Chapter 7

Protecting against advanced attacks

horizontal line

# Comparing common attacks

**MITM**

Man in the middle (MITM) attacks: Attacker inserts himself in the middle, and intercepts conversation. Kerberos can prevent this with mutual authentication, so an attacker can’t insert himself between 2 systems without their knowledge.

**ARP poisoning**

ARP resolves IP addresses to MAC addresses using the following protocol:

1. ARP request: It broadcasts the IP address and essentially asks “Who has this IP address?”
2. ARP reply: Computer with that IP address responds with their MAC address, and the requesting computer saves it in the cache

Attackers can easily create an ARP reply packet with spoofed MAC address, and poison the ARP cache with false MAC addresses. This can mitigate a MITM or DoS attack.

**DNS attacks**

**DNS poisoning**: Corrupts the DNS servers by changing the IP address mapped to a particular hostname, so user can be redirected to IP of malicious site.

**Pharming**: Similar to DNS poisoning

**Amplification attacks**

They increase the amount of traffic the victim receives, causing their system to crash.

**Smurf attack**: Type of DDoS attack where attacker spoofs the victim’s IP address and sends multiple ICMP echo requests to a computer. The victim is then spammed with ICMP replies.

**DNS amplification**: Attacker requests as much data as possible from DNS servers with a spoofed victim’s IP. The victim is then flooded with all the data, causing their system to crash.

**NTP amplification attack**: Network time protocol attack, where attacker uses spoofed victim’s IP address to use the monlist command which requests a list of the last 600 hosts that connected to the NTP server. The victim is then flooded with all this data.

**Password attacks**

Account lockout policies help protect against these attacks

**Brute force attacks**

* **Online attacks:** Attackers try to discover the password from an online system
* **Offline attack**: Attackers use a downloaded database or captured packet scan.

**Dictionary attacks:** Attempts every word in a dictionary of common passwords

Pass the hash attacks: Attacker discovers hash of the user’s password, and uses it to log on. This usually happens when a protocol such as Microsoft LAN management (LM) and NT LAN manager (NTLM) sends the hash in an unencrypted format

**Birthday attacks**

This is caused by a hash collision, where the hashing algorithm creates the same hash from 2 different passwords. They can be prevented by increasing the number of bits used in the hash, which increases the number of possible hashes

**Rainbow table attacks**

They attempt to discover the password from the hash, from a huge database of passwords and their precomputed hashes. If an attacker has a hash of a password...

1. The app guesses a password/uses one from a password dictionary
2. App hashes the guesses password
3. It compares the original hash with the guessed password hash. If the hashes are the same, it has the password
4. If they are not the same, it repeats steps 1-3 until finding a match

Some rainbow tables include hashes for every possible combination of chars, up to 8 chars length

Salting can prevent this kind of attack, as it adds random chars before hashing. This causes these attacks to fail because the password is not the same to one that might be in the dictionary.

**Plaintext attacks**

Can be prevented by not using legacy and deprecated encryption algorithms.

**Known plaintext attack**: Attacker has samples of plaintext and ciphertext, and can use these to discover the encryption and decryption method.

**Chosen plaintext attack**: Attacker doesn’t have access to all plaintext, so has to use different methods to decrypt the plaintext he actually has

**Ciphertext attack**: Attacker only has encrypted text. This attack is most time consuming.

**Hijacking and related attacks**

**Typosquatting**: Attackers buy domains that are close to legitimate ones, e.g. comptai.org. This can be used to host a malicious site.

**Clickjacking**: Tricks users into clicking something other than what they are clicking, using HTML frames

**Session hijacking**: Attacker impersonates a user on a website by using cross-site scripting attacks to read cookies installed on their systems. The attacker can then spoof these cookies and impersonate the user on a website.

**Domain hijacking**: Attacker changes the registration of a domain name without the owner’s permission, effectively stealing control of the domain from them.

**Man in the browser**

A type of proxy trojan that infects web browsers, and has tools such as keyloggers, and data capture techniques, so they can access info a user entered over the internet

**Driver manipulation**

Shimming provides a solution to make older drivers compatible with OSs, and involves shims which intercept a driver call, causing alternate code to be run instead of the original.

Refactoring code is the process of rewriting code without changing it’s internal behaviour, to correct software design problems.

Attackers can manipulate drivers by creating shims, or rewriting internal code, tricking OSs into using malicious drivers

**Zero day attacks**

Zero-day exploits are undocumented and unknown to the public, but attackers actively seek them out in order to exploit them.

**Memory buffer vulnerabilities**

**Memory leak**

A bug in a computer application that consumes more and more memory until it causes the system to crash.

**Integer overflow**

If an application attempts to store a value which exceeds it’s memory capacity. This can be prevented by checking the buffers to see if they can store data generated by the app, and with adequate input/error handling

**Buffer overflows**

An application receives more input than it can handle, allowing access to memory locations beyond the application’s buffer, enabling an attacker to write malicious code into this area of memory.

