Building a Web Application Security Program

Version 1.0
Released: March 9, 2009

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog but has been enhanced, reviewed, and professionally edited.

This report is sponsored by Core Security Technologies, Inc, iMPERVA, and Qualys.

Special thanks to Chris Pepper for editing and content support.

Licensed by Core Security Technologies

Core Security Technologies is the leader in comprehensive security testing software solutions that IT executives rely on to expose vulnerabilities, measure operational risk and assure security effectiveness. The company's CORE IMPACT product family offers a comprehensive approach to assessing the security of network systems, endpoint systems, email users and web applications against complex threats. All CORE IMPACT security testing solutions are backed by trusted vulnerability research and leading-edge threat expertise from the company's Security Consulting Services, CoreLabs and Engineering groups. Based in Boston, MA and Buenos Aires, Argentina, Core Security Technologies can be reached at 617-399-6980 or on the Web at: www.coresecurity.com

Licensed by Imperva, Inc.

More organizations trust Imperva for the activity monitoring, real-time protection and risk management of their critical business data and applications than any other vendor. Imperva's award-winning application firewall and database security solutions provide full visibility and granular control over the usage of enterprise business data to more than 4,500 organizations worldwide. For more information, visit www.imperva.com.

Licensed by Qualys

Qualys, Inc. is the leading provider of on demand IT security risk and compliance management solutions – delivered as a service. Qualys' Software-as-a-Service solutions are deployed in a matter of hours anywhere in the world, providing customers an immediate and continuous view of their security and compliance postures.

The QualysGuard® service is used today by more than 3,500 organizations in 85 countries, including 40 of the Fortune Global 100 and performs more than 200 million IP audits per year. Qualys has the largest vulnerability management deployment in the world at a Fortune Global 50 company. For more information, visit www.qualys.com.

Contributors

The following individuals contributed significantly to this report through comments on the Securosis blog and follow-on review and conversations:

Marcin Wielgoszewski
Andre Gironda
Scott Klebe
Sharon Besser
Mike Andrews
ds

Copyright

This report is licensed under Creative Commons Attribution-Noncommercial-No Derivative Works 3.0.

http://creativecommons.org/licenses/by-nc-nd/3.0/us/

Table of Contents

Introduction 5

The Web Application Security Problem 5

The Business Justification 7

Web Applications Are Different 9

Why Web Application Security Is Different Than Host and Network Security 9

The Web Application Security Lifecycle 11

Secure Development 12

Secure Deployment 12

Secure Operation 13

Secure Development 14

Web Application Security: Training and the SDLC 14

Static Analysis Tools 15

Dynamic Analysis Tools 15

Secure Deployment 17

Vulnerability Assessment 17

Penetration Testing 18

Secure Operations 21

Web Application Firewalls (WAF) 21

Monitoring 22

WAF + VA 23

Putting It All Together 24

Large Enterprise with Customer Facing Web Applications 24

Mid-sized Firm and PCI Compliance 26

Internal Web Application Development 27

Conclusion 28

Who We Are 29

About the Authors 29

About Securosis 29

Introduction

Current web applications exist in an environment markedly different from the early days of businesses entering the Internet. They have become essential tools interconnecting organizations in ways never anticipated when the first web browsers were designed. These changes have occurred so rapidly that, in many ways, we've failed to adapt operational processes to meet current needs. This is particularly apparent with web application security, where although most organizations have some security controls in place, few organizations have comprehensive comprehensive web application security programs. This is problematic for two reasons. First, the lack of a complete program materially increases the chance of failure resulting in a loss-bearing security breach. Second, the lack of a coordinated program is likely to increase overall costs — not just losses from a breach, but the long term costs of maintaining an adequate security level (adequate being defined as meeting all compliance obligations and limiting losses to an acceptable level).

This report shows how to build a pragmatic web application security program that constrains costs while still providing effective security. Rather than digging into the specific details of any particular technology, we'll show you all the basic pieces and how to put them together. We start with some background on how web applications are different than traditional enterprise applications or commercial off-the-shelf products. Then we provide basic business justifications for investments in web application security you can use to gain management support (more detailed information can be found on the Securosis web site). We also focus of the particular security needs of web applications, and then delve into details of the major security components and how to pull them together into a complete program, with examples built around typical use cases.

The Web Application Security Problem

Enterprise web applications evolved in a way that has created a conundrum for security. Although we've always been aware of them, we initially treated them as low-risk endeavors almost fully under the control of the developers who created them. But before we knew it, they grew from experimental programs to critical business applications. Ask any web application developer and they can tell you the story of their small internal project that became an essential business application once they made the mistake of showing it off to a business unit or the outside world. We can break this general evolution down into some key trends which have led to the current security situation:

- Before web applications, very few businesses exposed their internal transactional systems to the outside world. Even those which did expose systems to business partners on a restricted basis rarely exposed them directly to customers.
- Web applications grew organically — starting from informational websites that were little more than online catalogs, through basic services, to robust online applications connected to critical back-end systems.
- Applications developed slowly over time, with increased functionality leading to increased reliance, often without the oversight they might have gotten had they been designed as massive projects from the beginning. This transition is the classic "frog in a frying pan" problem: Drop a frog into a hot frying pan, and it will hop right out. Slowly increase the heat, and it will fry to death without noticing or trying to escape.

- Web application protocols were designed to be lightweight and flexible; and lacked privacy, integrity, and security features.
- Web application development tools and techniques evolve rapidly, but we still rely on massive amounts of legacy code. Both internal and external systems, once deployed, migrate to new systems but only rarely are we able to clean up any of the existing complex interdependencies.
- Web application threats evolve as quickly as our applications, and apply to everything we've done in the past. We're constantly discovering entirely new classes of vulnerabilities we didn't anticipate before. The web itself was never designed to securely run mission critical applications, so we are building on a platform that was not intended to support its current uses.
- Few web applications are designed securely from the ground up, or maintained with processes designed to keep them secure over time.
- When security breaches do occur, they can be difficult to detect, analyze and measure.

In light of all these factors, web application security is typically underfunded in most organizations. And like all application development, web applications are subject to time pressures and deadlines that result in deployment tradeoffs and compromised functionality to meet business objectives. We categorize the web application security problems as follows:

- Few web applications were designed to be secure.
- The application is reliant on a remote web browser which is neither secure nor trusted.
- Web applications evolved to connect our most sensitive back end systems to an open, public network without inherent security controls.
- The web was not designed to be a secure platform, and is thus subject to trivial attacks on confidentiality, integrity and availability.
- Only a small percentage of security breaches are detected and noisy enough to result in measurable losses and/or gain management attention.
- We have a large volume of old application code to fix, while constantly writing new code.
- Web applications are complex combinations of platforms, tools, and services from multiple providers, each with its own security challenges.
- Many web applications are mission critical, but may not have been when they were designed.
- Due to the design of the web (and VPNs), even internal web applications are external applications.
- Web applications are custom applications; we are the vendors, and no one else will provide security fixes.
- Web application projects are funded to offer more services, easier, faster, and cheaper than before, while security tends to limit these benefits.
- No single web application security tool provides effective security on its own.

To update an old analogy: it's like trying to secure a fighter jet we're building in the middle of combat with constantly changing functional requirements, parts, fuel, and laws of physics — all while facing an army of millions of almost infinitely diverse anonymous attackers. Our core business computing platforms are moving under our feet in a dynamically changing risk environment.

The Business Justification

Application security is typically underfunded, but the reality is that it's difficult to convince management why an application that appears to be up and running just fine needs additional protection. It's hard to change a development process deeply ingrained in the development staff, or obtain funding for additional security controls when the benefits are difficult to quantify. Building a full business justification model for web application security is beyond the scope of this report, but we can't talk about building a program without providing at least some basic tools to determine how much you should invest and how to convince management to support you. The following list isn't a comprehensive business justification model, but includes drivers we typically see used to justify web application security investments:

Compliance: Like it or not, security controls are mandated by government regulation, industry standards & requirements, and contractual agreements. We like to break compliance into three separate justifications — industry or regulatory mandated controls (PCI web application security requirements), non-mandated controls that avoid other compliance violations (data protection to avoid breach disclosure), and investments to reduce the costs of compliance (lower audit costs or TCO). The average organization uses all three factors to gauge web application security investments.

