ISSN (e): 2250 – 3005 || Volume, 05 || Issue, 06 || June – 2015 ||
International Journal of Computational Engineering Research (IJCER)

Emerging Trend of Security Testing to Make Application Robust

Mr. Shiv kumar Goel[1], Prasad Kuvalekar[2]
[1]Professor in the Department of MCA, VESIT Mumbai, India.
[2]Post-Graduate Student in the Department of MCA, VESIT Mumbai, India.

ABSTRACT:

Typical security requirements may include specific elements of confidentiality, integrity, authentication, availability, authorization and non-repudiation. Actual security requirements tested depend on the security requirements implemented by the system. Security testing as a term has a number of different meanings and can be completed in a number of different ways. As such a Security Taxonomy helps us to understand these different approaches and meanings by providing a base level to work from.

Keywords: *Attacks, Security testing, software assurance, SQL injection, Vulnerabilities, Web application*

I. INTRODUCTION

Security testing is necessary because it has a distinct relationship with software quality. Just because software meets quality requirements related to functionality and performance, it does not necessary mean that the software is secure. The inverse however is true: i.e. software that is secure is software with added resiliency, thus software of higher quality. Poor architecture and implementation of the web application cannot assure the CIA aspect of software assurance, which would otherwise indicate the resiliency quality of the software.

Fig. 1 Disclosure Strategies.

Vulnerabilities in web applications are now the largest source of enterprise security attacks. Web application vulnerabilities accounted for over 55% of all vulnerabilities disclosed in 2010, according to an IBM X-Force study. That may be the tip of the iceberg as the study includes only commercial web applications.1

Stories about compromised sensitive data frequently mention culprits such as “cross-site scripting,” “SQL injection,” and “buffer overflow.” Vulnerabilities like these often fall outside the traditional expertise of network security managers. The relative obscurity of web application vulnerabilities thus makes them useful for attacks. As many organizations have discovered, these attacks will evade traditional enterprise network defenses unless you take new precautions.

Security Testing

Fig.2 Top hacking Countries

II. LITERATURE SURVEY

2.1 Overview of Web Application Security

Attacks on vulnerabilities in web applications began appearing almost from the beginning of the World Wide Web, in the mid-1990s. Attacks are usually based on fault injection, which exploits vulnerabilities in a web application‟s syntax and semantics. Using a standard browser and basic knowledge of HTTP and HTML, an attacker attempts a particular exploit by automatically varying a Uniform Resource Indicator (URI) link, which in turn could trigger an exploit such as SQL injection or cross-site scripting.

Fig.3 SQL injection

A significant number of attacks exploit vulnerabilities in syntax and semantics. You can discover many of these vulnerabilities with an automated scanning tool. Logical vulnerabilities are very difficult to test with a scanning tool; these require manual inspection of web application source code analysis and security testing. Web application security vulnerabilities can stem from misconfigurations, bad architecture, or poor programming practices within commercial or custom application code. Vulnerabilities may be in code libraries and design patterns of popular programming languages such as Java, .NET, PHP, Python, Perl, and Ruby. These vulnerabilities can be complex and may occur under many different circumstances. Using a web application firewall might control effects of some exploits but will not resolve the underlying vulnerabilities.

To protect Web applications against attacks, enterprises should employ generic preventive approaches as well as targeted technologies.

Typical Web application attacks

A Web application's specific vulnerabilities should dictate the technology you use to defend it. Figure 1 shows many points within a system that might require protection. Often, it is best to employ generic countermeasure concepts first to help ensure that you choose the technology best suited to your needs rather than one that claims to counter the latest hacking techniques.

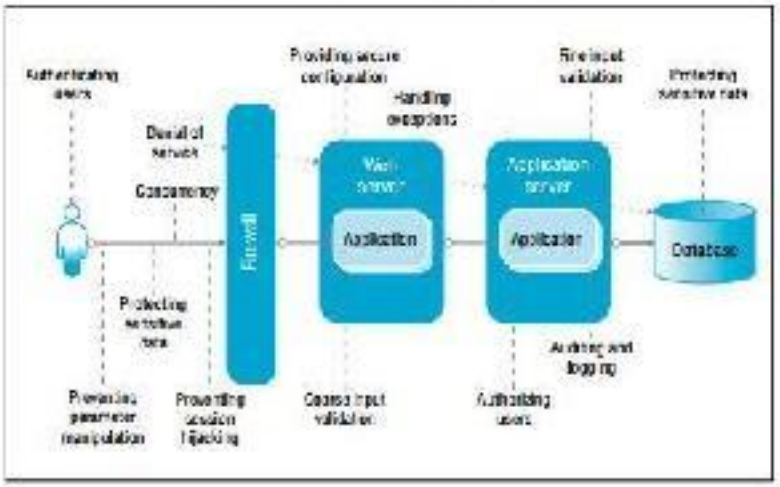


Fig.4 Web Application Attacks

2.2 Types of Web Application Vulnerabilities

Web applications may have any of two dozen types of vulnerabilities. Security consultants who do penetration testing may focus on finding top vulnerabilities.

