Notas AEV

Slide 1 Vulnerabilities:

Vulnerability-> It's a weakness in a system, it allows an attacker to violate a reasonable security policy for that system. They increase with software dev.
They allow the user to get restricted data, execute commands as another user (pose as another entity) and DOS (affect availability)

CIA -> confidentiality, integrity, availability

OWASP -> community that helps with web application security with free resources. They have a top 10 vuln.

CWE -> Common weakness enumeration, is a category system for software weaknesses and vulnerabilities.

Bugs vs Fault -> bug is an error in the code, Fault is because of the architecture or design of the software.

For public software, vulnerabilities are tracked by vendors globally and publicly, this helps attackers and defenders.

The ones that are not public by vendors, are used as weapons and can be sold for crimes or to the same companies that developed the software.

Vulnerability tracking systems: Dictionary of public vulnerabilities and exposures.

- CVE: A vulnerability of this dictionary. ex: CVE-2020-1472

The CVE contains:

- Basic information
- References
- Severity score
- links to solutions

CVSS - Common Vulnerability Scoring System, this results in a score from 0 to 10 with how critical the vulnerability is.
It is based on equations with multiple steps, such as attack vector, integrity, exploit availability etc.

How should a research proceed when a vulnerability is found:

- If private, contact the entity
- If says nothing, can sell or does not care, this can lead to 0-day vulnerability

0-day vulnerability: It's a vulnerability that is not known to those that want it fixed. It can be exploited by hackers, and if it has a big severity score (CVSS) can disrupt the globe when publicly known.

The controlled way:

- Researcher informs vendor
- They fix it and distribute it
- Most machines get fixed
- CVE maybe

Full vulnerability disclosure:
If the researcher finds the vulnerability and goes public with it, the company is pressured to fix it as quickly as possible, and hackers might use it while it is still not fixed.

Slide 2 Information Leakage:

Network access:

- Can enumerate hardware
- Information leaked
- Discover exploit

Information leakage:

A system that provides information enabling the attacker to discover known vulnerabilities. He can focus on that information.

Most relevant informations leakage:

Errors:

- Errors form the infrastructure, if the errors are different with different inputs, this is good information, or provides information about internal processes, data or versions. It can lead to enumeration.
- To mitigate, do not provide verbose output to the user, log it.

Web sources and Support Files:

- Additional information that is left in a website, for example by developers testing. This can lead to finding sensitive information, like backup files, robots.txt, README, logs, folders etc.

Cookies:

- Cookies sent in HTTP responses provide information about the server stack, like php, asp.net etc.

Ports:

- Check which services are running in a system, by sending a TCP SYN and checking the response. A service normally operates on the same port.
- Port scanning does exactly this by testing for a range of ports and checking the response. It might also indicate the existence of a firewall.
- The best way to mitigate is with firewalls that detect the enumeration

Banners:

- Textual snippets provided on connection.
- They help the attacker to know what software is running.
- Banner grabbing with probe entities for their banners.
- Examples are ssh, ftp etc.
- Mitigations can be removing banners or creating fake ones.

OS Fingerprinting:

- Normally Network stacks don't behave consistently, but they might have specific behaviours.
- With a sequence of probes, and by observing the responde, we can match the behaviour to a database and possibly find what system is running.

- It can be tricky, as machines can behave in different ways.
- Mitigation is for example, restricting the open ports, detecting enumeration with firewall and obfuscating behaviours.

Slide 3 Vulnerability Research:

Vulnerability Research: Process of finding vulnerabilities.

Vulnerability Assessment: Analyze, evaluate and review entities.

Assessment: Determines how good/bad something is, aims to help improve systems done before and after audits. It's specific.

Audit: Determines compliance to a standard, with multiple control points from the standard.

Penetration Test: This is when an individual tests infrastructure and systems in general normally from the outside with no specific knowledge. Test the domain and impact of the test.An assessment is specific, and with knowledge of the application.

It’s important to do security assessments because, as organizations get larger, with multiple servers, applications, communication systems etc, standard security measures like Firewalls, WAF and IDS are limited in their capabilities.

Scope -> What is going to be evaluated, the systems/software/approaches are going to be considered and used.
Too broad: Mimics a powerful attacker, but is expensive, and can never end.
Too narrow: Mimics a focused attack, it's cheaper, but as it is focused, easy issues can be missed.