Fraud Reduction: Depending on your ability to accurately measure fraud, it can be a powerful driver and justification for security investments. In some cases you can directly measure fraud rates and show how they can be reduced with specific security investments. Keep in mind that you may not have the right infrastructure to detect and measure this fraud in the first place, which might itself provide sufficient justification. Penetration tests are also useful for justifying investments to reduce fraud — a test may show previously unknown avenues for exploitation that could be under active attack or open to future attack. You can use the penetration test to estimate potential fraud and map that to security controls to reduce losses to acceptable levels.

Cost Savings: As we mentioned in the compliance section, some web application security controls can reduce the cost of compliance (particularly audit costs), but there are additional opportunities for savings. Using web application security tools and processes through development and maintenance can reduce the need for and costs of manual processes or controls to remediate software defects and flaws, and may help create general efficiency improvements. We can also calculate cost savings from incident reduction, including incident response and recovery costs.

Availability: When dealing with web applications, we look at both total availability (uptime), and service availability (loss of part of the application due to attack or to repair a defect). For example, while it's somewhat rare to see a complete site outage due to a web application security issue (although it definitely happens), it's not unusual to see an outage of a payment system or other functionality. We also see cases where due to active attack a site needs to shut down some of its own services to protect users, even if the attack didn't break the services directly.

User Protection: While this isn't quantifiable to a dollar amount, a major justification for investment in web security is to protect users from being compromised by their trust in you (yes, this has reputation implications, but we cannot precisely quantify them). Attackers frequently compromise trusted sites not to steal from that site, but to use it to attack the site's users. Even if you aren't concerned with fraud resulting in direct losses to your organization, it's a problem if your web application is used to defraud your users. Most organization derive value or direct revenue from customer data, and there is an implied custodial duty to protect the information you have gathered and use.

Reputation Protection: While many models attempt to quantify a company's reputation and potential losses due to reputation damage, the reality is all those models are bunk — there is no accurate way to measure the potential losses associated with a successful attack. Despite surveys indicating users switch to competitors if you lose their information, or that you'll lose future business, real world reports show that user behavior rarely aligns with survey responses. For example, TJX was the largest retail breach notification in history, yet sales went up after the widely reported incident. But just because we can't quantify reputation damage doesn't mean it isn't an important factor in justifying web application security. Just ask yourself (or management) how important that application is to the public image of your organization, and how willing you or they are to accept the risk of losses ranging from defacement to lost customer information to downtime. User, investor, and partner trust in your company and services is complicated, and while it does not track directly with site security, trust remains important to the overall value of the business.

Breach Notification Costs: Aside from fraud, we also see direct losses associated with breach notifications (if sensitive information is involved). Ignore all the fluffy reputation/lost business/market value estimates and focus on the hard dollar costs of making a list, sending out notifications, and staffing the call center for customer inquiries. You might also factor in the cost of credit monitoring, if you'd offer that to your customers.

You will know which combination of these works best for you based on your own organizational needs and management priorities, but the key takeaway should be that you likely need to mix quantitative and qualitative assessments to prioritize your investments. If you're dealing with private information (financial/retail/healthcare), compliance drivers and breach notification mixed with cost savings are your best option. For general web services user protection & reputation, fraud reduction and availability are likely at the top of your list. And let's not forget that many of these justifications are just as relevant for internal applications. Whatever your application, there is no shortage of business (as opposed to technical) reasons to invest in web application security.

Web Applications Are Different

We have many years invested in understanding network and host security issues, and have built nearly all of our Information Technology security programs to focus on the underlying infrastructure. But as we've discussed, web application security is fundamentally different than host or network security, and requires a different approach. Web application security is also different from traditional software security, although it has far more in common with that discipline. In previous sections we've focused on the business; now we're going to get (just a little) more technical and talk about the specific technical and non-technical reasons web application security is different, before delving into our recommended web application security lifecycle.

Why Web Application Security Is Different Than Host and Network Security

With network and host security, our focus is on locking down our custom implementations of someone else's software, devices, and systems. Even when securing enterprise applications, that typically involves locking down the platform, securing communications, authenticating users, and implementing security controls provided by the application platform. But with web applications we not only face all those issues — we are also dealing with custom code we've often developed ourselves. Whether internal-only or externally accessible, there are major differences:

- **Browser UI:** In most applications we program the user display/interface. With web applications, we rely on an external viewer (the browser) we can't completely control, which interacts with other applications at the same time. The browser may be used by a hostile party, or have been compromised by a third party. There is also a huge variety in web browsers- each comes with different vulnerabilities, rendering and code interpretation engines, and security features. Browser manufacturers also release new versions with new features on their own schedules. One tiny change, such as how self-signed digital certificates are managed or iframes rendered, may have large implications for your web application. In essence, you can never trust your display interface. While certain SOA applications have issues of trust establishment between components, they are far worse with web applications.
- **Custom code equals custom vulnerabilities:** With web applications you typically generate most of the application code yourself (even using common frameworks and plugins). That means most vulnerabilities will be unique to your application. Unless you are constantly evaluating your own application, there's no one to tell you when a vulnerability has been discovered, and no one else to provide a patch.
- **You are the vendor:** When a vulnerability appears, you won't have an outside vendor providing a patch (you will, of course, have to install patches for whatever infrastructure components, frameworks, and scripting environments you use). If you provide external services to customers, you may need to meet your service level agreements and must be prepared to be reviewed by customers — just as you review your own software vendors — even if software isn't your business. You have to patch your own vulnerabilities, deal with your own customer relations, and provide everything you expect from those who provide you with software and services.
- **Firewalls/shielding alone can't protect web applications:** When we experience software vulnerabilities with our enterprise software, from operating systems, to desktop applications, to databases and everything else, we use tools

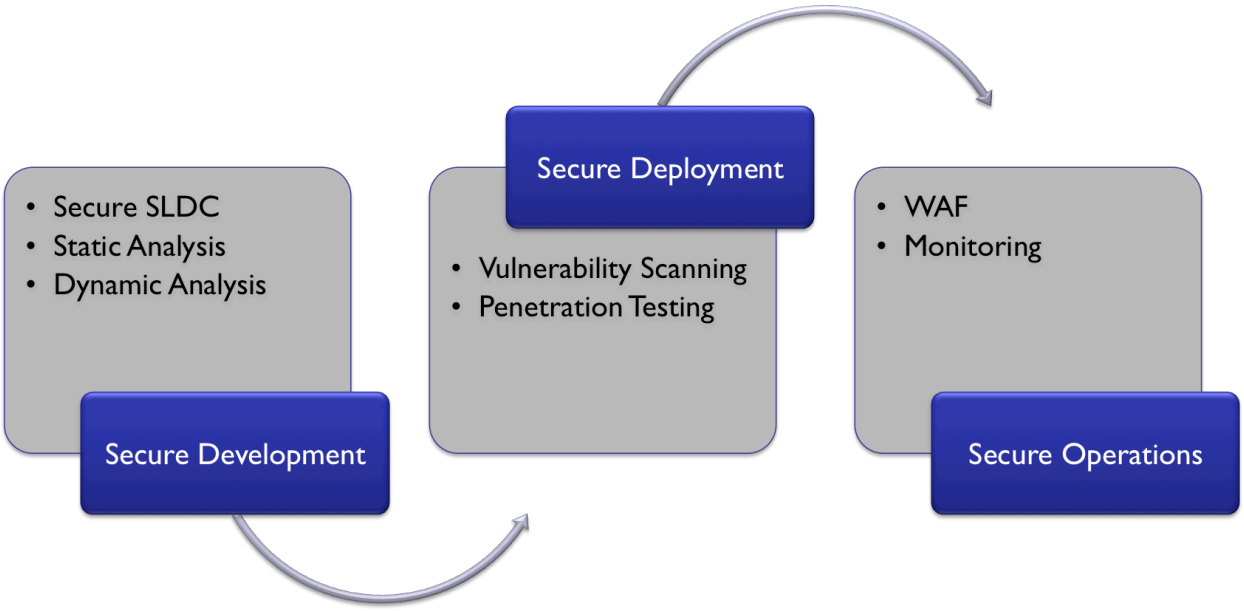
like firewalls and IPS to block attacks while we patch the vulnerable software. This shield-then-patch model has only limited effectiveness for web applications. A web application firewall (WAF) can't protect you from logic flaws. While WAFs can help with certain classes of attack, out of the box they don't know or understand your application and thus can't shield custom vulnerabilities they aren't tuned for. WAFs are an important part of web application security, but only work properly as part of a comprehensive program, as we'll discuss later.

