Six Essential Elements of Web Application Security

Cost Effective Strategies for Defending Your Business

QUALYS®

An Introduction to Defending Your Business Against Today's Most Common Cyber Attacks

When web applications are breached, enormous amounts of sensitive business data can be lost. According to Verizon's 2014 Data Breach Investigations Report, web application attacks more than doubled in 2013 to become the #1 cause of security incidents.

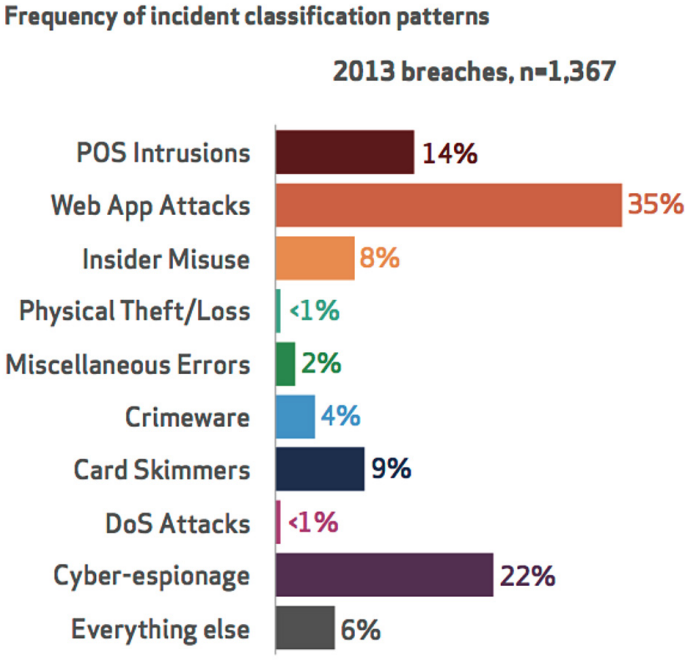

Figure 1: Verizon 2014 Data Breach Investigation Report

These types of attacks can occur at organizations of all sizes and levels of IT sophistication, and can affect tremendous amounts of data. In the spring and summer of 2013 alone, there were numerous high-profile web-related security incidents. Attackers were able to steal passwords from a site run by NASDAQ, the second breach at NASDAQ in recent years. Shortly thereafter, Apple's developer website was breached, placing registered developer names and mailing addresses at risk.

Web applications are popular targets because:

- They are accessible to almost anybody in the world.
- They are a conduit to an enormous amount of valuable data.
- They are commonly riddled with weaknesses.

If you're like many organizations, your IT teams and developers have very little time or resources to devote to performing security tests (particularly manual ones). When security testing does get done, it tends to be focused on the highest-profile web applications, leaving the security of other apps to chance. Even then, testing can be sporadic, enabling vulnerabilities to creep in unnoticed and create opportunities for exploits.

The financial impact of such exploits is substantial: according to the Ponemon Institute's 2013 Cost of a Data Breach Study, U.S. breaches cost $188 per record stolen, with an average total cost of $5.4 million per incident.

Fortunately, most web application attacks follow a small number of patterns.

The most common classes of web application vulnerabilities

Like other application flaws, web application security defects arise during software development. The Open Web Application Security Project (OWASP) Top 10 provides the de facto standard for categorizing web app vulnerabilities (see appendix). The most common types include:

Cross-Site Scripting (XSS)
Cross-Site Scripting (XSS) is one of the most widely-found and dangerous vulnerabilities in web apps. XSS can have a big impact on your organization because it enables attackers to send untrusted code to users' web browsers under the guise of your business's legitimate app. This enables attackers to execute scripts in victims' browsers to hijack a session or download malware to take full control of their system. XSS vulnerabilities have been found by researchers to exist in the websites of security vendors, marketplaces, payments providers, merchants, and social networks.

SQL Injection
Injection attacks come in many different flavors, including: SQL injection, command injection (inserting system commands into a form field), and many others. SQL injection attacks are among the most widely known. Attackers send malformed inputs to your application (for example adding extra characters to the ends of a type-in field), which then gets passed to a database. The maliciously-formatted input tricks the database into returning excess information or performing unwanted actions. This type of attack has been used to expose hundreds of millions of records containing personally-identifiable information (PII) and credit card data; it can also be used to modify or delete sensitive data, sometimes without your ever knowing.

Fortunately, you can combat these and other vulnerabilities by following a few straightforward best practices and employing new automated technologies.

