



Ethical Hacking Essentials (EHE)

1 Master of Science in Cybersecurity



Dr. Kamal Alieyan

Everyone Smart University

Faculty of IT – 2022/2023

Module 07

Web Application Attacks and Countermeasures

Module Objectives

- 1 Understanding Web Server Concepts and Attacks
- 2 Understanding Different Web Server Attack Tools and Countermeasures
- 3 Overview of Web Application Architecture and Vulnerability Stack
- 4 Understanding Different Web Application Threats and Attacks
- 5 Understanding Different Web Application Attack Tools and Countermeasures
- 6 Overview of Different Types of SQL Injection Attacks
- 7 Understanding Different SQL Injection Tools
- 8 Understanding Different SQL Injection Attack Countermeasures



Web Server Attacks

Module Flow

Discuss Various Web Server Attacks



01



02



Discuss Web Server Attack Countermeasures

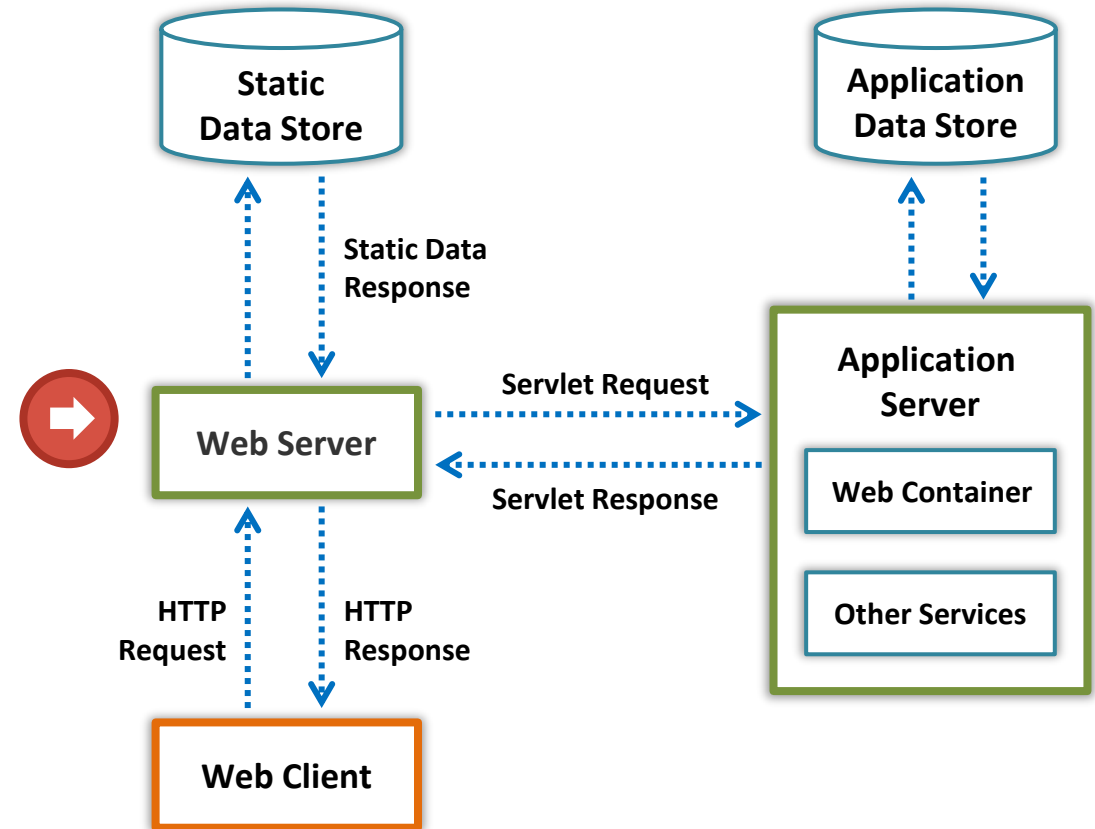


Web Server Operations



A web server is a computer system that **stores, processes, and delivers web pages** to clients via HTTP