- **Eternal Beta Cycles:** When we program a traditional stand-alone application it's usually designed, developed, and tested in a controlled environment before being carefully rolled out to select users for additional testing, then general release. While we'd like all web applications to run through this cycle, as we discussed earlier it's rarely so neat. Some applications are designated beta and treated as such by the development teams, but in reality they've quietly grown into full-bore essential enterprise applications. Other applications are under constant revision and don't even attempt to follow formal release cycles. Continually changing applications challenge both existing security controls (like WAFs) and response efforts.
- **Reliance on frameworks/platforms:** We rarely build our web applications from the ground up from shiny new C code. We use a mixture of different frameworks, development tools, platforms, and off-the-shelf components to piece them together. We are challenged to secure and deploy these pieces as well as the custom code we build with them. In many cases we create security issues through unintended uses of these components or interactions between multiple layers due to the complexity of the underlying code.
- **Heritage (legacy) code:** Even if we were able to instantly create perfectly secure code from here on forward, we still have massive code bases full of old vulnerabilities to analyze and perhaps fix. If older code is in active use, it needs just as much security as anything new we develop. With links to legacy systems, modification of older applications often ranges from impractical to impossible, placing the security burden on the newer web application.
- **Dynamic content:** Most web applications are extremely dynamic in nature, creating much of their content on the fly, not infrequently using elements (including code) provided by the user. Because of the structure of the web, while this kind of dynamically generated content would fail in a traditional application, web browsers try to render it to the user — thus creating entire new classes of security issues.
- **New vulnerability classes:** As with standard applications, researchers and bad guys are constantly discovering new classes of vulnerabilities. For example, using Flash to spy on user activities and pushing code into back-end databases are threat types unseen before web applications. In the case of the Web, these often affect nearly every web site on the face of the planet the moment they are discovered. Even if we write perfect code today, there's nothing to guarantee it will be safe tomorrow.

We've listed a number of reasons we need to look at web applications differently, but the easiest way to think about it is that web applications have the scope of externally-facing network security issues, the complexity of custom application development security, and the ramifications of ubiquitous host security vulnerabilities. Implementing a web security program may require consensus in your organization that is not currently present. Change is hard and risky, and while security may reduce costs in the long run, it forces us to pay costs immediately without knowing the full potential returns. This all means you need really good reasons to get executives and management to change what they do today. If you compare the costs to the benefits illustrated within this section, you can get a handle on how security expenditures affect your organization.

The Web Application Security Lifecycle

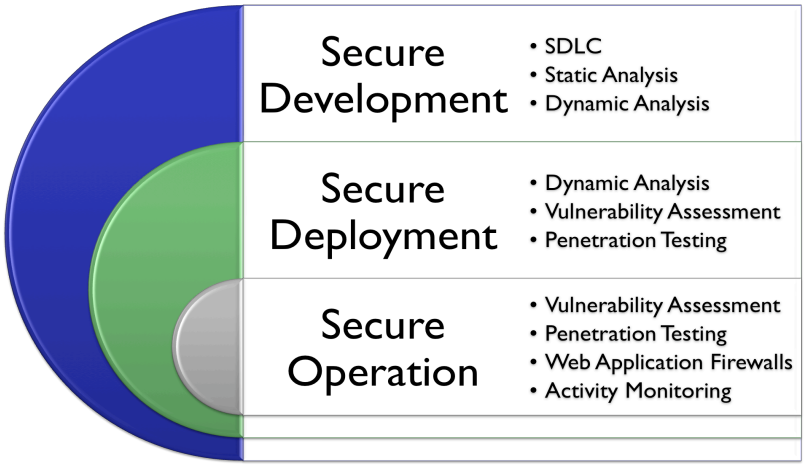
In the remainder of this report we will recommend actionable steps that promote web application security and are in keeping with existing development and management frameworks. Given the scope of the coverage area, we will not provide an in-depth review for many of these disciplines, but instead briefly cover the possibilities. While web applications present different challenges, additional steps to address these issues aren't radical deviations from what you likely do today. With a loose mapping to the Software Development Lifecycle, we divide our advice into three steps across seven coverage areas that look like this:



Secure Development

Process and Training: This section's focus is on building security into the software development lifecycle (SDLC). It includes training for people who deliver web applications and improved processes to guide their activity. Security awareness training is managed through education and supportive process modifications, as a precursor to making security a functional requirement of the application. We will also review tools that automate portions of the effort: static analysis to aid engineering in identifying vulnerable code, and dynamic analysis to detect anomalous application behavior.

- **Secure SDLC:** Introducing secure development practices and software assurance into the web application programming process.
- **Static Analysis:** Tools that scan the source code of an application to look for security errors. Often called "white box" tools.
- **Dynamic Analysis:** Tools that interact with a running application and attempt to 'break' it, but don't analyze the source code directly. Often called "black box" tools.



Controls overlap into multiple phases

Secure Deployment

At the stage where an application is code-complete, or is ready for more rigorous testing and validation, it is time to confirm that it doesn't suffer from serious known security flaws, and is configured such that it is not subject to any known compromises. This is where we start introducing vulnerability assessments and penetration testing — along with their respective approaches to configuration analysis, threat discovery, patch levels, and operational consistency checking.

- **Vulnerability Assessment:** Remote scanning of a web application both with and without access credentials to find vulnerabilities. Web application vulnerability assessments focus on the application itself, while standard vulnerability assessments focus on the host platform. May be a product, service, or both.
- **Penetration Testing:** Penetration testing is the process of actually breaking into an application to determine security vulnerabilities and the risks they pose. While vulnerability assessments find security flaws, penetration tests explore those holes to measure impact and categorize/prioritize. May be a product, service, or both.

Most people do not trust developers to thoroughly test their own code. They are incentivized to develop and release new features, not rigorously test and prove stable functionality. Testing of code often is conducive to a different mindset, and has evolved to include a specialized set of skills and tools to cope with the scope required to actually verify function. Security should be considered in the same light: we need the development team to understand the security issues, and should alter development processes to enable them to develop secure code and test cases to verify their code, but their efforts also need to be validated externally.

Secure Operation

In this section we move from preventative tools and processes to those which provide detection capabilities and can react to events from an operational application. Our primary focus is on web application firewalls' ability to screen an application from unwanted uses, and monitoring tools that scan requests for inappropriate activity against the application or associated components. Recent developments in detection tools promote enforcement of policies, react intelligently to events, and combine multiple services into a cooperative hybrid model.

- **Web Application Firewalls:** Network tools that monitor web application traffic and alert on — or attempt to block — known attacks.
- **Application and Database Activity Monitoring:** Tools that monitor application and database activity (via a variety of techniques) for auditing, and generate security alerts based on policy violations.

As many organizations are just starting to come to terms with application security, we will map out a framework and illustrate the options so web development organizations can start looking at the whole picture. We need to stress that we are only providing an overview of available options, and a basic introduction to how they work and where they fit in. While we discuss 'what' technologies and methods you should consider, the details on 'how' could encompass several books. The specifics of how to modify processes and apply different methodologies, and the associated nuances of Secure SDLC, are beyond the scope of the paper.

Web application security is a field undergoing rapid advancement — almost as fast as the bad guys come up with new attacks. We will discuss leading edge technologies and the future of the market, and specifically tools used to identify and protect against new security threats. While tools are important, they are not sufficient alone. You need the right tool in the right hands as part of the right process. That said, we will be spending a significant amount of time on tools throughout the remainder of this paper for several reasons:

- Many of the tools automate much of the leading edge approaches to security evaluation.
- Your development and QA staff may not be able to keep up with the evolving trends in security, or may not have the required depth of understanding in the relevant security fields, and the policies and knowledge built into tools can help fill the gap.
- The scope of security testing, like quality assurance, requires automation. The requirement vs. resource ratio for any assurance group is far too high without automation.
- Tools provide a platform to implement and customize policies and test cases, providing a knowledge repository for lessons learned over time as well as a testbed for regression efforts.

