SECURITY IN WEB APPLICATION

Unit 6

WEB APPLICATION SECURITY FUNDAMENTALS

Security fundamentals relies on following elements:

Authentication: Authentication addresses the question: who are you? It is the process of uniquely identifying the clients of your applications and services. These might be end users, Other services, processes, or computers. In security , authenticated clients are referred to as principals. Logging on to a PC with a username and password is **authentication**.

Authorization: Authorization address the question: what can you do? It is the process that governs the resources and operations that the authenticated client is permitted to access resources include files, databases, tables, rows and so on. In other words, **Authorization** is the process of verifying that you have access to something. Gaining access to a resource (e.g. directory on a hard disk) because the permissions configured on it allow you access is **authorization**.

WEB APPLICATION SECURITY FUNDAMENTALS

- **Auditing :**
- Effective auditing and logging is the key to non-repudiation. Non- repudiation guarantees that a user cannot deny performing an operation or initiating a transaction. For Example, in an ecommerce system, non-repudation mechanisms are required to make sure that a consumer cannot deny ordering 100 copies if a particular book.
- **Confidentiality:**
- Confidentiality, also referred to as privacy, is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized users. Encryption is frequently used to enforce confidentiality.

WEB APPLICATION SECURITY FUNDAMENTALS

- Integrity:
- Integrity is the guarantee that data is protected from accidental or deliberate(malicious) modification. Like privacy, integrity is a key concern, particularly for data passed across networks. Integrity for data in transit is typically provided by using hashing techniques and message authentication codes.

- Availability
- The availability of data is a measure of the data's accessibility. For example, if a server were down only five minutes per year, it would have an availability of 99.999 percent (that is, "five nines" of availability).

THREATS, VULNERABILITY AND ATTACKS

- A threat is any potential occurrence, malicious that could harm an asset. In other words, a threat is any bad thing that can happen to you assets.
- A vulnerability is an inherent weakness in the design, configuration, implementation, or management of a network or system that renders it susceptible to a threat. Vulnerabilities are what make networks susceptible to information loss and downtime. Every network and system has some kind of vulnerability. Weak input validation is an example of an application layer vulnerability, which can result in input attacks.

