MODULE 5 WEB SERVER SECURITY

Introduction

Internet facing web servers are exposed to high security risks. We quite commonly see web servers being hacked (eg: malicious code being injected in website content), and then clients that are browsing the website are most likely to be transparently compromised (aka drive-by download). And there is also the denial of service risk, the information leakage risk, etc.

Security to the web server needs to be divided in various sections, these sections are having different challenges which includes the zero day attacks, patching issues, infrastructure related problems.

Securing the Operating System

Web servers operate on a general-purpose operating system. Many security issues can be avoided if the operating systems underlying Web servers are configured appropriately. Default hardware and software configurations are typically set by vendors to emphasize features, functions, and ease of use at the expense of security.

Securely installing and configuring an Operating system

1. Patch and Upgrade Operating System
   All operating systems released today have some known vulnerabilities that should be corrected before using the operating system to host a Web server. The patching should be done through a process as follows:
   - Create and implement a patching process
   - Identify latest patches for the vulnerabilities in the wild
   - Mitigate vulnerabilities through some work around solution until patches are available
   - Install permanent fixes.

2. Remove or Disable Unnecessary Services and Applications
   A Web server should be on a dedicated, single-purpose host. Many operating systems are configured by default to provide a wider range of services and applications than required by a Web server; therefore, a Web administrator should configure the operating system to remove or disable unneeded services. Some common examples of services that should usually be disabled would include:
   - Windows Network Basic Input/Output System (NetBIOS), if not required
   - NFS, if not required „ File Transfer Protocol (FTP)
   - Berkeley “r” services (e.g., rlogin, rsh, rcp)
   - Telnet

- Network Information System (NIS)
- Simple Mail Transfer Protocol (SMTP)
- Compilers
- Software development tools

Removing unnecessary services and applications is preferable to simply disabling them through configuration settings, because attacks that attempt to alter settings and activate a disabled service cannot succeed when the functional components are completely removed.

When configuring the operating system, apply the principle "disable everything except that which is expressly permitted" – that is, disable or, preferably, remove all services and applications and then selectively enable those required by the Web server. If possible, install the minimal operating system configuration that is required for the Web server application. If the operating system installation system provides a "minimal installation" option, choose that because it will minimize the effort required to remove unnecessary services. Many uninstall scripts or programs do not completely remove all components of service; therefore, it is always better to avoid installing unnecessary services when possible.

The services enabled on a Web server will depend on the functions the organization wants the server to provide. Those services might include database protocols to access a database, file transfer protocols, and remote administration services. Each of these services, even though they may be required, comes with an increased risk to the server. Whether the risks outweigh the benefits is a decision each organization must make for itself.

3. Configuring Operating System User Authentication

   For Web servers, authorized users who can configure the system and initiate Web services are typically a small number of designated Web administrators and Webmasters. However, the users who can access the public Web server may range from unrestricted to restricted subsets of the Internet community. To enforce policy restrictions, if required, the Web administrator must configure the system to authenticate prospective users by requiring proof that each person is authorized for such access. Even though a Web server may allow unauthenticated access to most Web services, administrative and other types of specialized access should be limited to specific individuals and groups.

   To ensure the appropriate user authentication is in place, take the following steps:

   a) Remove or disable unneeded default accounts and groups

      The default configuration of the operating system often includes guest accounts (with and without passwords), administrator or root level accounts, and accounts associated with local and network services. The names and passwords for those accounts are well known. Remove or disable unnecessary accounts to eliminate their use by intruders, including guest accounts on computers containing sensitive information. If there is no requirement to retain a guest account or group, severely

restrict its access and change the password in accordance with the organizational password policy.