Typical Client-Server Web Server Operation



Web Server Components



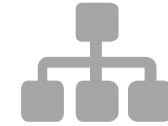
Document Root

Stores critical HTML files related to the web pages of a domain name that will be served in response to the requests



Server Root

Stores server's configuration, error, executable, and log files



Virtual Document Tree

Provides storage on a different machine or disk after the original disk is filled up



Virtual Hosting

Technique of hosting multiple domains or websites on the same server



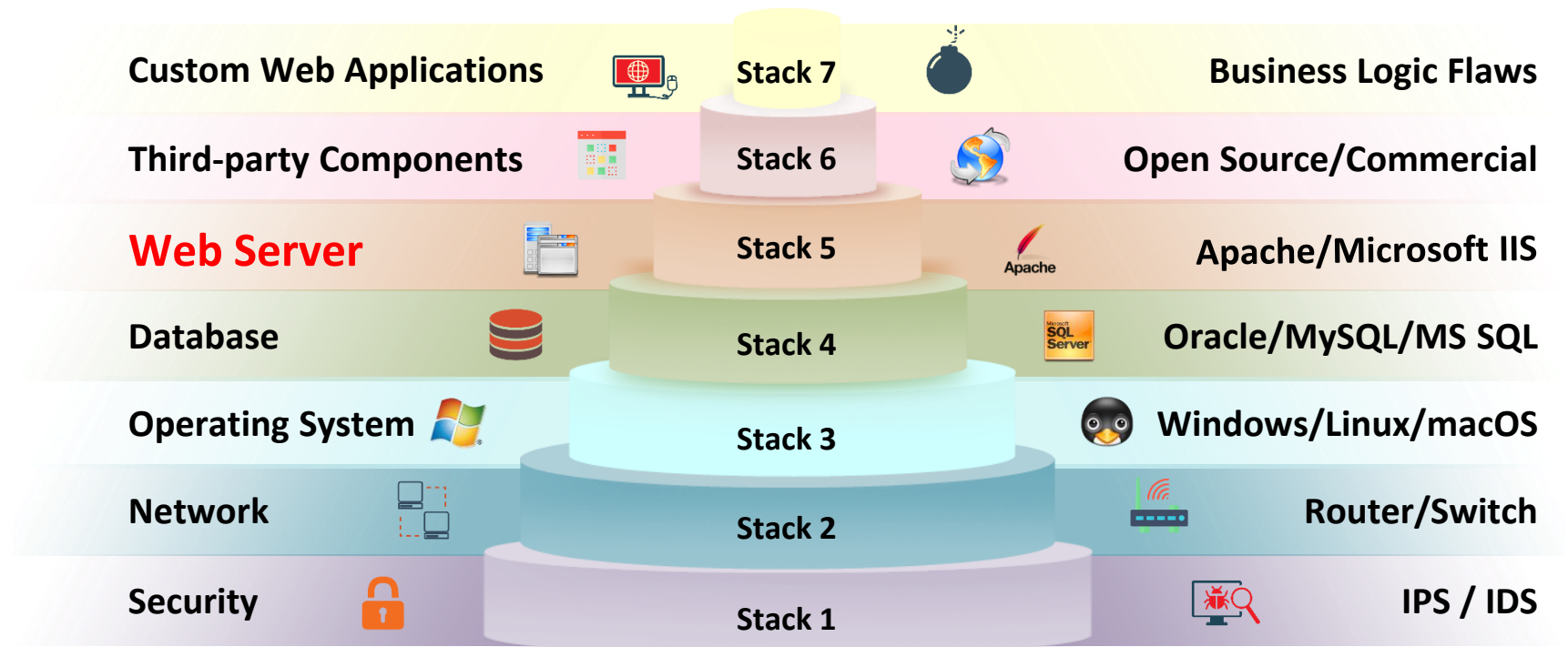
Web Proxy

Proxy server that sits between the web client and web server to prevent IP blocking and maintain anonymity