Buffer overflow attacks are a type of DoS attacks

Attackers can use NOP(No operation) commands to execute their code in the desired place, by writing hex90 chars (which are a NOP sled). These commands skip through the memory until they get to the desired memory location, and then the malicious code executes

**DLL injection**

Dynamic link Library is a compiled set of code that people can use. An attacker can attach a DLL to a running process, allocate memory for it, and then execute the malicious DLL functions.

**Secure coding concepts**

**Input validation**

Validating that input in text forms is legitimate, e.g. in phone number or email format, can help prevent SQL injection or cross scripting attacks. This can be done by checking for relevant chars, blocking HTML code, and chars found in SQL injection attacks.

**Client and server side manipulation**

**Client-side input validation:** validation code is included in HTML script, and doesn’t allow submission to server unless client input is validated

**Server input validation:** Checks inputted values when they reach the server, to ensure the client hasn’t bypassed client-side checks. It is the final check before the server uses the data, and more secure than client-side validation.

**Race conditions**

When 2 applications try to access a resource at the same time.

**Error handling**

Improper error handling can give attackers information about an application.

* **Errors should be general,** in order to minimise info given to attackers
* **Detailed info should be logged,** to make it easier for developers to identify what went wrong.

**Cryptographic methods**

Devs can provide a digital certificate for the code, which includes the hash of the code. This identifies the author, and the hash ensures integrity of the code.

**Code reuse**

Code reuse can lead to dead code which is never executed. They can also be caused by logic error.

SDK: Software development kits are third-party libraries, which have software tools which devs can use.

**Code obfuscation**

Replacing variable names, and strings with hex characters, in order to camouflage the code. However, an attacker can still decipher the code

**Code quality and testing**

Methods of testing:

* **Static code analysers**: Examine code without executing it, for example as the dev is typing.
* **Dynamic analysis**: Checks the code as it is running, with fuzzing. This uses a computer app to send random data to the program, in order to cause it to fail so devs can fix the bug.
* **Stress testing:** Simulating a live environment, with attacks that may happen
* **Sandboxing:** Isolated area using VM, where code is analysed
* **Model verification:** Ensuring that software meets specs

**Development life cycle models**

Software development life cycles (SDLC) give structure to dev projects

**Waterfall**

1. Requirements
2. Design
3. Implementation
4. Verification
5. Maintenance

In waterfall, you can’t go back to a previous stage, so it lacks flexibility.

**Agile**

Uses iterative cycles instead of stages. Product is created, then tested, and process is repeated from the start. It emphasises interaction between the team and customers. It is good if a client has a clear idea, else it draws out the development process

**Secure devops**

Agile development process, with security concepts in mind, with the following concepts:

* **Security automation:** Use automated tests on the code
* **Continuous integration**: Merging code into a central repository, allowing roll back changes if needed
* **Baselining**: Applying changes to the baseline code everyday and building from these changes. Bugs are corrected quicker this way.
* **Immutable systems:** We can create a secure image of the code, and deploy it as immutable to ensure it stays secure.

**Version control**

Tracks versions of software as it is updated, including who made the update and where. It allows devs to roll back changes if needed

**Provisioning and deprovisioning**

Provisioning refers to giving an app the appropriate functionality. Deprovisioning refers to removal of app upon deletion.

**Application attacks**

**Database concepts**

Normalisation: Organising tables and columns to reduce redundant data and improve database performance.

A database is in 1st normal form if:

* Each row is unique and identified by a primary key.
* Related data is kept in a separate table
* None of the columns include repeating groups, e.g. firstname and lastname must be separated, and not put together as “name”

2nd normal form if:

* It is in 1NF
* Non-primary key attributes are dependent on the composite private key.

3rd normal form if:

* In 2NF
* All columns that aren’t primary keys are only dependent on the primary key, so none of the columns in the table are dependent on the non-primary key attributes.

**Protecting against SQL injection**

Stored procedures pass on user input to a mini program, which performs data validation but also separately handles the input. Preventing direct access to the database prevents SQL injection attacks.

**Cross-site scripting (XSS)**

Attackers can embed malicious HTML/JS code into a website’s code, but this can be prevented by the use of security encoding library to sanitise code.

**Cross-site request forgery (XSRF or CSRF)**

This is when attackers create a malicious link and embed it on a website.

Devs can prevent these attacks by using dual authentication and forcing user to manually enter credentials prior to performing actions.

This can be prevented by using XSRF tokens, which are long numbers appended to the end of a URL. An attacker can’t guess this, so they can’t craft malicious links against the site.

**Frameworks and guides**

* Regulatory: They are based on relevant laws and regulations
* Non regulatory: A common standard and best practices
* National vs international: some are used within a single country, e.g. NIST, others internationally, e.g. ISO
* Industry-specific: Some frameworks only apply to certain industries

**ACRONYMS**

MITM: Man in the middle, a type of attack that can be prevented by kerberos

LM: MS LAN management, which sends hashes in an unencrypted format

NTLM: NT LAN manager, similar to LM

SDK: Software development kits, third party code libraries with code that devs can use

SDLC: Software development life cycle