b) Disable non-interactive accounts
   Disable accounts (and the associated passwords) that need to exist but do not require an interactive login. For Unix systems, disable the login shell, or provide a login shell with NULL functionality (/bin/false).
c) Create the user groups.
   Assign users to the appropriate groups. Then assign rights to the groups. This approach is preferable to assigning rights to individual users.
d) Create the user accounts
   Identify who will be authorized to use each computer and its services. Create only the necessary accounts. Discourage or prohibit the use of shared accounts.
e) Check the organization's password policy
   Set account passwords appropriately which should be based on Length, complexity, Aging, Reuse, Authority.
f) Configure computers to deny login after a small number of failed attempts
   It is relatively easy for an unauthorized user to try to gain access to a computer by using automated software tools that attempt all passwords. If the operating system provides the capability, configure it to deny login after three failed attempts. Typically, the account is "locked out" for a period of time (such as 30 minutes) or until a user with appropriate authority reactivates it.
g) Install and configure other security mechanisms to strengthen authentication
   If the information on the Web server requires it, consider using other authentication mechanisms such as tokens, client/server certificates, or one-time password systems. Although they can be more expensive and difficult to implement, they may be justified in some circumstances. When such authentication mechanisms and devices are used, the organization's policy should be reviewed to reflect in the way in which they are applied.
h) Security testing the OS
   It's necessary to validate all the controls through periodic assessment like penetration testing and vulnerability assessment.

Securely installing and configuring the web server

1. Securely Installing the Web Server
   The minimal amount of Web server services required and eliminate any known vulnerabilities through patches or upgrades. If the installation program installs any unnecessary applications, services, or scripts, they should be removed immediately once the installation process completes. During the installation of the Web server, the following steps should be performed:
   a) Install the server software on a dedicated host

b) Install the minimal Internet services required
c) Apply any patches or upgrades to correct for known vulnerabilities
d) Create a dedicated physical disk or logical partition (separate from operating system and server application) for Web content
e) Remove or disable all services installed by the Web server application but not required (e.g., gopher, FTP, and remote administration)
f) From the Web server application root directory, remove all files that are not part of the Web site
g) Remove all sample documents, scripts, and executable code
h) Remove all vendor documentation from server
i) Apply appropriate security template or hardening script to server Reconfigure HTTP service banner (and others as required) NOT to report Web server and operating system type and version. (This can be accomplished in IIS using the Microsoft's free IIS Lockdown Tool and in Apache via the "ServerTokens" directive.)

2. Configuring Access Controls

Most Web server host operating systems provide a capability to specify access privileges individually for files, devices, and other computational resources on that host. Any information that the Web server can access using these controls can potentially be distributed to all users accessing the public Web site. The Web server software is likely to provide additional file, device, and resource access controls specific to its operation.

Web administrators should consider from two perspectives how best to configure these access controls to protect information stored on their public Web server:

a) Limit the access of the Web server software to a subset of computational resources
b) Limit the access of users through additional access controls enforced by the Web server, where more detailed levels of access control are required

Typical files to which access should be controlled are as follows:

- Application software and configuration files
- Files related directly to security mechanisms:

- Password hash files and other files used in authentication
- Files containing authorization information used in controlling access
- Cryptographic key material used in confidentiality, integrity, and non-repudiation services.

- Server log and system audit files
- System software and configuration files.

3. Using File Integrity Checkers

A file integrity checker is an application that computes and stores a checksum for every guarded file and establishes a database of file checksums. It allows a system administrator to easily recognize changes to critical files, particularly unauthorized changes. Checksums should be recomputed regularly to test the current value against the stored value to identify any file modifications.

Although an integrity checker is a useful tool that does not require a high degree of human interaction, it needs to be used carefully to ensure that it is effective. To create the first reference database a file integrity checker requires a system that is known to be in a secure state.

Integrity checkers should be run nightly on a selection of system files that would be affected by a compromise. Integrity checkers should also be used when a compromise is suspected for determining the extent of possible damage. If an integrity checker detects unauthorized system file modifications, the possibility of a security incident should be considered and investigated according to the organization's incident response and reporting policy and procedures.

Securing Web Content

The two main components to Web security are the security of the underlying server application and operating systems, and the security of the actual content.

1. Publishing Information on Public Web Sites

   Many organizations do not have a Web publishing process or policy that determines what type of information to publish openly, what information to publish with restricted access, and what information should be omitted from any publicly accessible repository.