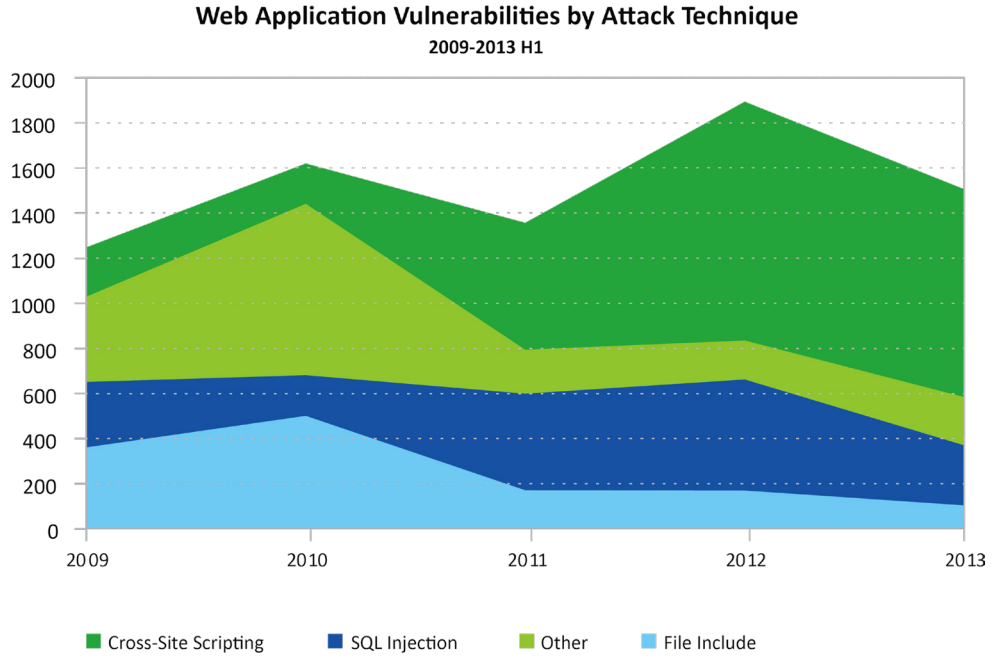

Figure 2: IBM X-Force 2013 Mid-Year Trend and Risk Report


All too often, IT teams and developers are limited in the time and resources they can devote to security tests.

Six Essential Elements of Web Application Security

Like all competitive businesses, your application developers and operation teams are constantly under pressure to move quickly. Everybody wants their application security efforts to be effective, but only if they don't unduly impede workflow or drive up costs. To balance these potentially-competing objectives, industry-leading organizations often use the following six elements in their approach to web application security.

Essential Element #1
Active Sponsorship

A visible, executive advocate or management sponsor is crucial to the success of any web application security initiative. Web application security requires ongoing collaboration among the involved teams: business leaders, IT leaders, development, operations, and security groups. Having demonstrable leadership backing makes it easier to put that collaboration in place and obtain necessary resources.

Essential Element #2
Development Discipline

Application security can't be bolted on, it has to be baked into the development process and enforced at key milestones. Security should be explicitly considered when the technical requirements of the application are being defined, during coding, in the QA phase, and when applications are put into production.

Essential Element #3
Developer Training

There's no way around it: secure coding is a skill unto itself that requires developer training. When developers, server admins, and others build and deploy applications, it's essential that they be aware of where security flaws can come from. A recent study by the Ponemon Institute found that more than half of the developers questioned had no formal training in application security. Making applications resilient to attack is tough if you don't know what to look for, even if you're equipped with the right tools (which most organizations still are not).

Essential Element #4
Threat Modeling

Before you can protect your applications, data and other IT assets, you have to understand the fundamentals behind a potential attack. In particular, it's critical think about who might have the motive, opportunity, and means to attack.

For example, would the data in your application be of interest to foreign governments, competitors, or criminals for financial gain? Will your application be accessed from public networks or limited to tightly-controlled corporate environments? What technologies will be needed (such as authentication, encryption, data storage, and integration with other systems) and how might they be exploited? Threat models bring this information together so that developers can think ahead and design the application from the start to be more secure.

Essential Element #5
Automated Testing

While a great many problems can be avoided by focusing on security during development, some vulnerabilities will inevitably sneak in. This is where having the right application security tools and technology makes a huge difference.

Finding security problems in today's web applications requires much more than just doing code reviews. New generations of manual and automated tools now enable your developers, testers and operations personnel to make apps more secure than ever before:

Static Application Security Testing (SAST) tools examine the raw "static" application code for specific types of programing errors, such as: logic bombs, SQL injection, buffer overruns, and other flaws. A qualified analyst who is familiar with the code and the business processes being implemented typically examines the results. This interactive review works particularly well during development when analyzing applications that are written in a support language and can help pinpoint where specific problems lie.