Security assessments limitations:
- Is only valid in a given point in time
- Researcher must be aware of latest vulnerabilities
- Limited to the scope

Types of assessments:

- Active: Send information, try to craft arguments, send payloads etc. this can disrupt the systems. (ex: MiTM, DoS)
- Passive: Listen to communications and logs inside the organization, this has a minimal impact.
- External: Focus on the public exposed services, ports, routers etc. Can find vulnerabilities and enables the deployment of countermeasures at the FW.
- Host Based: Focus on misconfigurations, permissions, updates etc. Finds vulnerabilities as if the attacker already has access to the systems.(this is more in general in a machine)
- Network: Focus on the network infrastructure, like rules and misconfigurations. Finds information that can be leaked.
- Wireless: Focus on the wireless communications similar to the network assessment, but with specific tools.
- Application: Focus on a single application, on the application io, logic, authentication and authorization etc. This will find bugs or flaws in the application.

Vulnerability Management Life Cycle:

- Establishing a baseline: what is going to be assessed.

- Vulnerability Assessment: finding vulnerabilities. They can be done with the following strategies:
    - Black Box: Researchers have no information. Assumes an actor with a specific set of resources. Aims to mimic outside attacks.
    - White Box: Full documentation of the system, with a limited scope. Finds bugs and faults at all scoped domains. Assumes an actor in any location. Extensive and expensive analysis of domains.
    - Gray Box: Some information is provided to the researcher. Aims to find faults and bugs at a limited set of scoped domains.

  Takes in consideration the scope and priorities and constructs a detailed report with: the found vulnerability, the affected entities, and recommendations to handle it. You don't exploit a vulnerability chain.

- Risk Assessment: Takes the vulnerability assessment report and assesses the risk. Assigns risk indicators. They also assess the combined exploit chain.

- Remediation: Tries to fix the vulnerability, by correcting software bugs or flaws, implementing specific configurations, updating software etc.
  When this is not possible, Try to reduce the impact of a successful exploitation like sandboxing.

- Verification: Verifies the effectiveness of the fix, from the same scope!

- Monitoring: Deploy mechanisms to detect the vulnerability being explored, by configuring firewalls, IDS etc.

SCAP - Security Automation Protocol

Protocol that assesses automatically the security status of a system
Objectives:
- Track system status
- Monitor system security policies
- Identify vuln
- Quantify risks

Enumeration:
- CVE : Common Vulnerabilities and Exposures -> Vulnerabilidades
- CCE : Common Configuration Enumeration -> Configurações
- CPE : Common Platform Enumeration -> Plataformas software que tens instalado

Languages:
- OVAL -> How can assess if the system complies to a security policy
- PCIL
- XCCDF -> Configurations Policy with fixes

Metric:
- CVSS

Slide 4 SQL Injections:

Injection: The software constructs all or part of a command using externally-influenced input.

Impact:
- Confidentiality
- Access Control
- Integrity
- Non-Repudiation

Input provided is not validated or filtered.
Never trust external data, sanitize all external data.
Never trust internal systems or private API's
Never trust data coming from the database
Never call a command using external data directly.

SQL Injection:

Database is expected to have ACID properties.
ACID:
- Atomicity
- Consistency
- Isolation
- Durability

Exemplos SQLi:

$result = mysql_query(" SELECT * from Users where(username='john' or 1=1); -- ' and password='abc');");

The user controls the execution flow.

SQL Injection can be leveraged to other attacks.

Some DBMS can execute code (Sql server de BD) or change files.

SQLi Types:

- In Band: Payload is provided and the result is determined directly.
  Arrives from the same channel that was used to provide the payload.

- In Band - Error based: Knowing the input is vulnerable to a sql injection with ‘ for example.

- In Band - Union Based: Exploits union operator to get information on the query.

- Blind - SQLi occurs, but the result is not provided to the attacker.

- Blind - Content based: Using forced booleans like, id=2 and 1=1 checks if id=2 exists based on the response vs id=2 and 1=2

- Blind - Time Based: Using delay commands to check if the system takes the malicious predictable time to execute

- Out Of Band: Using other services to see results or execute commands, with the database.

How to avoid:

- Sanitize data: check types, find invalid characters.
- Use prepared statements: Multiple programming languages have their own version of prepared statements, there is a separation between structure and data.

Slide 5 OS Command Injection:

Improper Neutralization of Special Elements used in an OS Command.