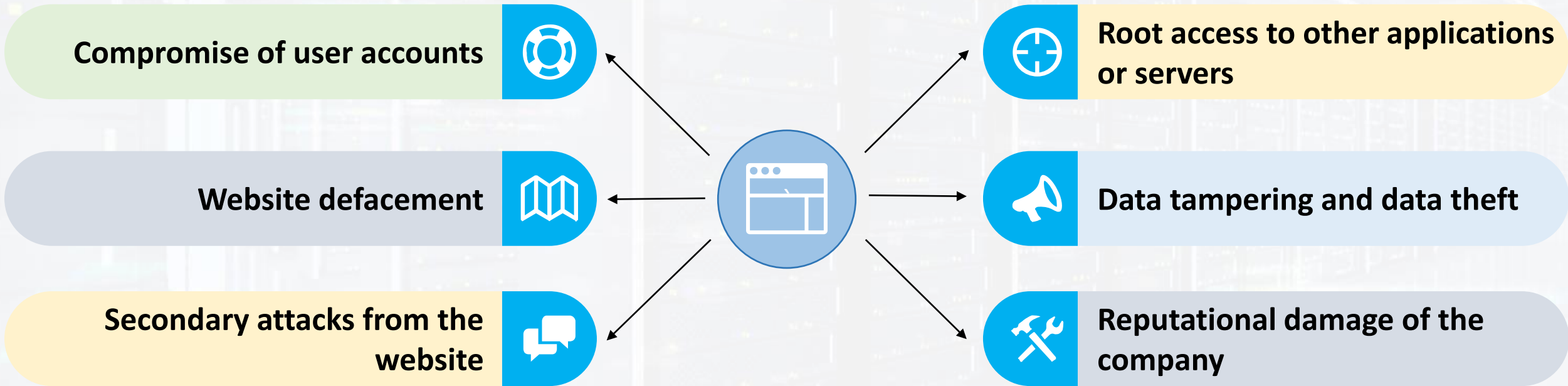


Web Server Security Issues

- ❑ Attackers usually target **software vulnerabilities** and configuration errors to compromise web servers.
- ❑ Network and OS level attacks can be well defended using proper **network security measures** such as firewalls, IDS, etc. However, web servers can be accessed from anywhere via the Internet, which renders them **highly vulnerable** to attacks



Impact of Web Server Attacks



Why are Web Servers Compromised?



Improper file and directory **permissions**



Server installation with **default settings**



Enabling of **unnecessary services**, including content management and remote administration



