified, developers should carry out a root-cause analysis of the identified vulnerabilities. For example, if a majority of vulnerabilities are buffer overflows, the development organization should determine just how these bugs made it into the code base. In such a scenario, lack of developer training, misapplication (or nonexistence of) standard coding practices, poor choice of languages and libraries, intense schedule pressure, failure to use a source code analysis tool, or any combination thereof may ultimately represent an important cause." (Jones, 2006)

6. Technical Tools

After dealing with several non-technical issues, ultimately any web application security program will have to rely on some technical tools and services in order to find and remediate existing vulnerabilities. The important key to remember is that technical tools are not the only piece of a solid program, in fact they should be the last item of focus when jumpstarting the program. When first starting the program, think of technical tools in two general categories: detection and remediation technologies.

6.1. Detection Technologies and Services

Detection technologies are designed to find vulnerabilities in existing systems. Each has different positives and negatives, but the reality is that most mature programs will employ some examples of each. The most common examples of detection technologies include: automated code analysis - dynamic, automated code analysis - static and penetration testing.

6.1.1. Automated Code Analysis – Dynamic

Dynamic Analysis is a simple and popular method for automating web application testing. Dynamic Analysis treats the application as a black box. Dynamic analysis involves crawling an application and testing inputs and outputs for security vulnerabilities. (Glynn, 2012) Dynamic analysis is the likely first choice for most web application security programs. There a many commercial and open source tools capable of crawling and scanning a web site for the most common vulnerabilities. This is a great first place to start in order to detect the most obvious problems in an organization's web applications. A list of commercial and open source web application security scanners can be found in Appendix A. Organizations without existing tools should consider utilizing an open source tool or software as a service model before investing in more robust solutions. Also, many vulnerability assessment scanners also include web scanning modules so these tools should also be considered if already installed in an organization.

6.1.2. Automated Code Analysis – Static

Static analysis is a software testing technique that allows for testing of an application without actually running the program. Static analysis looks at the actual code used to create an application. Static analysis is nice because it does not require a large environment to test an application and can also be performed early in the development lifecycle. (Glynn, 2012) Static code analysis has several advantages. These tools scale much better than their dynamic counterparts and are very good at finding vulnerabilities that can be automated such as injection flaws. Alternatively, these tools are not very good at finding vulnerabilities which cannot be easily found through automated methods such as authentication problems and business logic flaws.

6.1.3. Penetration Testing

Penetration testing is the third option for detecting web application vulnerabilities. Many organizations first begin to test web applications through a formal penetration test. Web application penetration testing involves hiring a third party company or individual to attempt to break into an organization's web applications. Penetration testing offers two major advantages over using dynamic and static tools. First, penetration tests actually exploit vulnerabilities found in web applications and, therefore, avoid arguments about whether a flaw is real or not. Second, penetration tests are often done by outsiders and, therefore, have the advantage of an external opinion which is sometimes more valued by upper management. Regardless of the motivations, penetration testing web applications is a critical piece to an overarching program.

6.2. Remediation Technologies and Services

Remediating web application security vulnerabilities is where many programs struggle. The information security team is typically skilled at finding web application vulnerabilities, but struggles helping remediate issues. Often the security team lacks that skills to help developers fix code issues or simply feels that coding is not their responsibility. Worse yet, some teams simply hand a large vulnerability report to web developers and move on. Often the developers believe these are security issues that should be fixed by the security team. A successful program requires tools, documentation and canned solutions to regularly occurring security issues. Most importantly, a successful program brings the security and development teams together to work these issues in tandem.

There are so many possible issues with web applications it can be hard for teams to know where to begin. In order to jumpstart a secure coding program, the security team should focus efforts on one or two of the most common and most impacting issues. A great list to start with is the OWASP Top Ten list. Injection vulnerabilities and cross site scripting are two suggested issues to get started with. These issues are number 1 and 2 on the OWASP Top Ten list and two of the more common and impacting web application vulnerabilities. ("OWASP top 10 for 2010," 2010) Once the program has successfully dealt with these two issues, pick two more and iteratively move down the list. Having this kind of an approach avoids overwhelming the team and allows the program to show value more quickly.