   To ensure a consistent approach, an organization should create a formal policy and process for determining and approving the information to be published on a Web server. In many organizations, this is the responsibility of the CIO and/or public affairs officer. Such a process should include the following steps:

   - Identify information that should be published on the Web
   - Identify the target audience (Why publish if no audience exists?)
   - Identify possible negative ramifications of publishing the information
   - Identify who should be responsible for creating, publishing, and maintaining this particular information
   - Create or format information for Web publishing
   - Review the information for sensitivity and distribution/release controls (including the sensitivity of the information in aggregate)
   - Determine the appropriate access and security controls
   - Publish information
   - Verify published information

- Periodically review published information to confirm continued compliance with organizational guidelines.

2. Securing Active Content and Content Generation Technologies

   Active content refers to interactive program elements downloaded to the client (i.e., a Web browser) and processed there instead of the server. A variety of active content technologies exists; some of the more popular examples are ActiveX, Java, VBScript, JavaScript, and Asynchronous JavaScript and XML (AJAX). The use of active content often requires users to reduce the security settings on their Web browsers for processing to occur. If not implemented correctly, active content can present a serious threat to the end user.

   For example, active content can take actions independently without the knowledge or expressed consent of the user. While active content poses risk to the client, it can also pose risk to the Web server. The reason is that information processed on the client is under the control of the user, who can potentially manipulate the results by reverse engineering and tampering with the active content. For example, form validation processing done with active content elements on the client side can be changed to return out-of-range options or other unexpected results to the server. Therefore, the results of processing done on the client by elements of active content should not be trusted by the server; instead, the results should be verified by the server. Organizations considering the deployment of client-side active content should carefully consider the risks to both their users and their Web servers.

3. Server-Side Content Generator Security Considerations

   When examining or writing an active content executable or script, consider the following:

   - The executable code should be as simple as possible. The longer or more complex it is, the more likely it will have problems.
   - The executable code's ability to read and write programs should be limited. Code that reads files may inadvertently violate access restrictions or pass sensitive system information. Code that writes files may modify or damage documents or introduce Trojan horses.
   - The code's interaction with other programs or applications should be analyzed to identify security vulnerabilities. For example, many CGI scripts send e-mails in response to form input by opening up a connection with the sendmail program. Ensure this interaction is performed in a secure manner.
   - On Linux/Unix hosts, the code should not run with suid (set-user-id).
   - The code should use explicit path names when invoking external programs. Relying on the PATH environment variable to resolve partial path names is not recommended.
   - Web servers should be scanned periodically for vulnerabilities, even if they do not employ active content. Network security scanners may detect vulnerabilities in the Web server, OS, or other services running on the Web server. Web application vulnerability scanners specifically scan for content generator vulnerabilities.

- Web content generation code should be scanned and/or audited (depending on the sensitivity of the Web server and its content). Commercially available tools can scan .NET or Java code. A number of commercial entities offer code review services. Web content generation code should be developed following current recommended practices.
- For data entry forms, determine a list of expected characters and filter out unexpected characters from input data entered by a user before processing a form. For example, on most forms, expected data falls in these categories: letters a-z, A-Z, and 0-9. Care should be taken when accepting special characters such as &, ′, ″, @, and !. These symbols may have special meanings within the content generation language or other components of the Web application.
- Character set encoding should be explicitly set in each page. Then the user data should be scanned for byte sequences that represent special characters for the given encoding scheme.
- Each character in a specified character set can be encoded using its numeric value. Encoding the output can be used as an alternate for filtering the data. Encoding becomes especially important when special characters, such as copyright symbols, can be part of the dynamic data. However, encoding data can be resource intensive, and a balance must be struck between encoding and other methods for filtering the data.
- Cookies should be examined for any special characters. Any special characters should be filtered out.
- An encryption mechanism should be used to encrypt passwords entered through script forms
- For Web applications that are restricted by username and password, none of the Web pages in the application should be accessible without executing the appropriate login process.
- Many Web servers and some other Web server software install sample scripts or executables during the installation process. Many of these have known vulnerabilities and should be removed immediately. See appropriate manufacturer's documentation or Web sites for more information.