Dynamic Application Security Testing (DAST) technologies analyze the output of applications "dynamically" as they're running, interacting with apps the way an attacker would. This makes it possible to test for conditions that are only detectable during actual use (such as issues resulting from access control). DAST can be used on apps written in any language, throughout the app's lifecycle: in development, QA and production. It also enables the infrastructure underneath the web app to be tested, not just the portions for which code is available.

Organizations that employ both approaches often use manual SAST tools to spot-check the logic in their most important apps and automated DAST systems to repeatedly test the behavior. Now, new cloud-based DAST systems enable you to test large numbers of applications in a short period of time, without the costs or burdens of setting up on-premises software. This makes it feasible for you to do ongoing testing of all your applications —not just a few—even in live environments.

Essential Element #6
Attack Blocking

In the ideal world, applications would always be perfectly secure; realistically, bugs happen and vulnerabilities inevitably appear. But fixing and deploying changes to applications takes time—when it can be done at all.

During this time, your business or your customers are open to attack. This is where Web Application Firewalls (WAFs) come in: they can give you time to fix problems by automatically blocking certain web app attacks, like Cross-Site Scripting.

WAFs sit in front of your web applications, examining the HTTP traffic sent between the user's browser and your application. Requests and replies that look inappropriate are blocked or modified. Unusual patterns, such as malicious values attached to the end of input strings, can be stripped out so that applications only see valid input from the user. WAFs can also be used to limit access from undesirable or suspicious networks and alter the way browsers and apps communicate (for example, by adding the Strict-Transport-Security header to instruct browsers to always connect using encrypted HTTPS sessions), all without having to make changes in the application or reconfigure each web server.

WAFs also can shield your organization against vulnerabilities found in third-party applications while vendors are working on fixes. You can even use WAFs to protect your business when using shared Software-as-a-Service (SaaS) applications whose terms of service prohibit you from performing vulnerability scans or whose vendors are not fast enough in applying important security fixes to their service.

Conclusion

Planning and cloud-based automation make global web application security possible.

Just as there is no single "silver bullet" in IT security, there is no one way to mitigate all web application vulnerabilities. Fortunately, the approach used by many successful organizations can put you on the right track.

Start by laying a foundation of executive support, instilling security into development processes, training software writers to avoid security problems, and identifying where threats are likely to come from. With this knowledge, you'll be able to take advantage of the new generation of testing and protection technologies such as SAST, DAST and WAF tools.

The simplicity and cost-effectiveness of cloud-based solutions, in particular, make high-end protection possible where it never was before. Now, you can use automated vulnerability assessment and web application firewalls, accessed and managed directly from the cloud, to secure all of your applications – anywhere around the globe.

Together, these essential practices and technologies can help you and your business dramatically reduce the risk of web application breaches.

For more information about Qualys and how its QualysGuard Cloud Platform integrated suite of security and compliance solutions can help your organization secure your web applications and reduce the risk of breaches, or to try QualysGuard Web Application Scanning (WAS) or QualysGuard Web Application Firewall (WAF) with your own applications, please visit qualys.com/was or qualys.com/waf.

The OWASP 2013 Top 10
Web Application Vulnerabilities

A1—Injection

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 data without proper authorization.

A2—Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

A3—Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface websites, or redirect the user to malicious sites.

A4—Insecure Direct Object References

A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key. Without an access control check or other protection, attackers can manipulate these references to access unauthorized data.

A5—Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

A6—Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A7—Missing Function Level Access Control

Most web applications verify function level access rights before making that functionality visible in the UI. However, applications need to perform the same access control checks on the server when each function is accessed. If requests are not verified, attackers will be able to forge requests in order to access functionality without proper authorization.

A8—Cross-Site Request Forgery (CSRF)

A CSRF attack forces a logged-on victim's browser to send a forged HTTP request, including the victim's session cookie and any other automatically included authentication information, to a vulnerable web application. This allows the attacker to force the victim's browser to generate requests the vulnerable application thinks are legitimate requests from the victim.

A9—Using Components with Known Vulnerabilities

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

A10—Unvalidated Redirects and Forwards

Web applications frequently redirect and forward users to other pages and websites, and use untrusted data to determine the destination pages. Without proper validation, attackers can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

Source: OWASP (www.owasp.org/index.php/Top_10_2013-Top_10)