In the rest of this report we'll break down each of these areas and show how they fit into a larger program, as well as their respective advantages and disadvantages. Keep in mind that we could probably write a book or two on each of these tools, technologies, and processes, so we will stick to the highlights.

Secure Development

In web application security insufficient attention is often paid to process modification, education, and development tool choices. Security is frequently bolted on as an afterthought rather than designed in. Implementing security during the development phases of the web application lifecycle produces stronger application security at lower cost. This section will illustrate the best options for integrating security during pre-deployment phases of application development (*i.e.,* requirements gathering, design, implementation, and QA).

Web Application Security: Training and the SDLC

Most web applications today were designed, built, and deployed before web application security was considered. Secure coding practices are just now entering the consciousness of most web development teams, and usually only after a security 'event'. Project management and assurance teams typically take on security only when a compliance requirement is dropped into their laps. News of major breaches has raised awareness of SQL injection attacks, but many developers remain unaware of how reflected Cross Site Scripting and Cross Site Request Forgeries are conducted, much less what can be done to protect against them. Secure application development practices, and a Secure Software Development Lifecycle, are in their infancy in terms of both maturity and adoption. The knowledge exists but hasn't reached everyone or been widely acted upon yet.

Regardless of what drives your requirements, education and process modification are important first steps for producing secure web applications. Whether you are developing a new code base or retrofitting older applications; project managers, developers, and assurance personnel need to be educated about security challenges to address; as well as secure design and coding techniques. The curriculum should cover both the general threats to account for and the methods hackers typically employ to subvert systems. Specialized training is necessary for each sub-discipline, including process modification options, security models for the deployment platform, security tools, and testing methodologies. If your team does not know what needs to be secured, or how to do it, you are dependent on third parties (service providers or hackers) to discover these flaws for you, and that is a far more expensive way to learn.

Project managers needs to be aware of what types of threats are relevant to the web applications they are responsible for, and how to make trade-offs to minimize risk while still providing desired capabilities. Developers and QA need to understand how common exploits work, how to test for them, and how to address weaknesses. Whether your company creates its own internal training program, organizes peer educational events, or invests in third party classes, understanding is key for producing secure applications. Threat modeling, secure design principles, threat surface reduction, functional security requirements, secure coding practices, and security review/testing form the core of an effective secure SDLC, and are relatively straightforward to integrate into nearly all development processes.

Process also plays an important role in code development, and affects security much the same as it affects employee productivity and product quality. If the product's specification lacks security requirements, you can't expect it to be secure. A product that doesn't undergo security testing, just like a product that skips functional testing, will suffer from flaws and errors. Modification to the Software Development Lifecycle to include security considerations is called Secure SDLC, and includes simple sanity checks throughout the process to help discover problems early. While Secure SDLC is far too involved for in-depth discussion here, our goal is to highlight the need for development organizations to consider security as a requirement during each phase of development.

Tools and test cases, as we will discuss below, can be used to automate testing and assurance, but training and education are essential for taking advantage of them. Using them to augment the development and assurance process reduces overhead compared to *ad hoc* security adoption, and cuts down on vulnerabilities within the code. Team members educated on security issues are able to build libraries of tests that help catch typical flaws across all newer code. This is not just for in-house tools and scripts used for various testing phases, but also for off-the-shelf tools as well. Extreme Programming techniques can help certify that modules and components meet security requirements as part of unit testing, alongside non-security functional testing and regression sweeps by assurance teams. Remember — you are the vendor, and your team should know your code better than anyone, including how to break it.

Static Analysis Tools

There are a number of purpose-built third party tools to help with code review for security. Static analysis examines the source code of a web application for common vulnerabilities, errors, and omissions within the constructs of the language itself. This serves as an automated counterpart to peer review. Among other things, these tools generally scan for unhandled error conditions, object availability and/or scoping, and potential buffer overflows. The technique is called "static analysis" because it examines the source code files, rather than execution flow of a running program.

These products run during the development phase to catch problems prior to more formalized testing procedures. The earlier a problem is found the easier (and cheaper) it is to fix. Static analysis supplements code review performed by developers, speeding up scans and finding bugs more cheaply than humans. The tools can integrate with source code management for automated execution, which again helps identify issues early.

Static analysis is effective at discovering 'wetware' problems, or problems in the code that are directly attributable to programmer error. The better tools integrate well with various development environments (providing educational feedback and suggesting corrective actions to programmers); can prioritize discovered vulnerabilities based on included or user-provided criteria; and include robust reporting to keep management informed, track trends, and engage the security team in the development process without requiring them to also be programmers.

Static analysis tools do not account for all pathways within the code, and are blind to certain types of vulnerabilities and problems that are only apparent at runtime. To fill this gap, dynamic analysis tools have emerged over the last few years, as well as hybrid tools which combine both static and dynamic analyses.

Dynamic Analysis Tools

Dynamic analysis is intended to identify problems which cannot be detected from source code, or those more easily seen during execution. One type is 'fuzzers', which sends intentionally bogus or harmful inputs to the application, and looks for unintended results or crashes. There are many variations in this space, but three basic differentiators across the market:

White Box vs. Black Box: The test application that 'exercises' the application may not have any prior knowledge, probing the application as a "black box", and testing for exploits as it navigates the application. In most cases, the tests are a "white box" derivative of the functional tests, traversing known pathways and supplying malicious or garbage input values. Random black box testing is more representative of how an outside hacker will act, and is unburdened by assumptions, but takes longer to run and cannot focus on known or suspected weak spots. For web applications, it is important to test both credentialed and anonymous access. Some vulnerabilities may not be visible to a random attacker, but show up when logging in as a known user.

Input Values: Input values may be random, they may be deduced on the fly based upon input type, or they may be targeted. Random inputs are a good way to verify basic integrity checks are working, and help finds generic issues across the code, while targeted inputs are useful for checking against known vulnerabilities.

Output and Behavior: With either white box or black box testing, human examination of the results is required to distinguish real problems from false positives. Error conditions are easy to find, and most dynamic test tools discover and report on these. Some monitor resource usage and detect not only error conditions but also resource allocation issues. Similar to the way debuggers work, dynamic analyzers can monitor the internal resources of an application while it is under test — observing memory, pointers, message queues, and input variables. The tests can both highlight specific effects of system usage patterns and identify areas of concern. The former is easy to use and understand, and works with any web application, while the later requires specific knowledge of the application.

Partially because of these variables, dynamic analysis tools vary in speed, effectiveness, and automation. As they focus on application behavior and results, they provide concrete results for 'what-if' scenarios that static analysis cannot. Dynamic and static analysis are complimentary technologies, with different strengths, and intended for slightly different audiences. Some vendors provide both tools together, which allows them to share results and provide common reporting.

A well-structured web application security program starts with good education and integration of security into the SDLC — with threat modeling, secure design, functional security requirements, appropriate use of static and dynamic analysis, secure coding practices, ongoing security education, and formalized security testing.

Secure Deployment

Following the progression through the software development lifecycle, we now shift our focus to security considerations when launching a web application. During the deployment stage — after quality and security assurance testing, we move into vulnerability assessment and penetration testing to ensure we are in compliance with security best practices and our application is hardened against attack. Keep in mind that we look at web application security as an ongoing process with overlapping stages. Although we've divided things up into phases to facilitate our discussion, that doesn't mean one technique ends before the next begins. For example, you'll likely continue using dynamic analysis in the deployment stage, and will definitely use vulnerability assessment and penetration testing in the operations phase. Keep in mind that we're showing you the big picture and everything that's available. We'll discuss prioritization and where to focus efforts for those of you on a limited budget later, as we're not naive enough to think you all can afford everything on the market.

Vulnerability Assessment

In a vulnerability assessment we scan a web application to identify anything an attacker could potentially use against us (some assessments also look for compliance/configuration/standards issues, but the main goal in a VA is security). We can do this with a tool, service, or combination.

