Attacks on Web Servers and Prevention Methodologies

Today, most of on-line services are implemented as web applications. On-line banking, web search engines, email applications, social networks are just few examples of such web services. Web content is generated in real time by a software application running at server-side, i.e. the so-called web application. For example, a request on the following link
http://www.google.com/search?q=wapa&hl=en generates a HTML page containing search results for wapa in en language.

According to this scheme, the operations performed by web applications depend on their inputs (e.g. the strings wapa and en). This scheme is indeed flexible and has been proven very successful to support the information exchange on the Internet. On the other hand, unexpected -well-crafted- inputs may be submitted by cyber-criminals to remotely gain confidential data or perform unauthorized operations. Since web services are becoming even more complex and ubiquitous, it is very difficult to encounter all possible exceptions to the expected behavior of web servers and web applications. This explains why server-side web security is currently one of the key problems of the Internet [SANS (2009)].

In order to gain access to the many and various services on the Internet, a web application server is required. Web application servers are designed specifically to extend web servers to support dynamic content. The application server software “hooks in” to the web server software and automatically intercepts any user requests for dynamic content. The web server still sends out static web pages and graphic files–just like before. But now, the application server can create dynamic content by mixing data with templates, running programs, or by accessing databases.
If one is not rigorous in securely configuring and operating a public Web site, you leave yourself and your organization vulnerable to a variety of security problems. You could find yourself in an embarrassing situation because malicious intruders have changed the content of your Web pages. Compromised Web sites have served as the entry point for intrusions into an organization’s internal networks for the purpose of accessing confidential information. Your organization can face business losses or legal action if an intruder successfully violates the confidentiality of customer data. Denial-of-service attacks can make it difficult, if not impossible, for users to access your Web site. This is especially critical if you are using your site to conduct business.

As a general principle, services that are not required for a web server to function properly should not be run. This will vary depending upon the operating system and Web platform you're using, but you should always strive to have the minimal set of services running. The presence of extraneous services offers malicious individuals additional attack vectors to exploit, compromising the security of your system while providing no business benefit to you.

Along these same lines, critical components of an organization’s infrastructure should be separated so that each server only hosts one critical service. For example, if you have a dynamic Web application, it is good practice to host the database and Web server on different systems. This provides a degree of isolation and allows you to more easily implement layered protection against attacks.

Enterprise organizations have been under security attacks for the past decade, but the security events in 2011 have created a ripple effect that will be felt for years to come and will actually start to shift the way enterprise organizations view security. For example, 2011 saw a significant increase in activity from “hacktivist” groups Anonymous and Lulz Security.
(LulzSec). The motivation for these groups’ organized, systematic attacks on businesses or individuals—retaliation for perceived wrongdoing—brings new visibility to a security threat that has been looming for years and highlights a new era of security risk that must be addressed. In addition, highly publicized attacks on major corporations such as Sony, RSA, and the United States Postal Service demonstrated the significant financial loss that can result from a vulnerable system.

In the 2011 top cyber security risks report, HP Enterprise Security provides a broad view of the vulnerability threat landscape, as well as in-depth research and analysis on security attacks and trends presented four key findings:

1. Continued decline of new, disclosed vulnerabilities in commercial applications.
2. Changes in attack motivation are increasing security risk.
3. Increase in the number of attacks against a “smaller” set of known vulnerabilities.
4. Improved techniques for executing security attacks.

Due to the widespread use of web applications in the business sector, securing web applications is one of the most important things that need attention as an end-user or as a business user. A user with malicious intent has numerous methods at his disposal to compromise Web servers. Web applications require a number of interlaced applications to function properly. This means the Web server's administrator must monitor databases, extended markup languages, script interpreters and much more. Every website running on a Web server has the potential, through coding, to compromise the server. The attacks are broad and numerous, but they all fall into similar categories.

Henry Osborne
Though there are many web attack vectors in existence, SQL Injection, Cross Site Scripting, Remote Code Execution, Session Hijacking, and Username Enumeration will be the attacks under consideration as they are the more common methods.

**SQL Injection**

SQL injection is a very old approach but it's still popular among attackers. This technique allows an attacker to retrieve crucial information from a Web server's database. Depending on the application's security measures, the impact of this attack can vary from basic information disclosure to remote code execution and total system compromise.

