Volume : 1, Issue : 1
January - June 2011

ISSN : 2229 - 3515

international journal of ADVANCES IN SOFT COMPUTING TECHNOLOGY

Editor-in-Chief
Dr.Vaka Murali Mohan

Published by
BHAVANA RESEARCH CENTER

Impact of Multiple attacks and damages on web based applications and security testing

Pratap Singh, S[1]* and Ekambaram Naidu, M[2]

1.TRR College of Engineering, Inole (V), Patancheru (M), Hyderabad, AP, INDIA.
2.TRR College of Engg. & Tech, Inole (V), Patancheru (M), Hyderabad, AP, INDIA.

KeyWords:
web application, web security, HTTP attacks, Secure Sockets Layer (SSL), firewalls.

Abstract: *Now a days, many Business to Business (B2B), Customer to Business (C2B) and Business to Customers (B2C) transactions are being done through web based applications. This method of web based application's service is attracting the attention of both small and medium-sized businesses and to those providers of application-based services because of its easier access, more affordable fees, and greater efficiency in delivering services to massive numbers of receivers. However, this web based applications needs to face several technical challenges, among which Data security is of top priority for most users because their data reside on remote servers whose resources are shared among different peoples over the globe. So there is a great need of Data Security and applications security. In this paper, we discuss about what you can do to protect your organizations sensitive data, and also we discuss an approach for improving your organization's Web application security.*

1. Introduction:

As businesses grow increasingly dependent upon Web applications, these complex entities grow more difficult to secure. Most companies equip their Web sites with firewalls, Secure Sockets Layer (SSL), and network and host security, but the majority of attacks are on applications themselves - and these technologies cannot prevent them. In the Open System Interconnection (OSI) reference model, every message travels through seven network protocol layers. The application layer at the top includes HTTP and other protocols that transport messages with content, including HTML, XML, Simple Object Access Protocol (SOAP) and Web services.

This paper focuses on application attacks carried by HTTP HTTP attacks are one the most popular hacking techniques. Hackers chiefly target a HTTP request and manipulate or modify the requests to cause the requisite damage. The attacks are usually performed using HTTP port 80 or other HTTP communications. To carry out effective and hassle-free web communication, HTTP is the most widespread protocol utilized today. Thus, web attacks related to HTTP are the most common ones occurring in the online sector.

*** Mr. S. Pratap Singh**
Associate Professor
Dept. of Computer Science & Engineering
TRR College of Engineering
Inole (v), Patancheru (m), Hyderabad, AP
Ph. No.: 91-9701457154
E-Mail: pratap_545@yahoo.co.in

A web server must control port 80 to transfer an HTTP request for a web page in order to operate the website. Since the requests are processed through HTTP, attackers manipulate or change these HTTP requests to gain entry into the web server. Teamed with the legitimate entry, they get the 'green signal' to bypass firewalls and several other security standards. Easy access to the web server assists them to pose any kind of attack. Defectively written applications are more susceptible to such attacks. HTTP attacks bear no classification as such. However, HTTP attacks are possibly categorized into 'buffer overflow', 'SQL Injection', and 'HTTP request smuggling attacks', and more as shown in figure 1

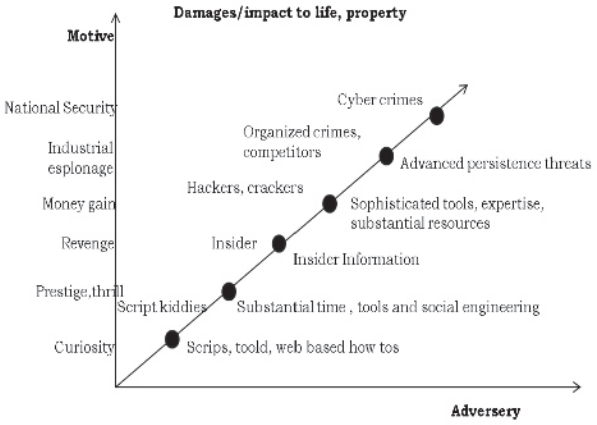

Fig.1. Relationship b/w the web based Applications attacks and adversary to the motivation of attack.

A web application security risk is multiplying and presents a number of unique security challenges. One is exposure because web applications reach millions of users, they also reach millions of potential hackers. Web applications stretch across multiple infrastructure tiers and incorporate many process layers, elements that expose them to a wide range of prospective attackers.

As web applications get more complex, so do their vulnerabilities; as they become more useful and pervasive, they become higher value targets. And cybercriminals are taking note.

2. Attacks and Damages/problems:

As the ensuing consequences of HTTP attacks result in the easy admission of the hacker into the web server, thus allowing him/her to cause immense damage. He/she can delete information, steal data or add info. He/she can cause endless harm to a website and even go to the extent of closing it down. If a website gets affected, it can result in serious damage to its online business, tarnish the website's image or make the business face huge financial loss. Thus, to prevent such mishaps, an enterprise must conduct thorough check of the user input data prior to transferring them for some other process. Moreover, constant examination during the development cycle for any vulnerability will expose the existing weak points.