THREATS, VULNERABILITY AND ATTACKS

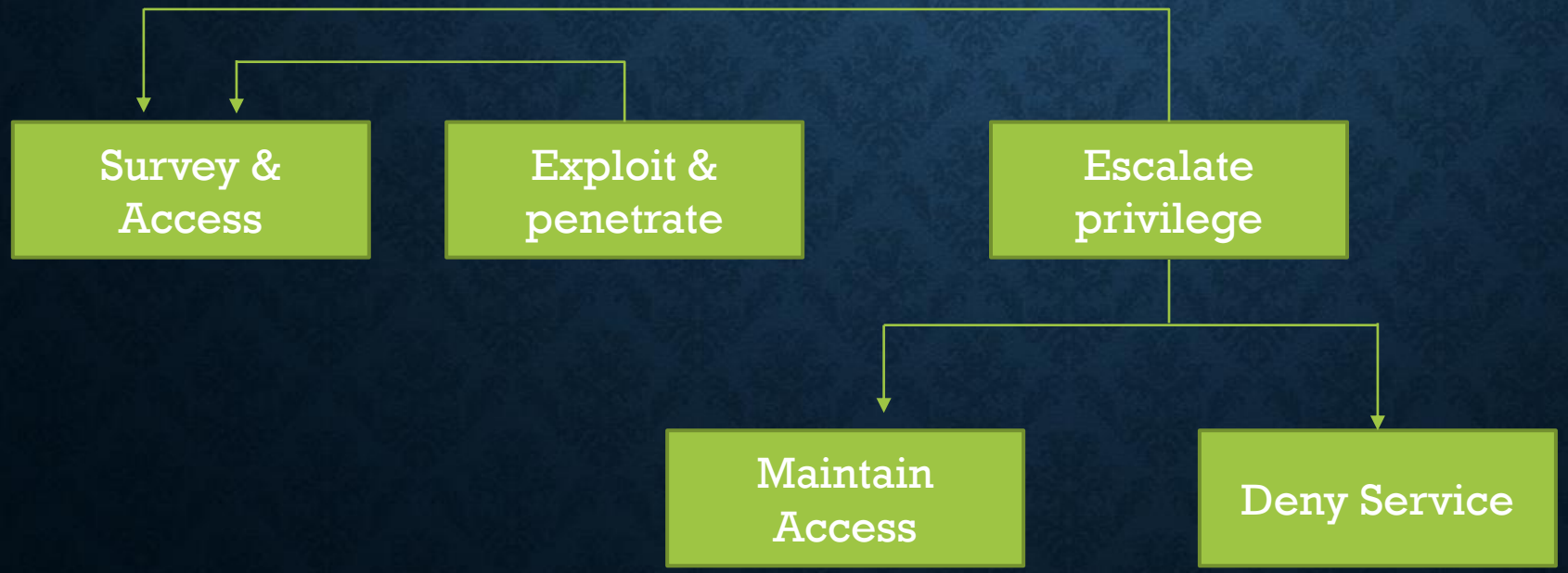
- An attack is a specific technique used to exploit a vulnerability. For example, a threat could be a denial of service. A vulnerability is in the design of the operating system, and an attack could be a "ping of death." There are two general categories of attacks, passive and active. Passive attacks are very difficult to detect, because there is no overt activity that can be monitored or detected. Examples of passive attacks would be packet sniffing or traffic analysis. These types of attacks are designed to monitor and record traffic on the network. They are usually employed for gathering information that can be used later in active attacks.
- Active attacks, as the name implies, employ more overt actions on the network or system. As a result, they can be easier to detect, but at the same time they can be much more devastating to a network. Examples of this type of attack would be a denial-of-service attack or active probing of systems and networks.
- Networks and systems face many types of threats. There are viruses, worms, Troja

SECURITY DESIGN PRINCIPLES

- 1. Secure the weakest link
- 2. Grant least privilege
- 3. Economize mechanism
- 4. Don’t trust user input easily
- 5. Assume your secrets are not safe
- 6. Promote privacy
- 7. Fail Securely

THREATS AND COUNTERMEASURES

- Assets – A resource of value such as the data in a database or on the file system, or a system resource.
- Countermeasures – A safeguard that addresses a threat and mitigates risk.
- Basis Steps in attacker methodology:
- **Anatomy of Attack**



THREATS AND COUNTERMEASURES

- **Survey and Access** : Survey and Accessing the potential target are done in tandem. The first step an attacker usually takes is to survey the potential target to identify and access its characteristics. These characteristics may include its supported services and protocols together with potential vulnerabilities and entry points. The attackers uses the information gathered in the survey and assess phase to plan an initial attack.
- For Example an attacker can detect a cross-site scripting (XSS) vulnerabilities by testing to see if any controls in a web page echo back output.
- **Exploit and Penetrate :** Having surveyed target, the next step is to exploit and penetrate. If the network and host are fully secured, your application (the front gate) becomes the next channel for attack.
- For an attacker, the easiest way to enter into an application is through the same entrance that legitimate users use.

THREATS AND COUNTERMEASURES

- Escalate Privilege : After attackers manage to comprise an application or network, perhaps by injecting code into an application or creating an authenticated session with the MS 2000 Operating System, they immediately attempt to escalate privileges. Specifically, they look for administration privileges provided by accounts that are members of the administrators group. They also seek out the high level of privileges offered by the local system account.
- Using least privileged service accounts throughout your application is a primary defense against privilege escalation attacks. Also, many networks level privilege escalation attacks require an interactive logon session.

THREATS AND COUNTERMEASURES

- **Maintain Access**: Having gained access to a system, an attacker takes steps to make future access easier and to cover his or her tracks. Common approaches for making future access easier include planting back-door program or using an existing account that lacks strong protection. Covering tracks typically involves clearing logs and hiding tools. As such, audit logs are a primary target for the attacker.
- Log files should be secured, and they should be analyzed on the regular basis. Log file analysis can often uncover the early signs of an attempted break-in before damage is done.