Security conflicts with business ease-of-use case

Lack of proper security policies, procedures, and maintenance



Improper authentication with external systems



Default accounts having default passwords, or no passwords



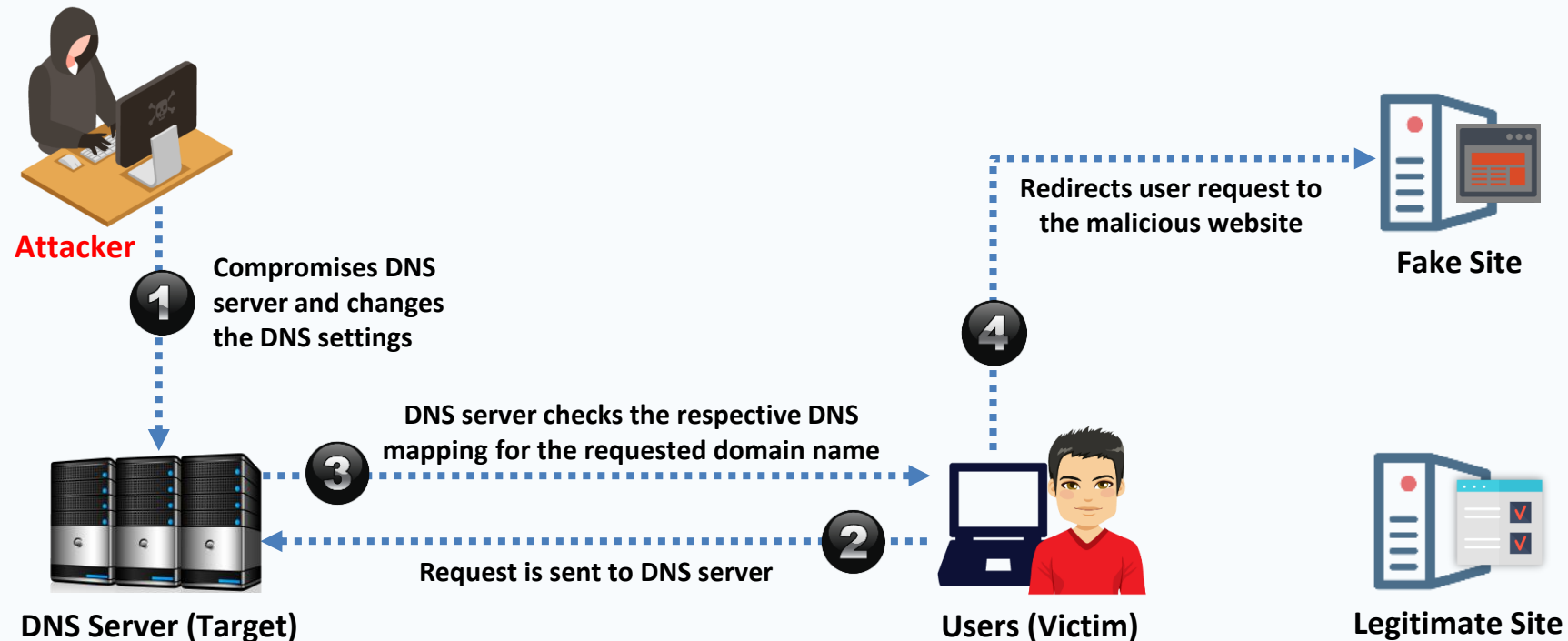
Misconfigurations in web server, operating systems, and networks



Web Server Attacks: DNS Server Hijacking



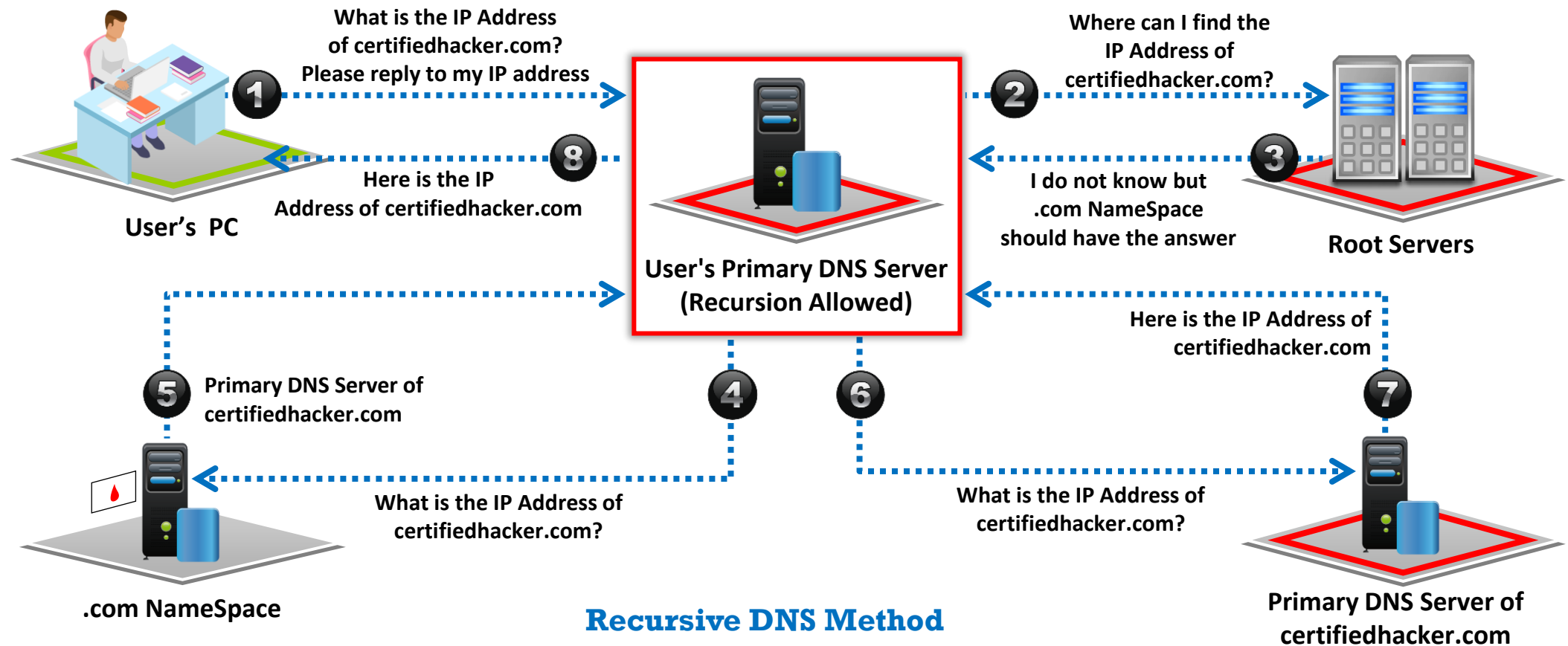
Attacker compromises the DNS server and **changes the DNS settings** so that all the requests coming towards the target web server are redirected to his/her own malicious server