A web application vulnerability assessment is very different than a general vulnerability assessment where we focus on networks and hosts. In those we scan ports, connect to services, and use other techniques to gather information revealing the patch levels, configurations, and potential exposures of our infrastructure. Since as we've discussed even "standard" web applications are essentially custom, we need to dig a little deeper, examine application function and logic, and use more customized assessments to determine if a web application is vulnerable. With so much custom code and implementation, we have to rely less on known patch levels and configurations, and more on actually banging away at the application and testing attack pathways. As we've said before, custom code means custom vulnerabilities.

For web application security we are skipping over traditional host and network vulnerability assessments, and focusing on third party web applications and frameworks, and the technical and business flaws of custom applications.

The web application vulnerability assessment market includes both tools and services. Even if you decide to go the tool route, it's absolutely critical that you place the tools in the hands of an experienced operator who will understand and be able to act on the results. It's also important to run both credentialed and un-credentialed assessments. In a credentialed assessment, the tool or assessor has usernames and passwords of various levels to access the application. This allows them to assess the application as if they were an authorized user attempting to exceed authorization.

Tools and Solutions

There are a number of commercial, free, and open source tools and solutions available for assessing web application vulnerabilities, with widely varying capabilities. Some tools only focus on a few kinds of exploits, and experienced

assessors use a collection of tools and manual techniques. For example, there are tools that focus exclusively on finding and testing SQL injection attacks. Enterprise-class solutions are broader, and should include a wide range of tests for major web application vulnerability classes, such as SQL injection, cross site scripting, and directory traversals. The OWASP Top 10 is a good starting list of major vulnerabilities, but an enterprise class solution shouldn't limit itself to just one list or category of vulnerabilities. An enterprise solution should also be capable of scanning multiple applications, tracking results over time, providing robust reporting (especially compliance reports), and providing reports customized for local requirements (*e.g.,* add/drop scans).

Tools are typically software, but can also include dedicated appliances. VA tools can run either manual scans with an operator behind them, or automatic scans on a schedule. Since web applications change so often, it's important to scan any modifications or new applications before deployment, as well as live applications on an ongoing basis.

Services

Not all organizations have the resources (including expertise) or need to buy and deploy tools to assess their own applications, and in some cases external assessments may be required for compliance.

There are three main categories of web application vulnerability assessment services:

- **Fully automatic scans:** These are machine-run scans that don't involve a human operator. The cost is low, but they are more prone to false positives and negatives. They work well for ongoing assessments on a continuous basis, but due to their limitations you'll likely still want more in-depth assessments from time to time.
- **Automatic scans with manual evaluation:** An automatic tool performs the bulk of the assessment, followed by human evaluation of the results and additional testing. These provide a good balance between automated assessments and the costs of completely manual assessment. You get deeper coverage and more accurate results, but at a higher cost.
- **Manual assessments:** A trained security assessor evaluates your web application to identify vulnerabilities. Typically an assessor uses their own tools, then validates the results and provides custom reports. The cost is higher per assessment than the other options, but a good assessor may find more flaws.

Penetration Testing

The goal of a vulnerability assessment is to find potential avenues an attacker can exploit, while a penetration test goes a step further and validates whether attack pathways result in risk to the organization. In a web application penetration test we attempt to penetrate our own applications to determine what an attacker can do, and what the consequences might be.

Vulnerability assessments and penetration tests are highly complementary, and frequently performed together since the first step in any attack is to find vulnerabilities to exploit. The goal during the vulnerability assessment phase is to find as many of those flaws as possible, and during the penetration test to validate those flaws, determine potential damages, and prioritize remediation efforts.

That's the key value of a penetration test — it bridges the gap between the discovered vulnerability and the exploitable asset so you can make an informed risk decision. VA finds vulnerabilities but not their potential consequences. For example, your vulnerability scan may show a SQL injection vulnerability, but when you attempt to exploit it in the penetration test it may also reveal sensitive information, and damage the web application so it no longer functions as designed. A seemingly minor vulnerability might turn out to allow exploitation of the entire web application.

Some experts consider penetration tests important because they best replicate the techniques and goals an attacker will use to compromise an application, but we find that a structured penetration test is more valuable as a risk prioritization tool. If we think in terms of risk models, a good penetration test helps fill in the potential severity/impact side of the analysis. Of course, this also means you need to focus your penetration testing program on risk assessment, rather than simply "breaking" a web application.

As with vulnerability assessments, penetration tests can include the entire vulnerability stack from the network and operating system up through custom application code, and both tools and services are available. Again, we'll limit this discussion to aspects of penetration testing specific to web applications.

Tools and Solutions

A web application penetration testing solution provides a framework for identifying and exploiting web vulnerabilities, and measuring or estimating their potential impact. While many tools are used by penetration testers, most of them are point tools rather than broad solutions. They are run prior to deployment with test data, as these tools exploit vulnerabilities that are commonly disruptive or damaging to the application and data. An enterprise-class solution adds features to better assist the risk management process, support internal assessments, and reduce the costs of internal penetration tests. This includes broad coverage of application vulnerability classes, "safe" techniques to exploit applications without interfering with their ongoing use, workflow, extensive reporting, and automation for ongoing assessments. One advantage of penetration testing solutions is you can integrate them into the development and deployment process, rather than only assessing live web applications.

Penetration testing tools and solutions are always delivered as software and should be used by a trained operator. While parts of the process can be automated, once you dig into active exploitation and analysis of results, you need a human being.

When using penetration testing (or vulnerability assessment) tools, you have the choice of running them in a safe mode that reduces the likelihood of causing a service or application outage (this is true for all kinds of VA and penetration testing, not just web applications). Should you test a live application, you'll want to use safe mode, but be extremely careful as even 'safe' tests often cause problems with live applications, ranging from service outages to database cruft. But these tools are extremely valuable in assessing development/test environments before applications are deployed, are especially important in staging, and may be used on live applications under carefully controlled circumstances.

Services

Unlike vulnerability assessments, we are unaware of any fully automated penetration testing services (although some of the automatic/manual services come close). Due to the more intrusive and fluid nature of a web application penetration test you always need a human to drive the process. Penetration testing services are typically offered as either one-off consulting projects or a subscription service for scheduled tests. Testing frequency varies greatly based on business needs, exposure, and the nature of the targeted web applications. Internal applications might only be assessed annually as part of a regularly scheduled enterprise-wide test.

When evaluating a penetration testing service it's important to understand their processes, experience, and results. Some companies offer little more than a remote scan using standard tools, and fail to provide prioritized results usable in your risk assessment. Others simply "break into" web applications and stop as soon as they get access. You should give preference to experienced organizations with an established process which can provide sample reports (and references) that meet your needs and expectations. Most penetration tests are structured as either fixed-time or fixed-depth. In a fixed-time engagement (the most common) the penetration testers discover as much as they can in a specified amount of time. Engagements may also divide the time used into phases — *e.g.,* a blind attack, a credentialed attack, and so on.

Fixed-depth engagements stop when a particular penetration goal is achieved, no matter how much time it takes (*e.g.,* administrative access to the application or access to credit card numbers).

Integrating Vulnerability Assessment and Penetration Testing into Secure Deployment

Although most organizations tend to focus web application vulnerability assessments and penetration tests on externally accessible production applications, the process really begins during deployment, and shouldn't be limited to external applications. It's important to test applications in depth before you expose them to either the outside world or internal users. There's also no reason, other than resources, not to integrate assessment into the development process itself at major milestones.

Before deploying an application, set up your test environment so it accurately reflects production. Everything from system configurations and patch levels through application connections to outside services must match production or results won't be reliable. Then perform your vulnerability assessment and penetration test. If you use tools, you'll do this with your own (appropriately trained) personnel. If you use a service, you'll need to grant them remote access to your test environment — review and carefully consider their access requirements. Since few organizations can afford to test every single application to the same degree, you'll want to prioritize based on the exposure of the application, its criticality for business operations, and the sensitivity of the data it accesses.

For any major externally accessible application you'll want to engage a third party for both VA and penetration testing before deployment. If your business relies on it for critical operations, and it's publicly accessible, you should have an external assessment.

Once an application is deployed, your goal should be to perform at least basic assessment of any major modifications before they are committed. For critical applications, engage a third-party service for ongoing vulnerability assessments and periodic penetration tests (at least annually or after major updates) in addition to your own testing.