THREATS AND COUNTERMEASURES

- **Deny Service** : Attackers who cannot gain access often mount a denial of service attack to prevent others from using the application. For other attackers, the denial of service option is their goal from the outset. An example is the SYN flood attack, where the attacker uses a program to send a flood of TCP SYN request to fill the pending connection queue on the server. This prevents other users from establishing network connections.

THREATS AND COUNTERMEASURES

- The primary components that make up your network infrastructure are routers, firewalls and switches. They act as the gatekeepers guarding your servers and applications from attacks and intrusions. An attacker may exploit poorly configured network devices. Top network level threats include:
- 1. Information Gathering :
- 2. Sniffing
- 3. Spoofing
- 4. Session Hijacking
- 5. Denial of Service

. INFORMATION GATHERING

- Network devices can be discovered and profiled in much the same way as other types of systems. Attackers usually start with port scanning. After they identify open ports, they use banner grabbing and enumeration to detect device types and to determine operating system and application versions. Armed with this information, an attacker can attack known vulnerabilities that may not be updated with security patches.
- Countermeasures to prevent information gathering includes:
- 1 configure router to restrict their responses to foot printing requests.
- 2 configure operating systems that host network software (for example, software firewalls) to prevent foot printing by disabling unused protocols and unnecessary ports.

SNIFFING

- Sniffing is the act of monitoring traffic on the network for data such as plaintext passwords or configuration information. With a simple packet sniffer, an attacker can easily read all plaintext traffic. Also, attackers can crack packets encrypted by lightweight hashing algorithms and can decipher the payload that you considered to be safe. The sniffing of packets requires a packet sniffer in the path of the server/client communication.
- **Countermeasures to help prevent sniffing include:**
- 1. Use strong physical security and proper segmenting of the network. This is the first step in preventing traffic from being collected locally.
- 2. Encrypt communication fully, including authentication credentials. This prevents sniffed packets from being usable to an attacker. SSL and IPSec (Internet Protocol Security) are examples of encryption solution

PRESENTATION

- Spoofing+Session Hijacking+Denial of Service Attacks
- Host Threat and countermeasures
- Application Threat and Coutermeasures

SPOOFING

- Spoofing is a means to hide one's true identity on the network. To create a spoofed identity, and attacker uses a fake source address that does not represent the actual address of the packet. Spoofing may be used to hide the original source of an attack to work around network access control lists (ACLs) that are in place to limit host access based on source address rules.
- Although carefully crafted spoofed packets may never be tracked to the original sender, a combination of filtering rules prevents spoofed packets originating from your network, allowing you to block obviously spoofed packets.
- **Countermeasures to help prevent Spoofing include:**
- Filter incoming packets that appear to come from an internal IP address at your perimeter.
- Filter outgoing packets that appear to originate from an invalid local IP address.

SESSION HIJACKING

- Session hijacking is a method of taking over a Web user session by secretly obtaining the session ID by false identity and masquerading as the authorized user. Once the user's session ID has been accessed (through session prediction), the attacker can masquerade as that user and do anything the user is authorized to do on the network.
- **Countermeasures to help prevent Spoofing include:**
- **Use encrypted session negotiation.**
- **Use encrypted communication channels.**
- **Stay informed of platform patches to fix the TCP/IP vulnerabilities such as predictable packet sequence.**

DENIAL OF SERVICE ATTACKS

- A **denial-of-service** (**DoS**) **attack** is an attempt to make a machine or network resource unavailable to its intended users, such as to temporarily or indefinitely interrupt or suspend services of a host connected to the Internet. Denial of Service denies legitimate users access to a server or services. The SYN flood attack is a common example of a network level denial of service attack. It is easy to lunch and difficult to track. The aim of the attack is to send more requests to a server than it can handle. The attack exploits a potential vulnerability in the TCP/IP connection establishment mechanism and floods the server's pending connection queue.
- Countermeasure to prevent Denial of Service include:
- Apply the latest packs.
- Harden the TCP/IP stack by applying the appropriate registry setting to increase the size of the TCP connection queue, decrease the connection establish