SQL injection attacks apply to any database, but from an attacker's perspective there are a few "favorites." MS SQL has the feature of an extended stored procedure call, which allows any system level command to be executed via the MS SQL server – such as adding a user. Also, the error messages displayed by the MS SQL server reveal more information than a comparable MySQL server. While MS SQL server is not especially prone to SQL injection attacks, there are security measures which should be implemented to make it secure and not allow the SQL server to give out critical system information.

The error messages that typically get displayed help an attacker to get a hold of the information which they are looking for (such as the database name, table name, usernames, password hashes etc.). Thus displaying customized error messages may be a good workaround for this problem, however, there is another attack technique known as Blind SQL Injection where the attacker is still able to perform a SQL injection even when the application does not reveal any database server error message containing useful information for the attacker.

Henry Osborne
Some countermeasures to prevent SQL Injections include:

1. Avoid connecting to the database as a superuser or as the database owner. Always use customized database users with the bare minimum required privileges required to perform the assigned task.

2. If the PHP magic_quotes_gpc function is on, then all the POST, GET, COOKIE data is escaped automatically.

3. PHP has two functions for MySQL that sanitize user input: addslashes (an older approach) and mysql_real_escape_string (the recommended method). This function comes from PHP >= 4.3.0, so you should check first if this function exists and that you're running the latest version of PHP 4 or 5. MySQL_real_escape_string prepends backslashes to the following characters: \x00, \n, \r, \, ", and \x1a.

**Cross Site Scripting**

Cross-Site Scripting attacks are a type of injection problem, in which malicious scripts are injected into the otherwise benign and trusted web sites. Cross-site scripting (XSS) attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user in the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user’s browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access

Henry Osborne
any cookies, session tokens, or other sensitive information retained by your browser and used with that site. These scripts can even rewrite the content of the HTML page.

Cross Site Scripting is generally made possible where the user's input is displayed. The following are the popular targets:

1. On a search engine that returns 'n' matches found for your '$_search' keyword.
2. Within discussion forums that allow script tags, which can lead to a permanent XSS bug.
3. On login pages that return an error message for an incorrect login along with the login entered.

Additionally, allowing an attacker to execute arbitrary JavaScript on the victim's browser can also allow an attacker to steal victim's cookie and then hijack his session. The following are some rules that are intended to prevent all XSS in an application. While these rules do not allow absolute freedom in putting untrusted data into an HTML document, they should cover the vast majority of common use cases.

1. Never Insert Untrusted Data Except in Allowed Locations
2. HTML Escape Before Inserting Untrusted Data into HTML Element Content
3. Attribute Escape Before Inserting Untrusted Data into HTML Common Attributes
4. JavaScript Escape Before Inserting Untrusted Data into JavaScript Data Values
5. CSS Escape And Strictly Validate Before Inserting Untrusted Data into HTML Style Property Values

**Remote Code Execution**
The Remote Code Execution vulnerability allows an attacker to run arbitrary, system level code on the vulnerable server and retrieve any desired information contained therein. Improper coding errors lead to this vulnerability. At times, it is difficult to discover this vulnerability during penetration testing assignments but such problems are often revealed while doing a source code review. However, when testing Web applications it is important to remember that exploitation of this vulnerability can lead to total system compromise with the same rights as the Web server itself.

Two such types of critical remote code vulnerabilities are “Exploiting register_globals” and “XML Remote Procedure Calls (XMLRPC)”. Register_globals is a PHP setting that controls the availability of "superglobal" variables in a PHP script (such as data posted from a user's form, URL-encoded data, or data from cookies). In earlier releases of PHP, register_globals was set to "on" by default, which made a developer's life easier - but this lead to less secure coding and was widely exploited. When register_globals is set to "on" in php.ini, it can allow a user to initialize several previously uninitialized variables remotely. Many a times an uninitialized parameter is used to include unwanted files from an attacker, and this could lead to the execution of arbitrary files from local/remote locations.