We know not all of you have the resources to support internal and external tools and services for VA and penetration testing on an ongoing basis, so you'll need to adapt these recommendations for your own organizations. We've provided a high level overview of what's possible, and some suggestions on where and how to prioritize.

Secure Operations

So far we've covered secure development and secure deployment; now it's time to move on to secure operations. This is the point where the application moves out of development and testing, and into production. Keep in mind that much of what we've talked about until now is to augment what you have today — just because you have a production system doesn't mean you throw away all your tools and processes. Updates still need to go through secure development, systems and applications still need to vulnerability assessments and penetration testing (although you need to test live applications differently than staging), and configuration management and ongoing secure management are more important than ever before.

In the secure operations phase we add two new technology categories to support two additional processes — Web Application Firewalls (WAF) to *shield* from certain types of attacks, and *monitoring* at the application and/or database levels to support auditing and security alerts.

Web Application Firewalls (WAF)

The role of a web application firewall is to sit in front of or next to a web application, monitor application activity, and alert or block policy violations. Thus it potentially serves two functions — as a detective control for monitoring web activity, and as a preventative control.

A web application firewall is a firewall specifically built to watch HTTP requests and block those that are malicious or don't comply with specific rules. The intention is to catch SQL injection, Cross Site Scripting (XSS), directory traversal, and various HTTP abuses, as well as misuse of valid authorization, request forgeries, and other attempts to manipulate web application behavior. WAF rules and policies are effectively consistency checks, for both the HTTP protocol and application functionality. WAFs can alert or block activity based on generic attack signatures (such as a known SQL injection attack for a particular database), or application-specific signatures for the web application being protected.

WAF products examine inbound and outbound HTTP requests, compare them with the firewall rules, and create alerts for conditions of concern. Finally, the WAF selects a disposition for the traffic: 1) let it pass, 2) let it pass but audit, 3) block the transaction, or 4) reset the connection.

WAFs are typically network appliances. They are normally placed in-line as a filter for the application (proxy mode); or 'out-of-band', receiving traffic from a mirror or SPAN port. In the former scenario, all inbound and outbound requests are intercepted and inspected prior to the web server receiving the request or user receiving the response, reducing load on the web application. For SSL traffic, inline WAFs also need to proxy the SSL connection from the browser so they can decrypt and inspect traffic before it reaches the web server, or after responses leave the server. In out-of-band mode, there are additional techniques to monitor the encrypted connections by placing a copy of the server certificate on the

WAF, or positioning it behind an SSL concentrator. Some vendors also provide WAF capabilities via plug-ins for specific platforms, rather than through external devices.

The effectiveness of any WAF is limited by the quality of its policies. Policies are important not merely for recognizing and stopping known specific attacks, but also for flexibly dealing with ambiguous and unknown threat types — while keeping false positives manageable, without preventing normal transaction processing. The complexity of the web application, combined with the need for continuous policy updates, and the wide variety of deployment options to accommodate, pose a complex set of challenges for any WAF vendor. Simply dropping a WAF in front of your application and turning on all the default rules in blocking mode is a recipe for disaster. There is no way for a black box to effectively understand all the intricacies of a custom application, and customization and tuning are essential for keeping false positives and negatives under control.

When deployed in monitoring mode, the WAF is used in a manner similar to an intrusion detection system (IDS). It's set to monitor activity and generate alerts based on violations. This is how you'll typically want to initially deploy the WAF, even if you plan on blocking later. It gives you an opportunity to tune the system and better understand application activity before you start trying to block connections. An advantage of monitoring mode is that you can watch for a wider range of potential attacks without worrying that false positives will result in inappropriate blocking. The disadvantages are that your incident handlers will spend more time dealing with these incidents and false positives, and also that bad activity won't be blocked immediately.

In blocking/enforcement mode, the WAF will break connections by dropping them (proxy mode) or sending TCP reset packets (out of band mode) to terminate the connection. The WAF can then ban the originating IP, permanently or temporarily, to stop additional attacks from that origin. Blocking mode is most effective when deployed as part of a "shield-then-patch" strategy to block known vulnerabilities in your application.

When a vulnerability is discovered in your application, you can build a specific signature to block attacks against it and deploy that to the WAF (the "shield"). This protects your application as you go back and fix the vulnerable code, or wait for an update from your software provider (the "patch"). The shield-then-patch strategy greatly reduces potential false positives that interfere with application usage and also improves performance, but is only possible when you have adequate processes to detect and analyze these vulnerabilities.

You can combine both approaches by deploying a larger signature set in monitoring mode, but only enabling a few policies in blocking mode.

Given these challenges, satisfaction with WAF products varies widely among security professionals. While WAFs are effective against known threats, they are less capable of discovering new issues or handling questionable use cases. Some WAF products are addressing these issues by linking web application firewalls more tightly to vulnerability assessment process and tools, as we'll discuss in a moment. Regardless, policy set augmentation and maintenance remain issues, so you need to be aware that a WAF will require additional investment in this area.

Monitoring

Monitoring is primarily used for discovery, both of how an application is used by real users, and also for how it can be misused. The fundamental value of monitoring is to learn what you do not already know — this is important not only for setting up a WAF, but also for tuning an application security strategy more generally. Although WAFs provide some level of application activity monitoring, there are three additional ways to monitor web applications, each with a different perspective on application activity:

- Network Monitoring: Monitoring network activity between the user (Internet) and the web server. This category includes web application firewalls, intrusion detection systems, packet sniffers, and other external tools. While generic network security and sniffing tools can capture all network traffic, they have much less capability to place it in context and translate network activity into application transactions. Simply viewing HTTP traffic is often insufficient for understanding what users are attempting in an application — this is where interpretation is required. If a solution includes web application specific analysis and the ability to (potentially) audit all web activity, we call it Web Application Monitoring (WAM). While network monitoring is easy to implement and doesn't require application changes, it can only monitor what's going into and coming out of the application. This may be useful for detecting traditional attacks against the application stack, but much less useful than seeing traffic fully correlated to higher-level transactions.
- Application Auditing/Logging: Collection and analysis of application logs and internal activity. Both custom and off-the-shelf applications often include internal auditing capabilities, but there is tremendous variation in what they capture and how it's formatted and stored. While you gain insight into what's actually occurring in the application, not all applications log equally (or at all) — you are limited to whatever the programmers decided to track. For major enterprise applications, such as SAP, we've seen third party tools that either add additional monitoring or can interpret native audit logs. Log management and SIEM tools can also be used to collect and interpret (to a more limited degree) application activity when audit logs are generated.
- Database Activity Monitoring: DAM tools use a variety of methods to (potentially) record all database transactions and generate alerts on specific policy violations. By monitoring activity between the application and the database (or inside it), DAM can provide a more precise examination of data and usage, as well as awareness of multi-step transactions which directly correspond to business functions. Some DAM tools have specific plugins for major application types (*e.g.,* SAP & PeopleSoft) to translate database transactions into application activity. Since the vast majority of web applications run off databases, this is an effective point to track activity and look for policy violations. A full discussion of DAM is beyond the scope of this post, but more information is available in our DAM white paper.

WAFs can be used for monitoring, but dedicated monitoring tools offer a wider range of activity collection, which helps to detect probing which may not be obvious from HTTP requests alone. The focus of web application monitoring (WAM) is to examine behavior across data sources and provide analysis, recording activity trails and alerting on suspicious events, whereas WAFs are more focused on detection and blocking of known threats. There are significant differences between WAM and WAF in areas such as activity storage, aggregation of data from multiple sources, and examination of the collected data, so choosing the best tool depends on the specifics of the requirement. We must point out that web application monitoring products are not fully mature, and the handful of available products are early in the product evolution cycle.

WAF + VA

Several vendors have begun providing a hybrid model that combines web application vulnerability assessment with a web application firewall. As mentioned earlier, one of the difficulties with a shield-then-patch strategy is detecting vulnerabilities and building the WAF signatures to defend them. Coupling assessment tools or services with WAFs by feeding the assessment results to the firewall and having the WAF adjust its policy set accordingly, can make the firewall more effective. The intention is to bridge the gap between exploit discovery/announcement and deployment of tested patches in the application, by instructing the WAF to block access to the particular weakness. In this model the assessment determines that there is a vulnerability and feeds the information to the WAF. The assessment policy configures the WAF with what to look for and how to react. The WAF then dynamically incorporates the policy and protects the application.