HOST THREATS AND COUNTERMEASURES

- Host threats are directed at the system software upon which your applications are built. This includes Windows 2000, Microsoft Windows Server 2003, Internet Information Services (IIS), the .NET Framework, and SQL Server depending upon the specific server role. Top host level threats include:
- **Viruses, Trojan horses, and worms**
- **Footprinting**
- **Profiling**
- **Password cracking**
- **Denial of service**
- **Arbitrary code execution**
- **Unauthorized access**

VIRUSES, TROJAN HORSES, AND WORMS

- A virus is a program that is designed to perform malicious acts and cause disruption to your operating system or applications. A Trojan horse resembles a virus except that the malicious code is contained inside what appears to be a harmless data file or executable program. A worm is similar to a Trojan horse except that it self-replicates from one server to another. Worms are difficult to detect because they do not regularly create files that can be seen. They are often noticed only when they begin to consume system resources because the system slows down or the execution of other programs halt. The Code Red Worm is one of the most notorious to afflict IIS; it relied upon a buffer overflow vulnerability in a particular ISAPI filter.
- Although these three threats are actually attacks, together they pose a significant threat to Web applications, the hosts these applications live on, and the network used to deliver these applications. The success of these attacks on any system is possible through many vulnerabilities such as weak defaults, software bugs, user error, and inherent vulnerabilities in Internet protocols.
- **Countermeasures that you can use against viruses, Trojan horses, and worms include:**
- **Stay current with the latest operating system service packs and software patches.**
- **Block all unnecessary ports at the firewall and host.**
- **Disable unused functionality including protocols and services.**
- **Harden weak, default configuration settings**.

FOOTPRINTING

- Examples of footprinting are port scans, ping sweeps, and NetBIOS enumeration that can be used by attackers to glean valuable system-level information to help prepare for more significant attacks. The type of information potentially revealed by footprinting includes account details, operating system and other software versions, server names, and database schema details.
- **Countermeasures to help prevent footprinting include:**
- **Disable unnecessary protocols.**
- **Lock down ports with the appropriate firewall configuration.**
- **Use TCP/IP and IPSec filters for defense in depth.**
- **Configure IIS to prevent information disclosure through banner grabbing.**
- **Use an IDS that can be configured to pick up footprinting patterns and reject suspicious traffic.**

PASSWORD CRACKING

- If the attacker cannot establish an anonymous connection with the server, he or she will try to establish an authenticated connection. For this, the attacker must know a valid username and password combination. If you use default account names, you are giving the attacker a head start. Then the attacker only has to crack the account's password. The use of blank or weak passwords makes the attacker's job even easier.
- **Countermeasures to help prevent password cracking include:**
- **Use strong passwords for all account types.**
- **Apply lockout policies to end-user accounts to limit the number of retry attempts that can be used to guess the password.**
- **Do not use default account names, and rename standard accounts such as the administrator's account and the anonymous Internet user account used by many Web applications.**
- **Audit failed logins for patterns of password hacking attempts.**

DENIAL OF SERVICE

- Denial of service can be attained by many methods aimed at several targets within your infrastructure. At the host, an attacker can disrupt service by brute force against your application, or an attacker may know of a vulnerability that exists in the service your application is hosted in or in the operating system that runs your server.
- **Countermeasures to help prevent denial of service include:**
- **Configure your applications, services, and operating system with denial of service in mind.**
- **Stay current with patches and security updates.**
- **Harden the TCP/IP stack against denial of service.**
- **Make sure your account lockout policies cannot be exploited to lock out well known service accounts.**
- **Make sure your application is capable of handling high volumes of traffic and that thresholds are in place to handle abnormally high loads.**
- **Review your application's failover functionality.**
- **Use an IDS that can detect potential denial of service attacks.**