Web Server Attacks: DNS Amplification Attack



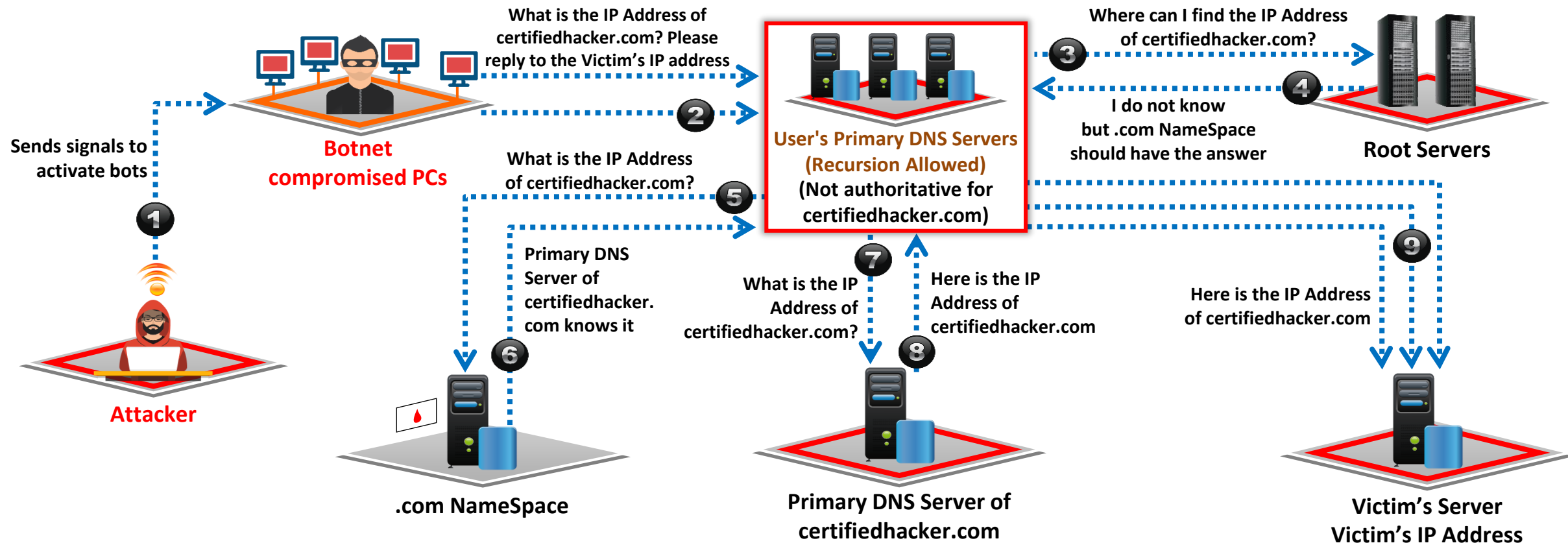
Attacker takes advantage of the **DNS recursive method** of DNS redirection to perform DNS amplification attacks