Putting It All Together

In this section we will put all the pieces together and guide your considerations for designing a program that meets the needs of *your* organization. Web application security is not a "one size fits all" problem. The risks, size, and complexity of applications differ, the level of security awareness among team members varies, and most importantly the goals of each organization are different.

In order to offer practical advice, we needed to approach program development in terms of typical goals. We picked three use cases to represent common challenges organizations face with web app security, and will address each with an appropriate program model. We discuss a large enterprise looking to improve security across customer-facing applications, a mid-sized firm tackling a compliance mandate for the first time, and a mid-to-large organization dealing with security for internal applications. Each perspective has its own drivers and assumptions, and in each scenario different security measures are already in place, so the direction of each program is different. First we describe the environment for each case, then overall strategy and specific recommendations.

Large Enterprise with Customer Facing Web Applications

For our first scenario, let's consider a large enterprise with multiple customer-facing web applications. They evolved to offer core business functions and are a principal contact point with customers, employees, and business partners. Primary business drivers for security are fraud reduction, regulatory compliance, and service reliability. Secondary factors are breach preparedness, reputation preservation, and asset protection — all considerations for security spending. The question is not whether these applications need to be secured, but how. Most enterprises have a body of code with questionable security, and let's be totally honest here — these issues are flaws in your code. No single off-the-shelf product is going to magically make your application secure, so you invest not only in third-party security products, but also in improvements to your own development processes which improve the product with each new release.

We assume our fictitious enterprise has an existing security program and the development team has some degree of maturity in their understanding of security issues, but how best to address problems is up for debate. The company will already have a "security guy" in place, but the development organization is not tasked with security assessment and problem identification. Your typical CISO comes from a network security background, lacks secure development experience, and is not part of the code assessment effort. We find their security program includes vulnerability assessment tools, and they have conducted a code review for typical SQL injection and buffer overflow attacks. Overall, security is a combination of a couple third-party products and the security guy pointing out security flaws which are patched in upcoming release cycles.

Recommendations

The strategy is to include security within the core development process, shifting the focus from external products to internal products and employee training. Tools are selected and purchased to address particular deficiencies in team skills and organizational processes. Web application firewalls are retained to shield applications during patching efforts.

Training, Education, and Process Improvements: The area where we expect to see the most improvement is the skill and awareness of the web application development team. OWASP's top flaws and other sources point out issues that can be addressed by proper coding and testing … provided the team knows what to look for. Training helps staff find errors and problems during code review, and iteratively reduces flaws through the development cycle. The development staff can focus on software security and not rely on one or two individuals for security analysis.

Secure SDLC: Knowing what to do is one thing, but actually doing it is something else. There must be an incentive or requirement for development to code security into the product, assurance to test for compliance, and product management to set the standards and prioritize security. Otherwise security issues get pushed to the side while features and functions are implemented. Security needs to be part of the product specification, and each phase of the development process should provide verification that the specification is being met through assurance testing. This means building security testing into the development process and QA test scenarios, as well as re-testing released code. Trained development staff can provide code analysis and develop test scripts for verification, but additional tools to automate and support these efforts are necessary, as we will discuss below.

Heritage Applications: Have a plan to address legacy code. One of the more daunting aspects for the enterprise is how to deal with existing code, which is likely to have security problems. There are several possible approaches for addressing this, but the basic steps are 1) identification of problems in the code (code scanning, VA, and penetration testing), 2) prioritization on what to fix, and 3) planning how to fix individual issues. Common methods of addressing vulnerabilities include 1) rewriting segments of code, 2) method encapsulation (*e.g.,* interfaces), 3) supplementing existing code by building in run-time validation routines/methods within the execution path, 4) temporary shielding by WAF ("shield-then-patch"), 5) moving SQL processing and validation into databases, and 6) discontinuing use of insecure features. We recommend static source code analysis or dynamic program analysis tools for the initial identification. These tools are cost-effective and suitable for scanning large bodies of code to locate common risks and programming errors. They detect and prioritize issues, and reduce human error associated with tedious manual scanning by internal or external parties. Analysis tools also help educate staff about issues with certain languages and common programming patterns. The resulting arguments over what to do with 16k insecure occurrences of IFRAME are never fun, but acceptance of the problem is necessary before it can be addressed.

External Validation: Periodic external review through vulnerability assessment, penetration testing, or source code review, is highly recommended. Skilled unbiased professionals with experience in threat analysis often catch items which slip by internal scans, and can help educate development staff on different threat vectors. Plan on external penetration testing on a quarterly or biannual basis — the expertise and training of good consultants goes far beyond the basic threats, and trained humans monitoring the output of sophisticated tools are very useful for detecting weaknesses that a hacker could exploit. We recommend the use of static testing tools for internal testing of code during the QA sweep, with internal penetration testing just prior to deployment so they can fully stress the application without fear of impairing the production environment. Major releases should also undergo an external penetration test and review before deployment.

Blocking: This is one area that will really depend upon the specifics of your organization. In the enterprise use case, plan on using a Web Application Firewall. They provide basic protection and give staff a chance to remove security issues from the application. You may find that your code base is small and stable enough that you do not need WAF for protection, but for larger organizations this is not an option. Development and patching cycles are too long and cumbersome to counter threats in a reasonable timeframe. We recommend WAF + VA because in combination they can relieve your organization from much of the threat research and policy development work for firewall rules. If your staff has the skill and time to develop WAF policies specific to your organization, you get customized policies at slightly greater expense. WAF isn't cheap, so we don't make this recommendation lightly, but it provides a great deal of flexibility in how and when threats are dealt with today and as new threats evolve.

We recommend you take steps to improve security in every part of the development process. We focused on improvements to the initial phases of development lifecycle (requirements gathering, architecture, design) as efforts early in the process yield the greatest rewards. For these major applications that are already in production we also recommend external evaluations, and if budget allows,blocking. Vulnerability assessments and penetration testing are critical regardless of whether the application is new or old, and WAFs are important for applying quick patches for new vulnerabilities. The risks to large enterprises are greater, the issues to overcome are more complex, and the corresponding security investment must therefore be larger. Modifications and requirements added to the development process should be formally documented for each stage of an application's lifecycle — from development through ongoing maintenance — with checkpoints at major milestones. The security group shouldn't and can't be responsible for oversight of the inner workings of the development organization, but should help with audits and verify the proper process is followed and maintained.

Mid-sized Firm and PCI Compliance

If we are discussing web application security and compliance, odds are the discussion centers on the Payment Card Industry's Data Security Standard (PCI-DSS). No other compliance standard specifies steps to secure web applications like the PCI standard does. We can grouse about ambiguities and ways that it could be improved, but PCI is clearly the most widespread driver for web application security today, which is why our second use case is a mid-sized firm that needs to secure its web applications to satisfy PCI-DSS.

The profile for our company is a firm that generates a large portion of its revenue through Internet sales, which has become a Tier 3 merchant through recent growth. The commerce web site is relatively new (< 3 years) and the development team is small and not trained in security. Understanding the nuances of how criminals approach breaking code is not part of the team's skill set. PCI compliance is the mandate, and the team knows that they are both missing the basic requirements and susceptible to some kinds of attacks. The good news is that the body of code is small, and the web application accounts for a significant portion of the company's revenue, so management is supporting the effort.

In a nutshell, the PCI Data Security Standard is a security program specifically for companies that process credit card transactions for Internet commerce. In terms of compliance regulations, PCI-DSS requirements are clearer than most, including specific requirements for security tools and processes around credit card data. However, a company may also satisfy the spirit of the requirements in an alternate way if it can demonstrate that the concern has been addressed. We will focus on the requirements outlined in sections 6.6 & 11.3, but will refer to section 10 and compensating controls as well.

Recommendations

Our strategy focuses on education and process modifications to bring security into the development lifecycle. Additionally, we suggest assessment or penetration testing services to quickly identify areas of concern. Deploy a WAF to address the section 6.6 requirement immediately. Focus on the requirements to start, but plan for a more general program, and use compensating controls as your organization evolves. Use outside help and education to address immediate gaps, in both PCI compliance and more general application security.