- Can lead to indirect command execution in the operating system (RCE)
- Can allow the attacker to specify commands that aren't normally accessible.

- Can lead to privilege escalation

Many languages can run exec.

Argument exploitation:
- programs run as part of a normal operation, but a crafted payload may lead the user to execute commands before or after the expected program.
- Caso do ping e nome do ficheiro do TAR dos slides.

Environment variables:
- Command execution affected by env variables.
- PATH env variable.

CGI -> Way of executing scripts that interact with clients through a web server.
By setting the headers as env variables as it prepares to start the script execution, these can be modified to allow remote code execution on the machine.

Parameter Expansion:
- Special character that the shell expands
- The best example is *
- A command with * lists filenames provided by bash
- ls * on a folder with File.txt and ola.txt does the following: ls File.txt ola.txt
- If files have names of valid commands they can be executed with *

How to avoid:
- Never execute system commands from an application
- Careful with dependencies.

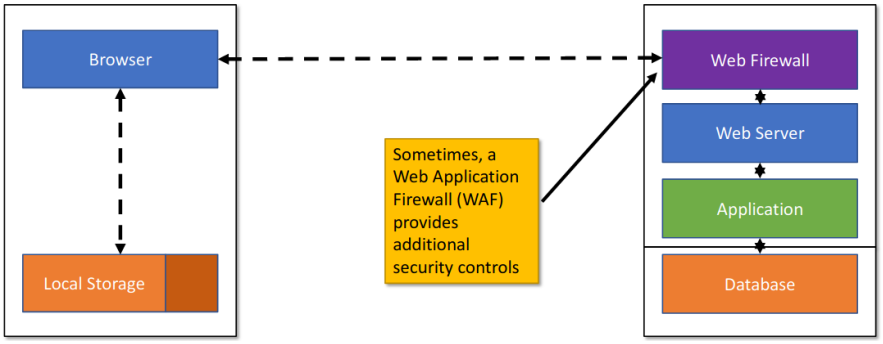
Sanitize inputs, Escape special characters etc, but this is still very vulnerable.
Limit with permissions.
Use sandbox / vm / docker etc.
Use absolute path for commands.

Slide 6 Broken Authentication:

Normally related to authentication and session management implemented incorrectly.

Http basics:

- Ask DNS
- Connect to port 80
- Send GET/POST



Http is stateless by design!
To keep state tokens and cookies are used.

Server only replies to requests, never contacts clients directly.
Using web sockets they can contact clients directly.
Any participant can send messages.

Control must reside in the server-side context.

Authentication:
Determine the identity of an entity.

Authentication Header:

- Basic: Base64 user and pass
- Digest: server sends the request of authentication with a nonce, then user sends back the nonce with the hash of the user/password, then server compares to stored hashes.

Sessions: As http is stateless, you need a way to keep state.
4 ways:

- Referer header
- Session_ID
- Cookie
- JWT

GET is used to request info
Post is used to update info

The Referer request header: contains the address of the page making the request. The idea is that if you come from a page that you have to be authenticated to access, you are logged in.

Session ID: value set by the server in html/js, can be included in URLs or headers.

Cookies: information that server sends to a client, that the client stores and sends in all preceding requests to that server.
They are used as a token enabling authorization, when the cookie is set, the user is authenticated.

JWT: A concatenation of the header, payload and signature. 2 Tokens are sent by the server when the user is authenticated, the access token and the refresh token. When the access token expires, the refresh token is used to get a new access token.

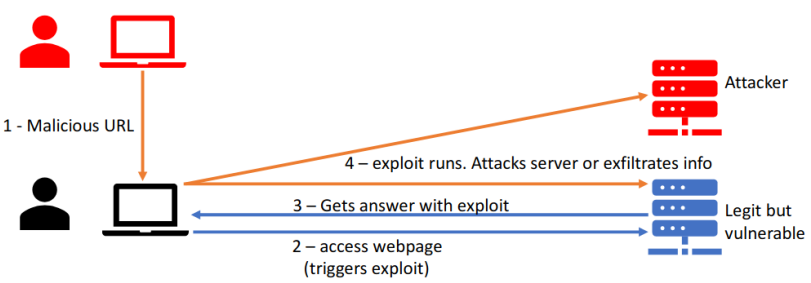
What makes applications vulnerable:
- Guesses credentials
- Brute force
- Stores in vulnerable format
- Credentials sent over unencrypted connections
- No multi-factor authentication

What applications should not do:
- Avoid passwords.
- Use secure storage
- Require rotation, but not frequent rotation
- Rate limit authentication functions
- Use multi-factor authentication

Session hijacking: stealing cookies, tokens, session ids etc.
Brute force can also be applied.
Session Fixation: SID must be invalidated before authorizing a new session, else the person can send https://server.com?SID=KNOWN_TO_ATTACKER.