Ideally, the information security team will work with a select group of developers to create standard operating procedures to guide development efforts. These operating procedures will define how to handle various issues in the OWASP top ten. The idea is not simply to tell developers that they have vulnerabilities in their code, but to create simple processes to prevent future vulnerabilities from occurring. The examples below show possible solutions for two of the more common web application security issues, however organizations need to investigate these solutions on their own to find the best fit. Not all organizations develop in the same languages or follow similar processes, therefore a one size fits all solution does not exist.

6.2.1. Injection Vulnerabilities

Injection vulnerabilities are some of the more common and serious issues facing web applications. "Injection flaws, such as SQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data." ("OWASP top 10 for 2010", 2010) According to the OWASP Top Ten project injection vulnerabilities can be prevented using three options. First, and most effective method, is to use an API which removes the need for an interpreter or has a parameterized API. Second, special characters should be escaped to avoid their use in queries. Third, whitelisting input validation is also effective. ("OWASP top 10 for 2010", 2010) OWASP ESAPI (Enterprise Security API) can help with both escaping of special characters and whitelisting input. The Apache commons validator page at http://commons.apache.org/validator/ is another good reference for whitelisting input.

6.2.2. Cross Site Scripting (XSS) Vulnerabilities

"XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation and escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites." (OWASP top 10 for 2010, 2010) Cross site scripting vulnerabilities, both stored and reflected, can be prevented by doing server side validation and escaping. This is most effectively done using an existing library such as OWASP's ESAPI or Microsoft's Anti Cross Site Scripting library. (OWASP, 2012) Creating standard procedures governing the use of these libraries can effectively eliminate cross site scripting vulnerabilities from newly developed web applications.

7. Conclusion

Jumpstarting a web application security program is not a simple task, but can be successful with proper planning, communication and execution. Information security teams will put themselves in the best position for success when they focus on the five areas discussed: Leading Change, Organizational Support, Application Inventory, Training and Technical Tools. Teams must resist the urge to jump into technical details without first addressing the other areas. In addition, web application security does not require the purchase of a seven figure software solution. Organizations that follow the principles outlined can create a highly effective program with minimal investment.

8. References

Baker, W., Hutton, A., Hylender, C., Parmula, J., Porter, C., & Spitler, M. (2011). 2011 verizon data breach investigations report. Retrieved from http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf

Carney, M. (2007, October 15). *Carney: How to create an effective application security program*. Retrieved from http://www.csoonline.com/article/216752/carney-how-to-create-an-effective-application-security-program-

Clarke, J. (2012). *Sql injection attacks and defense*. Waltham, MA: Syngress.

Glynn, F. (2012, May 12). *A ciso's guide to application security - part 4: Weighing appsec technology options*. Retrieved from http://threatpost.com/en_us/blogs/cisos-guide-application-security-part-4-weighing-appsec-technology-options-050712

Hulme, G. (2009, May 14). So, you want to build an effective application security program? how good are you at politics?. Retrieved from http://www.informationweek.com/news/security/229206326

Jones, A. (2006). *Software security: Building security in*. Addison-Wesley Professional.

Lynch, M., & Roecker, J. (2007). *Project managing e-learning*. New York, NY: Routledge.

OWASP. (2012, Febuary 23). *Xss (cross site scripting) prevention cheat sheet*. Retrieved from https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet

Owasp top 10 for 2010. (2010, June 01). Retrieved from https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project

Schwartz, M. (2012, July 12). *Yahoo hack leaks 453,000 voice passwords*. Retrieved from http://www.informationweek.com/news/security/attacks/240003587

Shura, B. (2010, November 28). *Web application security consortium / web application securiy scanner list*. Retrieved from http://projects.webappsec.org/w/page/13246988/Web Application Security Scanner List

State of web application security. (2011, February). Retrieved from https://www.barracudanetworks.com/ns/downloads/White_Papers/Barracuda_Web_App_Firewall_WP_Cenzic_Exec_Summary.pdf

Viega, J. (2009). *The myths of security: What the computer security industry doesn't want you to know*. Sebastopol, CA: O'Reilly Media Inc.