XML-RPC is a specification (http://www.xmlrpc.com/spec) and a set of implementations that allow software running on disparate operating systems and in different environments to make procedure calls over the Internet. It is commonly used in large enterprises and Web environments. XML-RPC uses HTTP for its transport protocol and XML for data encoding. Several independent implementations of XML-RPC exist for PHP applications. A common flaw is in the way that several XML-RPC PHP implementations pass un-sanitized user input to the eval() function within the XML-RPC server. It results in a vulnerability that could allow a remote
attacker to execute code on a vulnerable system. An attacker with the ability to upload a crafted XML file could insert PHP code that would then be executed by the Web application that is using the vulnerable XML-RPC code.

Remote Code Execution can be prevented by properly initializing the `register_global` variable to “on” in either the php.ini or .htaccess file. Administrators who are unsure should question application developers who insist on using register_globals. Additionally, it is an absolute must to sanitize all user input before processing. As far as possible, avoid using shell commands however, if they are required, ensure that only filtered data is used to construct the string to be executed and make sure to escape the output.

**Session Hijacking**

The Session Hijacking attack consists of the exploitation of the web session control mechanism, which is normally managed for a session token. Because http communication uses many different TCP connections, the web server needs a method to recognize every user’s connections. The most useful method depends on a token that the Web Server sends to the client browser after a successful client authentication. A session token is normally composed of a string of variable width and it could be used in different ways, like in the URL, in the header of the http requisition as a cookie, in other parts of the header of the http request, or yet in the body of the http requisition.

The Session Hijacking attack compromises the session token by stealing or predicting a valid session token to gain unauthorized access to the Web Server. The session token could be

Henry Osborne
compromised in different ways; the most common are predictable session token, session snifff, client-side attacks, man-in-the-middle attack, and man-in-the-browser attack.

There are four main methods of perpetrating a session hijacking attack. These are session fixation, session side-jacking, stealing the session key, and cross-site scripting. Preventative methods include encryption of the data traffic being passed between parties, the use of a long random string as the session key, the regeneration of the session key after a successful login, and users can simply log out of websites whenever they are finished using them.

**Username Enumeration**

Username enumeration is a type of attack where the backend validation script tells the attacker if the supplied username is correct or not. Exploiting this vulnerability helps the attacker to experiment with different usernames and determine valid ones with the help of these different error messages.

Username enumeration can help an attacker who attempts to use some trivial usernames with easily guessable passwords, such as test/test, admin/admin, guest/guest, and so on. These accounts are often created by developers for testing purposes, and many times the accounts are never disabled or the developer forgets to change the password.

To prevent this type of attack consistent error messages should be displayed to prevent disclosure of valid usernames. One should also ensure that if trivial accounts have been created for testing purposes that their passwords are either not trivial or these accounts are absolutely removed after testing is over - and before the application is put online.

Henry Osborne
There are many ways to protect a web application, such as implementing secure coding practices, managing secure configuration, performing vulnerability assessment and deploying a web application firewall, but there is no silver bullet that it will protect the application entirely. A partially configured and/or patched server should not be exposed to external networks (e.g., the Internet) or external users. In addition, internal network access should be as limited as possible until all software is installed, patched, and configured securely. Insecure Web servers can be compromised in a matter of minutes after being placed on the Internet.

While it is ideal to fully harden the platform before placing it on the network, it is not always feasible. For example, some application development tool combinations cannot be installed, configured, and tested on top of a pre-hardened OS and Web server configuration. In such situations, stepwise or incremental hardening is a viable option to consider, with full validation of complete hardening occurring at production deployment.

Using a web application firewall is a relatively new method compared with other technologies, but it can become a powerful solution when you configure and use it properly. A web application firewall (WAF) is a type of firewall that filters HTTP traffic based on a rule set. It inspects the application layer so it usually comes as an appliance type or as a server module. It generally identifies and blocks common web attacks such as cross-site scripting (XSS) and SQL Injection by customizing the rules. Therefore, its rule customization is very significant and requires high maintenance (Owasp, 2011).

Organizations must anticipate the occurrence of attacks and be prepared to deal with those directed against their public Web site. If these attacks are successful and one is not prepared, the organization may suffer the consequences of a defaced, inaccurate, or unavailable Web site. These could include being unable to conduct business and result in a loss of customer confidence.
confidence and reputation. Web applications have been evolving so fast and they have become one of the most important things that we can’t live without, such as electricity and water. The use of web applications will not stop increasing but, on the other hand, attackers will not stop trying to penetrate your applications.
References


Henry Osborne