ARBITRARY CODE EXECUTION

- If an attacker can execute malicious code on your server, the attacker can either compromise server resources or mount further attacks against downstream systems. The risks posed by arbitrary code execution increase if the server process under which the attacker's code runs is over-privileged. Common vulnerabilities include weak IIS configuration and unpatched servers that allow path traversal and buffer overflow attacks, both of which can lead to arbitrary code execution.
- **Countermeasures to help prevent arbitrary code execution include:**
- **Configure IIS to reject URLs with "../" to prevent path traversal.**
- **Lock down system commands and utilities with restricted ACLs.**
- **Stay current with patches and updates to ensure that newly discovered buffer overflows are speedily patched.**

UNAUTHORIZED ACCESS

- Inadequate access controls could allow an unauthorized user to access restricted information or perform restricted operations. Common vulnerabilities include weak IIS Web access controls, including Web permissions and weak NTFS permissions.
- **Countermeasures to help prevent unauthorized access include:**
- **Configure secure Web permissions.**
- **Lock down files and folders with restricted NTFS permissions.**
- **Use .NET Framework access control mechanisms within your ASP.NET applications, including URL authorization and principal permission demands.**

APPLICATION THREATS AND COUNTERMEASURES

- A good way to analyze application-level threats is to organize them by application vulnerability category. The various categories used in the subsequent sections of this chapter and throughout the guide, together with the main threats to your application, are summarized in Table
- Threats by Application Vulnerability Category

Category	Threats
Input validation	Buffer overflow; cross-site scripting; SQL injection
Authentication	Network eavesdropping; brute force attacks; dictionary attacks; cookie replay; credential theft
Authorization	Elevation of privilege; disclosure of confidential data; data tampering(to make change); luring(attract) attacks
Configuration management	Unauthorized access to administration interfaces; unauthorized access to configuration stores; retrieval of clear text configuration data; lack of individual accountability; over-privileged process and service accounts

APPLICATION THREATS AND COUNTERMEASURES

Sensitive data	Access sensitive data in storage; network eavesdropping; data tampering
Session management	Session hijacking; session replay; man in the middle
Cryptography	Poor key generation or key management; weak or custom encryption
Parameter manipulation	Query string manipulation; form field manipulation; cookie manipulation; HTTP header manipulation
Exception management	Information disclosure; denial of service
Auditing and logging	User denies performing an operation; attacker exploits an application without trace; attacker covers his or her tracks

CONFIGURATION MANAGEMENT

- Many applications support configuration management interfaces and functionality to allow operators and administrators to change configuration parameters, update Web site content, and to perform routine maintenance. Top configuration management threats include:
- **Unauthorized access to administration interfaces**
- **Unauthorized access to configuration stores**
- **Retrieval of plaintext configuration secrets**
- **Lack of individual accountability**
- **Over-privileged process and service accounts**

UNAUTHORIZED ACCESS TO ADMINISTRATION INTERFACES

- Administration interfaces are often provided through additional Web pages or separate Web applications that allow administrators, operators, and content developers to managed site content and configuration. Administration interfaces such as these should be available only to restricted and authorized users. Malicious users able to access a configuration management function can potentially deface the Web site, access downstream systems and databases, or take the application out of action altogether by corrupting configuration data.
- **Countermeasures to prevent unauthorized access to administration interfaces include:**
- **Minimize the number of administration interfaces.**
- **Use strong authentication, for example, by using certificates.**
- **Use strong authorization with multiple gatekeepers.**
- **Consider supporting only local administration. If remote administration is absolutely essential, use encrypted channels, for example, with VPN technology or SSL, because of the sensitive nature of the data passed over administrative interfaces. To further reduce risk, also consider using IPSec policies to limit remote administration to computers on the internal network.**

UNAUTHORIZED ACCESS TO CONFIGURATION STORES

- Because of the sensitive nature of the data maintained in configuration stores, you should ensure that the stores are adequately secured.
- Countermeasures to protect configuration stores include:
- Configure restricted ACLs on text-based configuration files such as Machine.config and Web.config.
- Keep custom configuration stores outside of the Web space. This removes the potential to download Web server configurations to exploit their vulnerabilities.