2.1 Buffer overflow: A buffer overflow is an exploit that takes advantage of a program that is waiting on a user's input. There are two main types of buffer overflow attacks: stack based and heap based. Heap-based attacks flood the memory space reserved for a program, but the difficulty involved with performing such an attack makes them rare. Stack-based buffer overflows are by far the most common.

2.2 SQL Injection: A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. SQL injection attacks allow attackers to spoof identity, tamper with existing data, cause repudiation issues such as voiding transactions or changing balances, allow the complete disclosure of all data on the system, destroy the data or make it otherwise unavailable, and become administrators of the database server A successful SQL injection exploit can read sensitive data from the database, modify database at a (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to effect the execution of predefined SQL commands.

2.3 Cross Site Scripting (XSS): XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.

2.4 Malicious File Execution: Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework which accepts filenames or files from users.

2.5 Insecure Direct Object Reference: A direct object reference occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, database record, or key, as a URL or form parameter. Attackers can manipulate those references to access other objects without authorization.

2.6 Cross Site Request Forgery (CSRF): A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, This then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.

2.7Information Leakage and Improper Error Handling: Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.

2.8 Broken Authentication and Session Management: Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.

2.9 Insecure Cryptographic Storage: Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.

2.10 Insecure Communications and failure to Restrict URL Access: Frequently, an application only protects Sensitive functionality by preventing the display of links or URLs to unauthorized users. Attackers can use this weakness to access and perform unauthorized operations by accessing those URLs directly. This are the few of the attacks, apart from this types of attacks there are many more.

3. Preventive measures: In order to provide the security to web based applications some of the preventive measures against the type of attack as has to follow as shown in the figure 2.

3.1 Guidelines for providing security for Web applications: By using security-specific processes to create applications, software development teams can guard against security violations like those shown in table 1. Specifically, you can apply several basic guidelines to existing applications and new or reengineered applications throughout your process to help achieve greater security and lower remediation costs, such as:

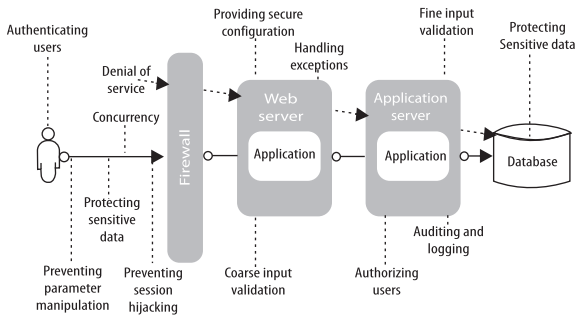

Fig.2. Preventive measures against the web based attacks.

Table 1: Web application security preventives

Description	Common causes	Preventive measures
Impersoning:		
Typing a different user's credentials or changing a cookie or parameter to impersonate a user or pretend that the cookie originates from a different server	* Using communications based authentication to allow access to any user's data * Using unencrypted credentials that can be captured and reused * Storing credentials in cookies or parameters * Using unproven authentication methods or authentication from the wrong trust domain. * Not permitting client software to authenticate the host	Use stringent authentication and protection for credential information using: * Operating system(OS) supplied frameworks. * Encrypted tokens such as session cookies * Digital signatures.
Tampering:		
changing or deleting a resource without authorization (e.g.. defacing a web site. Altering data in transit)	* Trusted data sources without validation * Sanitizing input to prevent execution o unwanted code * Running with escalated privileges * Leaving sensitive data unencrypted	Use OS security to lock down files, directories and other resources Validate your data Hash and sign data in transit(by using SSL or IPsec, for example)
Repudiation:		
Attempting to destroy.Hide or alter evidence that an action occurred (e.g. deleting logs, impersonating a user to request changes)	* Using a week or missing authorization and authentication process * Logging improperly * Allowing sensitive information on unsecured communication channels	* Use stringent authenti-cation, transaction logs and digital signatures. * Use Audit.

Description	Common causes	Preventive measures
Information Disclosure:		
Revealing personally identifiable information(PII) such as passwords and credit card data, plus information about and / or its host machines.	* Allowing an authenticated user access to other user's data. * Allowing sensitive information on unsecured communication channels * Selecting poor encryption algorithms and keys	* Store PII on a session (transitory) rather than permanent basis. * Use hashing and encryption for sensitive date whenever possible. * Match user data to user authentication
Denial of Service(DoS):		
* Flooding Sending many messages or simultaneous requests to over whelm a server. * Lockout sending a surge of requests to force a slow server response by consuming resources or causing the application to restart.	* Placing too many applications on a single server or placing conflicting applications on the same server * Neglecting to conduct comprehensive unit testing.	* Filter packets using a firewall * Use a load balancer to control the number of requests from a single source. * Use a synchronous protocols to handle processing intensive requests and errors recovery
Elevation of Privilege:		
Exceeding normal access privileges to gai administrative rights or access to confidential files	* Running web server processes as "root" or "administrator". * Using coding errors to allow buffer overflows and elevate application into a debug state.	* Use fewest-privileges context whenever possible. * Use type-safe languages and compiler options to prevent or control buffer overflows.