2.2.1 Authentication – stealing user account identities

>> Brute Force attack automates a process of trial and error to guess a person‟s username, password, credit-card number or cryptographic key.

>> Insufficient Authentication permits an attacker to access sensitive content or functionality without proper authentication.

>> Weak Password Recovery Validation permits an attacker to illegally obtain, change or recover another user‟s password.

2.2.2 Authorization – illegal access to applications

>> Credential / Session Prediction is a method of hijacking or impersonating a user.

>> Insufficient Authorization permits access to sensitive content or functionality that should require more access control restrictions.

>> Insufficient Session Expiration permits an attacker to reuse old session credentials or session IDs for authorization.

>> Session Fixation attacks force a user‟s session ID to an explicit value.

2.2.3 Client-side Attacks – illegal execution of foreign code

>> Content Spoofing tricks a user into believing that certain content appearing on a web site is legitimate and not from an external source.

>> Cross-site Scripting (XSS) forces a web site to echo attacker-supplied executable code, which loads into a user‟s browser.

2.2.4 Command Execution – hijacks control of web application

>> Buffer Overflow attacks alter the flow of an application by overwriting parts of memory.

>> Format String Attack alters the flow of an application by using string formatting library features to access other memory space.

>> LDAP Injection attacks exploit web sites by constructing LDAP statements from user-supplied input.
>> OS Commanding executes operating system commands on a web site by manipulating application input.

2.2.5 Information Disclosure – shows sensitive data to attackers

>> Directory Indexing is an automatic directory listing / indexing web server function that shows all files in a requested directory if the normal base file is not present.

>> Information Leakage occurs when a web site reveals sensitive data such as developer comments or error messages, which may aid an attacker in exploiting the system.

>> Path Traversal forces access to files, directories and commands that potentially reside outside the web document root directory.
>> Predictable Resource Location uncovers hidden web site content and functionality.

2.2.6 Logical Attacks – interfere with application usage

>> Abuse of Functionality uses a web site‟s own features and functionality to consume, defraud, or circumvent access control mechanisms.

>> Denial of Service (DoS) attacks prevents a web site from serving normal user activity.

>> Insufficient Anti-automation is when a web site permits an attacker to automate a process that should only be performed manually.

>> Insufficient Process Validation permits an attacker to bypass or circumvent the intended flow of an application.

III. TESTING APPROACH

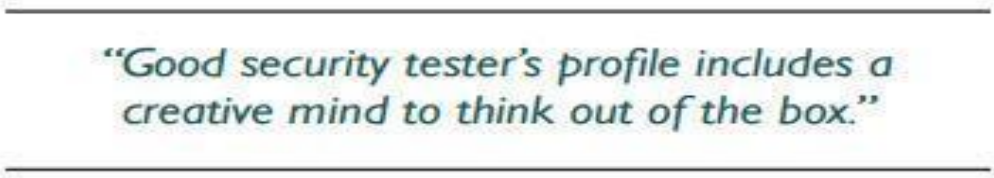
Criterea	Security Testing Approach	
	Black Box	White Box
Determination of root cause	Most likely to address the symptoms than the root cause.	Exact line of code or design issue causing the vulnerability can be identified.
Extent of code coverage	Limited as the analysis is behavioral; not all code paths may be covered.	Greater as the source code and configuration is available for review.
Detection of logic flaws	Not knowing the normal behavior of the software; anomalous behavior may not necessarily indicate flaws in logic.	The availability of design and architectural documents besides code can be used to detect logic flaws.
Issues with deployment	Assessment can be performed in pre- as well as post-deployment production or production-like simulated environment, giving insight into any potential issues after deployment.	Assessment is performed in pre-deployment environments usually providing limited issues pertaining to configuration and change management.

Fig.5 Security Testing Approach

IV. SECURITY TESTER

Security testers are a breed all their own. In addition to an anti-developer attitude, a good security tester‟s profile includes a creative mind to think out of the box. A good security tester thrives and excels when posed with challenges.

Security Testing



They have a constant thirst for learning newer attack techniques and coming up with all possible combinations of attack as they seek to exploit weaknesses in the software or system. They usually have selfdriven personalities and their motivations are often related to ego than materialistic pursuits. With appropriate training and skills on software security throughout the SDLC, these testers become very valuable to the organization.

V. TYPES OF SECURITY TESTING

Following are the types of security testing as per Open Source Security Testing methodology manual. They are explained as follows:

Vulnerability Scanning:

This is done through automated software to scan a system against known vulnerability signatures.

Security Scanning:

It involves identifying network and system weaknesses, and later provides solutions for reducing these risks. This scanning can be performed for both Manual and Automated scanning.