SENSITIVE DATA

- Sensitive data is subject to a variety of threats. Attacks that attempt to view or modify sensitive data can target persistent data stores and networks. Top threats to sensitive data include:
- Access to sensitive data in storage
- Network eavesdropping
- Data tampering
- **Access to Sensitive Data in Storage**
- You must secure sensitive data in storage to prevent a user malicious or otherwise from gaining access to and reading the data.
- Countermeasures to protect sensitive data in storage include:
- Use restricted ACLs on the persistent data stores that contain sensitive data.
- Store encrypted data.
- Use identity and role-based authorization to ensure that only the user or users with the appropriate level of authority are allowed access to sensitive data. Use role-based security to differentiate between users who can view data and users who can modify data.

NETWORK EAVESDROPPING

- The HTTP data for Web application travels across networks in plaintext and is subject to network eavesdropping attacks, where an attacker uses network monitoring software to capture and potentially modify sensitive data.
- Countermeasures to prevent network eavesdropping and to provide privacy include:
- Encrypt the data.
- Use an encrypted communication channel, for example, SSL.

DATA TAMPERING

- Data tampering refers to the unauthorized modification of data, often as it is passed over the network.
- One countermeasure to prevent data tampering is to protect sensitive data passed across the network with tamper-resistant protocols such as hashed message authentication codes (HMACs).
- An HMAC provides message integrity in the following way:
- The sender uses a shared secret key to create a hash based on the message payload.
- The sender transmits the hash along with the message payload.
- The receiver uses the shared key to recalculate the hash based on the received message payload. The receiver then compares the new hash value with the transmitted hash value. If they are the same, the message cannot have been tampered with.

SESSION MANAGEMENT

- Session management for Web applications is an application layer responsibility. Session security is critical to the overall security of the application.
- Top session management threats include:
- **Session hijacking**
- **Session replay**
- **Man in the middle**

SESSION HIJACKING

- A session hijacking attack occurs when an attacker uses network monitoring software to capture the authentication token (often a cookie) used to represent a user's session with an application. With the captured cookie, the attacker can spoof the user's session and gain access to the application. The attacker has the same level of privileges as the legitimate user.
- Countermeasures to prevent session hijacking include:
- Use SSL to create a secure communication channel and only pass the authentication cookie over an HTTPS connection.
- Implement logout functionality to allow a user to end a session that forces authentication if another session is started.
- Make sure you limit the expiration period on the session cookie if you do not use SSL. Although this does not prevent session hijacking, it reduces the time window available to the attacker.

SESSION REPLAY

- Session replay occurs when a user's session token is intercepted and submitted by an attacker to bypass the authentication mechanism. For example, if the session token is in plaintext in a cookie or URL, an attacker can sniff it. The attacker then posts a request using the hijacked session token.
- Countermeasures to help address the threat of session replay include:
- Re-authenticate when performing critical functions. For example, prior to performing a monetary transfer in a banking application, make the user supply the account password again.
- Expire sessions appropriately, including all cookies and session tokens.
- Create a "do not remember me" option to allow no session data to be stored on the client.

MAN IN THE MIDDLE ATTACKS

- A man in the middle attack occurs when the attacker intercepts messages sent between you and your intended recipient. The attacker then changes your message and sends it to the original recipient. The recipient receives the message, sees that it came from you, and acts on it. When the recipient sends a message back to you, the attacker intercepts it, alters it, and returns it to you. You and your recipient never know that you have been attacked.
- Any network request involving client-server communication, including Web requests, Distributed Component Object Model (DCOM) requests, and calls to remote components and Web services, are subject to man in the middle attacks.
- Countermeasures to prevent man in the middle attacks include:
- Use cryptography. If you encrypt the data before transmitting it, the attacker can still intercept it but cannot read it or alter it. If the attacker cannot read it, he or she cannot know which parts to alter. If the attacker blindly modifies your encrypted message, then the original recipient is unable to successfully decrypt it and, as a result, knows that it has been tampered with.
- Use Hashed Message Authentication Codes (HMACs). If an attacker alters the message, the recalculation of the HMAC at the recipient fails and the data can be rejected as invalid.