Training, Education, and Process Improvements: Once again, we are hammering on education and training for the development team, including project management and quality assurance. While it takes time to come up to speed, developer awareness helps keep security issues out of the code and is cost-effective (due to improved code quality and reduced need for fixes later in the process). Altering the processes to accommodate fixing the code is nearly free, and code improvements become part of day-to-day efforts — this improves overall quality, not just security. With a small code base, education and training are easy ways to reap significant benefits as the company and code grow.

External Help: Make friends with an auditor, or hire one as a consultant to help prepare and navigate PCI-DSS. While this is not a specific recommendation for any single requirement in PCI; auditors provide an expert perspective, help address some of the standard's ambiguity, and assist in strategy and trade-off evaluations to avoid costly missteps.

Section 11.3.2: 11.3 mandates penetration testing of the network and the web application. In this case we recommend external penetration testing as an independent examination of the code. It is easy to recommend penetration testing — not only because it is required in the DSS specification, but additionally because the independent expert review of your application behavior will closely mimic the approach hackers take. We also anticipate budget will require you to choose between WAF and code review in section 6.6, so this helps provide the necessary coverage. Should you use source code reviews, one could argue (likely unsuccessfully) that they serve as a compensating control for this section, but our recommendation is to stick with external penetration testing. External testers provide much more than just a list of specific flaws, but also identify risky or questionable application behaviors in a near-production environment.

Section 6.6: Our biggest debate internally was whether to recommend a web application firewall or expert code review to address section 6.6 of the PCI specification. The PCI Security Standards Council recommends that you do both, but it is widely recognized that this is prohibitively expensive. WAF provides a way to quickly meet the letter of 6.6's requirement (if not its spirit) provides basic monitoring, and serves as a flexible platform for blocking future attacks. The counter-arguments are significant and include cost, work required to customize policies for the application, and false positives and negatives. Alternatively, a code review by qualified security experts can identify weaknesses in application design and code usage, and assist in education of the development team by pointing out specific flaws. Outside review is a very quick way to assess where you are and what you need. Downsides of external review include cost, time to identify and fix errors, and that a constantly changing code base presents a moving target and thus requires repeated examinations.

Our recommendation here is to deploy a WAF. Engaging a team of security professionals to review the code is an effective way to identify issues, but much of its value overlaps with the requirement of section 11.3.2 — periodic penetration testing of the application. The time to fix identified issues (even with a small-to-average body of code), with a development organization just coming to terms with security issues is too great to meet PCI requirements in a timely fashion. Note that this recommendation is specific to this particular scenario — in other PCI audit scenarios, with a more experienced staff or a better handle on code quality, we might have a different recommendation.

Monitoring: Database Activity Monitoring (DAM) is a good choice for Section 10 compliance — specifically monitoring all access to credit card data. Web applications use relational database back ends to store credit card numbers, transactions, and related data. DAM products that capture all network and console activity on the database platform provide a focused and cost-effective audit for all access to cardholder data. Consider this option for providing an audit trail for auditors and security personnel.

Internal Web Application Development

Our last use case is an internal web application that serves employees and partners within a mid-to-large business. While this may not sound like a serious problem, given that companies have on average 1 internal application (web or traditional client/server) per 100 employees, even mid-sized companies have incredible exposure in this area. Using data from workflow, HR, accounting, business intelligence, sales, and other critical IT systems, these internal applications support employees and partners alike. And with access to pretty much all data within the company, security and integrity are major concerns. A common assumption is that these systems behind the perimeter firewall are not exposed to the same types of attacks as typical web applications, but this assumption has proven disastrous in many cases.

Investment here is motivated by fraud reduction, breach reduction, and avoidance of notification costs — and possibly by compliance. You may find it difficult to sell to management if there is not a compliance mandate and hasn't already been

a breach, but if basic perimeter security is breached these applications need some degree of resiliency rather than blind confidence in network security and access control.

Recommendations

Determine basic security of the internal application, fix serious issues, and leverage education, training, and process improvements to steadily improve the quality of code. We will assume that budget for security in this context is far smaller than for external-facing systems, so look to cooperate between groups and leverage tools and experience.

Vulnerability Assessment and Penetration Testing: Scanning web applications for significant security, patch, and configuration issues is a recommended first step in checking for glaring problems. Assessment tools are a cost-effective way to establish baseline security and ensure adherence to minimum best practices. Internal penetration testing will help determine the overall potential risk and prioritization, but be extremely cautious with live applications.

Training, Education, and Process Improvements: These may be even more important in this scenario than in our other use cases, where the business justification provides clearer incentive to invest in security, because internal web applications may not get the same level of attention. For applications that have a captive audience, developers have greater control over the types of environments that they support and what authentication can be required. Use this flexibility to your advantage. Training should focus on common vulnerabilities within the application stack being used, and give critical security errors the same attention other top priority bugs would receive. Verify that these issues are tested, either as part of the VA sweep, or as a component of regression testing.

Monitoring: Monitoring for suspicious activity and system misuse is a cost-effective way to detect issues and react to them. We find WAF solutions are often too expensive for deployment across hundreds of internal applications distributed across a company, and a more cost-effective approach to collecting and analyzing activity is highly recommended. Monitoring software that plugs into the web application is often very effective for delivering some intelligence at low cost, but the burden of analyzing the data then falls on development team members. Database Activity Monitoring can effectively focus on critical information at the back end and is more mature than Web Application Monitoring.

Conclusion

Our recommendations really depend upon the specifics of the situation and the organization. We chose use cases here to demonstrate how the motivating factors — combined with the current state of web security — guide the selection of tools, services, and process changes. And as the applications support multiple business functions, this process needs to be reviewed by business operations and project managers, along with technical staff. This means that the design and requirements phases are more difficult, but the entire effort moves forward with the understanding and support of the group.

We found in every case that security as part of the overall development process is the most cost-effective and least disruptive to normal operations, and is our principal recommendation for each scenario. However, as transformation of a web application does not happen overnight, we rarely have the luxury of waiting for the development team to address all security issues — in the meantime, external third-party services and products are invaluable for dealing with the immediate challenges.

Who We Are

About the Authors

Rich Mogull, Founder

Rich has over 17 years experience in information security, physical security, and risk management. Prior to founding Securosis, Rich spent 7 years as one of Gartner's leading security analysts, where he advised thousands of clients, authored dozens of reports, and was consistently rated one of Gartner's top international speakers. He is well known for his work on data security technologies and has covered issues ranging from vulnerabilities and threats, to risk management frameworks, to major application security. Rich is the Security Editor of *TidBITS*, a monthly columnist for *Dark Reading*, and a frequent contributor to publications ranging from *Information Security Magazine* to *Macworld*.

Adrian Lane, Senior Security Strategist

Adrian is a Senior Security Strategist with 22 years of industry experience, bringing over a decade of C-level executive expertise to the Securosis team. Mr. Lane specializes in database architecture and data security. With extensive experience as a member of the vendor community (including positions at Ingres and Oracle), in addition to time as an IT customer in the CIO role, Adrian brings a business-oriented perspective to security implementations. Prior to joining Securosis, Adrian was CTO at database security firm IPLocks, where he was responsible for product and technology vision, market strategy, PR, and security evangelism. Mr. Lane also served as Vice President of Engineering at Touchpoint, for three years as CIO of the brokerage CPMi, and for two years as CTO of the security and digital rights management firm Transactor/Brodia. Mr. Lane is a Computer Science graduate of the University of California at Berkeley with post-graduate work in operating systems at Stanford University.

About Securosis

Securosis, L.L.C. is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services.

We provide services in four main areas:

- Publishing and speaking: Including independent objective white papers, webcasts, and in-person presentations.
- Strategic consulting for end users: Including product selection assistance, technology and architecture strategy, education, security management evaluations, and risk assessments.
- Strategic consulting for vendors: Including market and product analysis and strategy, technology guidance, product evaluations, and merger and acquisition assessments.
- Investor consulting: Technical due diligence including product and market evaluations, available in conjunction with deep product assessments with our research partners.

Our clients range from stealth startups to some of the best known technology vendors and end users. Clients include large financial institutions, institutional investors, mid-sized enterprises, and major security vendors.

Securosis has partnered with security testing labs to provide unique product evaluations that combine in-depth technical analysis with high-level product, architecture, and market analysis.