Penetration testing:

This kind of testing simulates an attack from malicious hacker. This testing involves analysis of a particular system to check for potential vulnerabilities to an external hacking attempt.

Risk Assessment:

This testing involves analysis of security risks observed in the organization. Risks are classified as Low, Medium and High. This testing recommends controls and measures to reduce the risk.

Security Auditing: This is internal inspection of Applications and Operating systems for security flaws. Audit can also be done via line by line inspection of code

Ethical hacking:

It's hacking Organization Software systems. Unlike malicious hackers who steal for their own gains, the intent is to expose security flaws in the system.

Posture Assessment: This combines Security scanning, Ethical Hacking and Risk Assessments to show an overall security posture of an organization.

Vulnerability Assessment:

This is a security audit and privilege access and administrator assistance is required for configuration audit. This is done directly on the system with physical and logical access. System configuration checking and vulnerability scanning is performed to find out weaknesses, vulnerabilities and mis-configuration in the target hosts.

Penetration Testing:

Penetration Testing (PT) is normally done remotely from public domain (Internet) and also can be done from internal network to find out exploitable vulnerabilities from internal network. No privilege access is required. Series of testing conducted like information gathering from public domain, port scanning, system fingerprinting, service probing, vulnerability scanning, manual testing, password cracking etc. using state-of-the-art tools (commercial and open source) and techniques used by hackers with objective to unearth vulnerabilities and weaknesses of the IT infrastructure.

Application Security Assessment:

Different software testing techniques are employed to unearth application security vulnerabilities, weaknesses and concerns related to Authentication, Authorization, Session Management, Input/output Validation, Processing Errors, Information Leakage, Denial of Service etc. Typical issues which may be discovered in an application security audit include Cross-site scripting, Broken ACLs/Weak passwords, Weak session management, Buffer overflows, Forceful browsing, CGI- BIN manipulation, Form/hidden field manipulation, Command injection, Insecure use of cryptography, Cookie poisoning, SQL injection, Server mis-configurations, Well-known platform vulnerabilities, Errors triggering sensitive information leak etc. For web applications OWASP (Open Web Application Security Project) guidelines is used for the assessment. All the assessment are carried out using both state-of-the-art tools and manual testing methods.

VI. STRATEGIES

Since security testing is not an optional activity, it is imperative to incorporate this vital activity into the SDLC. Security testing should be made an integral part of the overall testing process. It must be included into the scope of the software development project, even before the code for that software is written. A proven strategy is to incorporate a library of security tests into the enterprise project template (assuming you already have one) and maintain these tests in the centralized test and bug tracking database/software. This will ensure that all projects will, at a bare minimum, automatically inherit any tests that need to be run as part of the project. Another strategy is to start generating the test cases during the requirement phase of the SDLC itself. When the requirements are known, there is no need to wait until code is complete to generate all the test cases. This proactive approach to testing gives time for both functional and security test cases to be generated in advance and the testing phase can be used efficiently to conduct these test cases.

VII. USING TOOLS TO IMPLEMENT WEB APPLICATION SECURITY

There are many kinds of scanners and other tools for IT security, audit, and operations. Our focus here is the web application security (WAS) scanner. Its fundamental purpose is to automatically examine custom web applications that respond to dynamic web page requests with protocols like hypertext transfer protocol (HTTP).

Some scanners use the „point-and-shoot‟ method of operation without imposing particular parameters. Others need to be „trained‟ or configured for a particular environment. However it works, the scanner should be able to find all vulnerabilities in all web applications with a high degree of accuracy. To miss these, or present false positive or false negative identifications is to leave a web application exposed to exploitation.

VIII. CONCLUSION

Software assurance is comprised of reliability, recoverability, and resiliency aspects of the software. Software testing must address all of these. Without testing the software for its security capabilities, it is only a matter of time before software will be exploited (hacked). Software testing for functionality should always be augmented with security testing for resiliency. Security is an attribute of quality, and software that is less prone to getting hacked can be said to b e of higher quality than software that doesn‟t take security into account. Security testing can be used to achieve software assurance – akin to setting up the chess board so that you are able to defend your organization and avoid checkmate.

REFERENCES

[1] "Improving Web Application Security: Threats and Countermeasures". Microsoft Corporation. June 2003.
[2] "Testing and Comparing Web Vulnerability Scanning Tools for SQL Injection and XSS Attacks" (PDF). Fonseca, J.; Vieira, M.; Madeira, H., Dependable Computing, IEEE. Dec 2007.
[3] "The Web Hacking Incidents Database". WASC. January 2010.
[4] "Web Application Vulnerability Scanners". NIST.
[5] "Source Code Security Analyzers". NIST.
[6] "Web application firewalls for security and regulatory compliance". Secure Computing Magazine. February 2008.