Discover and create baselines: Conduct a complete inventory of applications and systems, including technical information (e.g., Internet Protocol [IP], Domain Name System [DNS], OS used), plus business information (e.g., Who authorized the deployment? Who should be notified if die application fans?). Next, scan your Web infrastructure for common vulnerabilities and exploits. Check list serves and bug tracking sites for any known attacks on your OS, Web server and other third-party products. Prior to loading your application on a server, ensure that the server has been patched, hardened and scanned. Then, scan your application for vulnerabilities to known attacks, looking at HTTP requests and other opportunities for data manipulation. And, finally, test application authentication and user-rights management features and terminate unknown services.

* Assess and assign risks
* Shield your application and control damage
* Continuously monitor and review

3.2 Understanding the Web application lifecycle

The Rational Unified Process, or RUP, solution delivers a widely used iterative process framework for developing Web applications based on industry best practices. The core phases of the framework (which may require two or more iterations to complete) are: Inception, Elaboration, Construction and Transition.

Each of the four phases of the Rational Unified Process inception, elaboration, construction and transition spans multiple disciplines and may require multiple iterations. Fixing a design error after a Web application has been deployed costs approximately 30 times more than addressing it during design. To help prevent expensive fixes, enterprises can build application security testing approaches into their development and delivery process.

3.3 Considering the right testing approaches

To help prevent expensive fixes, organizations need to build application security testing approaches, such as Black Box, White Box and Gray Box testing, and also automated testing into their development and process alongside other quality management measures

4. Four strategic best practices for protecting Web applications:

To address security-related issues as they pertain to Web applications, organizations can employ four broad, strategic best practices.

1. Increase security awareness by training, communication and monitoring activities, preferably in cooperation with a consultant.
2. Categorize application risk and liability
3. Set a zero-tolerance enforcement policy
4. Integrate security testing throughout the development and delivery process

In addition to making security an integral part of the application development and delivery process, you can integrate security tests right into the application you are building to conduct event-driven testing. In this case, where a user makes a request and the application responds, the test compares the response to an expected or previously stored response to determine whether the system is operating properly or not. How you implement the code to review requests depends on the application architecture. For example, your spy component might be a mock data access object, a proxy or a class that inherits from the front-end service. You can also create code specifically for a test that you insert into the data stream to supply reporting data needed by the testing framework. Coordinating the testing objects gives you comprehensive, fine-grained control of a range of tests. You can perform these tests using either black-box or white-box testing, improving your chances of catching security problems early in the lifecycle before they pose a serious business risk.

5. CONCLUSION:

In this paper, we discuss about what you can do to protect your organizations sensitive data, by means of taking preventive measures and also we discuss an approach for improving your organization's Web application security by means of not only to perform applications testing but also to do the system security testing, for all the web based applications. Despite these satisfying results, a lot remains to be done.

6. REFERENCES:

[1] Specifcation-Based Unit Testing of Publish/Subscribe Applications. Anton Michlmayr,Pascal Fenkam, Schahram Dustdar 26th IEEE International Conference on Distributed Computing Systems Workshops(06)

[2] L. Baresi, C. Ghezzi, & L.Zanolin. Modeling & validation of publish/subscribe architectures. In S. Beydeda & V. Gruhn, editors, Testing Commercial-off-the-shelf Components and Systems, pp 273-292. Springer Verlag, 2005.

[3] C. Boyapati, S. Khurshid, and D. Marinov. Korat: Automatedtesting based on java predicates. In Proceedings of the 2002 International Symposium on Software Testing and Analysis (ISSTA), Italy, July 2002.

[4] L. Burdy, Y. Cheon, D. Cok, M. Ernst, J. Kiniry, G. T. Leavens,K. R. M. Leino, and E. Poll. An overview of jml tools and applications. International Journal on Software Tools for Technology Transfer (STTT), 7(3):212232, 2005.

[5] Y. Cheon and G. T. Leavens. The jml and junit way of unit testing and its implementation. Technical Report TR #04-02a, Department of Computer Science, Iowa State University, 2004.

[6] J. Dingel and H. Liang. Automating comprehensive safety analysis of concurrent programs using verisoft and txl. In Proceedings of the International Symposium on Foundations of Software Engineering (ACM SIGSOFT 2004/FSE-12), Nov. 2004.

[7] P. Fenkam, H. Gall, and M. Jazayeri. A systematic approach to the development of event-based applications. In roceedings of the 22nd IEEE Symposium on Reliable DistributedSystems (SRDS 2003), Florence, Italy. IEEE ComputerPress, October 2003.

[8] L. Fiege, G.M¨uhl, and F. C. G¨artner. A modular approach to build structured event-based systems. In SAC '02: Proceedings of the 2002 ACM symposium on Applied computing, pages 385392, New York, NY, USA, 2002. ACM Press.