Slide 7 XSS:

Reflected XSS: The application includes unvalidated unescaped user input as part of the html output.
If the server is vulnerable, you can send a legit link to a person, the xss can redirect to a compromised url and exfiltrate data.
The impact is moderate.



Stored XSS: For example a database stores unsanitized input. Input can be then viewed by another user later.

Dom XSS: Introduced with js on the dom, unsanitized input.

CSRF: Client browser issues requests to external server. If i have a link to post on facebook, if i give it to someone they just press it and publish it as if it has their own.

By using Synchronizer Tokens on a form, the person with the malicious request cant know the token at the time as it changes.
Cookie-to-header a cookie will be saved and when a request is done, the js puts the cookie on the header.
SameSite=Strict makes it so that no external js has access to the cookie.

Same Origin Policy (CORS):

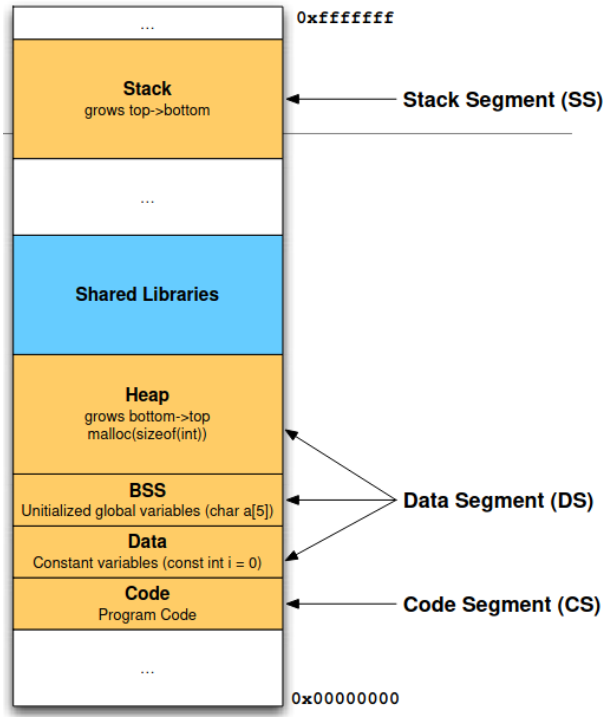
Sites may require external resources.
Same origin policy restricts which documents can be loaded from another origin.
Cross Origin Request Sharing -> data is exchanged with trusted sites while using a relaxed same origin policy mode.

A CORS request is sent first to check the cors policy. After the cors response responds with a Access-Control-Allow-Origin header.
If this Access-Control-Allow-Origin is a * it is always allowed.

If trust on websites is broken, it serves as nothing.
The origin header can also be manipulated if the Access-Control-Allow-Origin spoofing of the response.
Caching the preflight response can also be dangerous on the browser.

Slide 8 Buffer Overflow:

Memory structure:



Compiler will place variables are he seems adequate

Classic Overflow

Buffer Over-read: with %s or %2$llx

Stack overflow:

- Overwrite local variables
- DOS
- Change program flow
- Injection of malicious code

Canaries can be used to detect overflow.

ROP attacks: chain gadgets to execute malicious code. ‘pop rdi; ret’
command address
gadget address
command address
system address

Heap:

- Overflow normal
- dangling
- fastbin: double free

Slide 10 Mobile Security:

Trusted Execution Environment: environment that runs in parallel with the system that stores cryptographic material.

Can't use root normally.
Internal Storage is encrypted.

Intents are a Message Passing mechanism for IPC

Mobile apps are frequently populated with bugs/mistakes as other applications. Because the code is available to clients, inspection and abuse becomes more frequent.

Decompiled : jadx to a higher level representation.

Administrator Interfaces or api special access.

Hardcoded secrets.

Visibility issues, call activities that are not supposed to be called.
Provider exposure to query data.
Logging