Applications: Vulnerabilities and Exploits

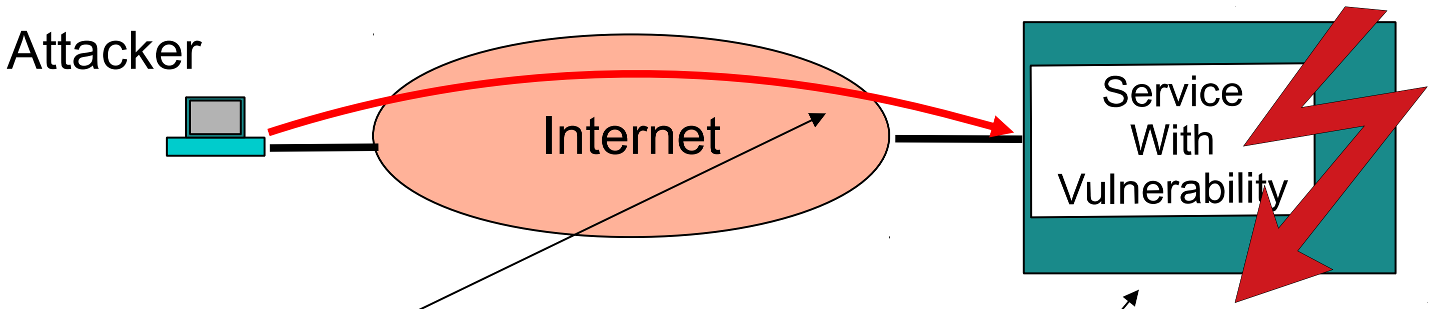
Content

- (Remote) Exploits
- Examples
- Protection against Exploits
- Summary

Vulnerability, Exploit, Remote Exploit

- Exploit: (Code to) exploit a vulnerability in a program/operating system by an attacker in order to compromise a system/resource
- Classification „how“:
  - Local: Attacker needs physical access or local user account (privilege escalation).
  - Remote: Over a network without the necessity of previous access to the system (and without interaction with human victim).
  - Web Application, Zero-Day

Remote Exploit



Attacker exploits vulnerability over a network

Vulnerability allows attacker to compromise system/resource

Aims of (Remote) Exploits

- Illegitimate access to data
- Code execution / administration rights
- Denial of Service

Examples / Important Types

- Misconfigurations
- Backdoors
- Command Injection
- Cross-Site Scripting
- Buffer Overflows

Security Misconfiguration

- One of most common security risks
- Occurs because of
  - insecure default configurations
  - incomplete or ad hoc configurations
- Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion

(Source: OWASP)

Malware: Backdoor

- Malware is software running without user‘s consent which is designed to potentially compromise the system.
- Backdoors are a type of malware allowing an attacker to bypass security mechanisms and gain access to a system
- Attackers often modify legitimate programs to backdoors after compromising a system so they can gain future access to it by using the backdoor

Command Injection

- Excution of arbitrary commands on the host operating system via a vulnerable application
- Possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell
- Usually executed with the privileges of the vulnerable application
- Command injection attacks are possible largely due to insufficient input validation

(Source: OWASP)

SQL Injection

- “Injection” of an SQL query via the input data from the client to the application
- Possible damage:
  - read sensitive data from the database
  - modify database data (Insert/Update/Delete)
  - execute administration operations on the database
  - issue commands to the operating system

(Source: OWASP)

Cross-Site Scripting (XSS)

- Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. (Source: OWASP)
- Client-side browser will execute the script
- Script may access cookies, session tokens, or other sensitive information retained by the browser and used with that site
- Aim: Run malicious script in client‘s browser with malicious intent
- Stored vs. Reflected

Stored Cross-Site Scripting (Stored XSS)

- Maliciously injected script is permanently stored on the target servers, such as in a database, in a message forum, visitor log, comment field, etc
- Victim retrieves malicious script from server when it requests information

(Source: OWASP)

Reflected Cross-Site Scripting (Reflected XSS)

- Injected script is reflected off the web server as response that includes some or all of the input sent to the server as part of the request
- Delivered to victims via another route, e.g., e-mail message, or on some other website
- Injected code travels from user to vulnerable web site which reflects the attack back to the user’s browser
- Browser executes the code because it came from a “trusted” server

(Source: OWASP)

Client-Side Cross Site Scripting (DOM-based XSS)

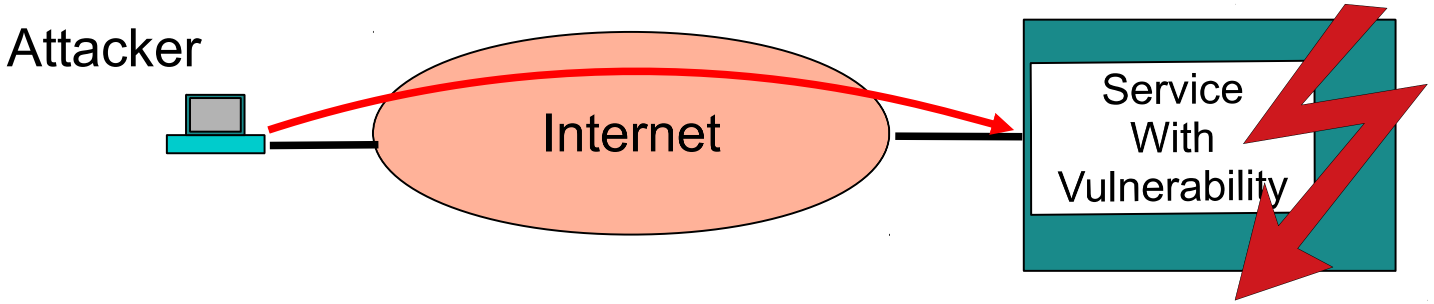
- Injected malicious script never leaves the client
- May also be stored or reflected, e.g., as part of a URL which is not sent to the server (e.g., document.location.href), or it could be an element of the HTML, and the sink is a sensitive method call that causes the execution of the malicious data (e.g., document.write).” reflected off the web server as response that includes some or all of the input sent to the server as part of the request

  (Source: OWASP)

Cross-Site Request Forgery (CSRF)

- Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated (Source: OWASP)
- Tricks the victim into submitting a malicious request, e.g., to change password, change email, make transaction
- Inherits authenticaion e.g., from existing connection or cookie
- Aim: Make a state change on the server side

Protection against such Attacks



Protection against such Attacks

- Switch off services/software which is not used or needed
- Limit access to system/services
  - Firewall
- Careful configuration
- Keep software up to date
  - Patches
  - Updates
- Careful programming
  - Use tools that allow mitigation, particularly for Web Apps

Missing Bits and Pieces

- Exploitation of vulnerabilities in network protocols
- How to avoid vulnerabilities during programming
- Classification and Collection of Exploits (Common Vunerabilities and Exposures)
- CVSS (Common Vulnerability Scoring System)

Summary

- Remote Exploit refers to (Code to) exploit a vulnerability in a program/operating system by an attacker in order to compromise a system/resource
- Aim is to gain acces to a system (data / code execution)
- Many different types of vulnerablilties exist
- Protection mechanisms may mitigate the exposition against attacks

References

- Common Vulnerabilities and Exposures: https://cve.mitre.org/
- Exploit-DB: https://www.exploit-db.com/
- Open Web Application Security Project Top 10: https://owasp.org/www-project-top-ten/
- Metasploitable2: https://metasploit.help.rapid7.com/docs/metasploitable-2
- Damn Vulnerable Web Application: http://www.dvwa.co.uk/