4. Cross-Site Scripting Vulnerabilities

   Cross-site scripting (XSS) is a vulnerability typically found in interactive Web applications that allows code injection by malicious Web users into the Web pages viewed by other users. It generally occurs in Web pages that do not do the appropriate bounds checking on data input by users. An exploited cross-site scripting vulnerability can be used by attackers to compromise other users' computers or to receive data from another user's Web session (e.g., user ID and password or session cookie). Thus,

although this is a client side exploit, it also impacts the server indirectly since a compromised user, particularly one with elevated privileges, represents a threat to the server. XSS vulnerabilities are used frequently to conduct phishing attacks or exploit Web browser vulnerabilities to gain control of end user PCs

The solution to XSS attacks is to validate all user input and remove any unexpected or potentially risky data. Another solution is to use an HTML-quoted of any user input that is presented back to other users. This will prevent the Web browsers of other users from interpreting that input and acting on any embedded commands present.

Authentication and Encryption Technologies

Public Web servers often support a range of technologies for identifying and authenticating users with differing privileges for accessing information. Some of these technologies are based on cryptographic functions that can provide an encrypted channel between a Web browser client and a Web server that supports encryption. Without user authentication, organizations will not be able to restrict access to specific information to authorized users. All information that resides on a public Web server will then be accessible by anyone with access to the server. In addition, without some process to authenticate the server, users will not be able to determine if the server is the "authentic" Web server or a counterfeit version operated by a malicious entity.

1. Address-Based Authentication

   The simplest authentication mechanism that is supported by most Web servers is address-based authentication. Access control is based on the IP address and/or hostname of the host requesting information.
2. Basic Authentication

   The basic authentication technology uses the Web server content's directory structure. Typically, all files in the same directory are configured with the same access privileges. A requesting user provides a recognized user identification and password for access to files in a given directory. More restrictive access control can be enforced at the level of a single file within a directory if the Web server software provides this capability. Each vendor's Web server software has its own method and syntax for defining and using this basic authentication mechanism.

   From a security perspective, the main drawback of this technology is that all password information is transferred in an encoded, rather than an encrypted, form. Anyone who knows the standardized encoding scheme can decode the password after capturing it with a network sniffer. Furthermore, any Web content is transmitted as unencrypted plaintext, so this content also can be captured, violating confidentiality.
3. Digest Authentication

   Because of the drawbacks with basic authentication, an improved technique known as digest authentication was introduced in version 1.1 of the HTTP protocol. Digest authentication uses a challenge-response mechanism for user authentication. Under this approach, a nonce or arbitrary value is sent to the user, who is prompted for an ID and password, as with basic

authentication. However, in this case, the information entered by the user is concatenated and a cryptographic hash of the result is formed. This hash is concatenated with the nonce and a hash of the requested method and URL, and the result is then rehashed as a response value that is sent to the server.

Because the user's password is not sent in the clear, it cannot be directly sniffed from the network. The user's password is not needed by the server to authenticate the user—only the hashed value of the user ID and password. Because the nonce can serve as an indicator of timeliness (e.g., it can be composed of date and time information), replay attacks are also thwarted. Digest authentication is also susceptible to offline dictionary attacks, where the attacker tries various passwords in an attempt to recreate the captured digest value. These limitations can be overcome using digest authentication in conjunction with SSL/TLS.

4. SSL/TLS

   The SSL and TLS protocols provide server and client authentication and encryption of communications. TCP/IP governs the transport and routing of data over the Internet. Other protocols, such as HTTP, LDAP, and Internet Message Access Protocol (IMAP), run "on top of" TCP/IP in that they all use TCP/IP to support typical application tasks, such as displaying Web pages or delivering e-mail messages. Thus, SSL/TLS can support more than just secure Web communications. Figure 7-1 shows how SSL/TLS fits between the application and network/transport layers of the Internet protocol suite.

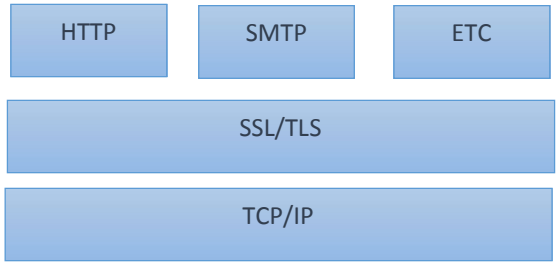

Fig: SSL/TLS Location within the Internet