Web Server Attacks: DNS Amplification Attack (Cont'd)



Attacker uses compromised PCs with **spoofed IP addresses** to amplify the DDoS attacks on victims' DNS server by exploiting the DNS recursive method



Web Server Attacks: Directory Traversal Attacks

- ❑ In directory traversal attacks, attackers use the **../ (dot-dot-slash)** sequence to access restricted directories outside the web server root directory.
- ❑ Attackers can use the **trial and error method** to navigate outside the root directory and access sensitive information in the system.

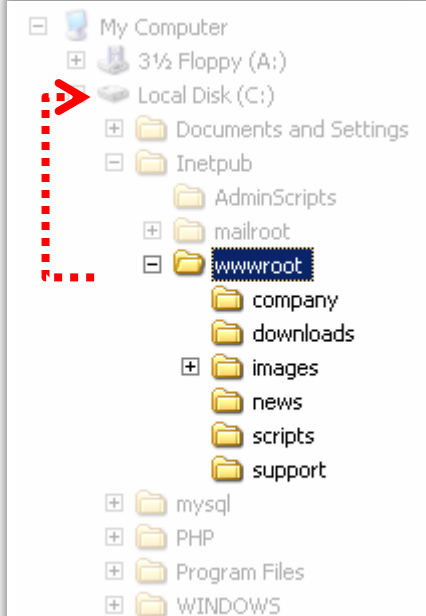


`http://server.com/scripts/..%5c../Windows/System32/cmd.exe?/c+dir+c:\`

Volume in drive C has no label.
Volume Serial Number is D45E-9FEE

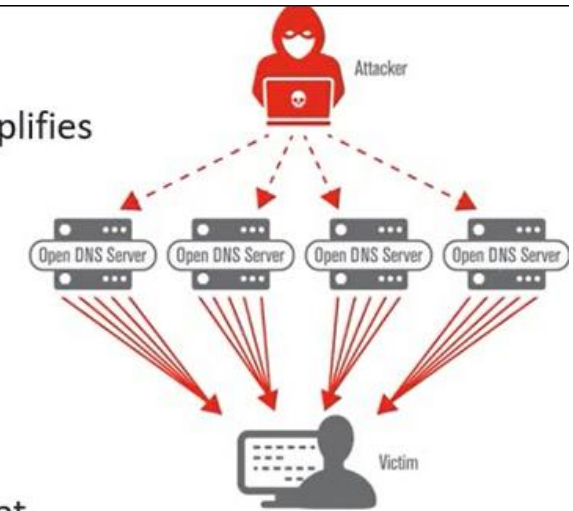
Directory of C:\

06/02/2017 11:31 AM	1,024 .rnd
09/28/2017 06:43 PM	0 123.text
05/21/2017 03:10 PM	0 AUTOEXEC.BAT
09/27/2017 08:54 PM	<DIR> CATALINA_HOME
05/21/2017 03:10 PM	0 CONFIG.SYS
08/11/2017 09:16 AM	<DIR> Documents and Settings
09/25/2017 05:25 PM	<DIR> Downloads
08/07/2017 03:38 PM	<DIR> Intel
09/27/2017 09:36 PM	<DIR> Program Files
05/26/2017 02:36 AM	<DIR> Snort
09/28/2017 09:50 AM	<DIR> WINDOWS
09/25/2017 02:03 PM	569,344 WinDump.exe
7 File(s) 570,368 bytes	
13 Dir(s) 13,432,115,200 bytes free	

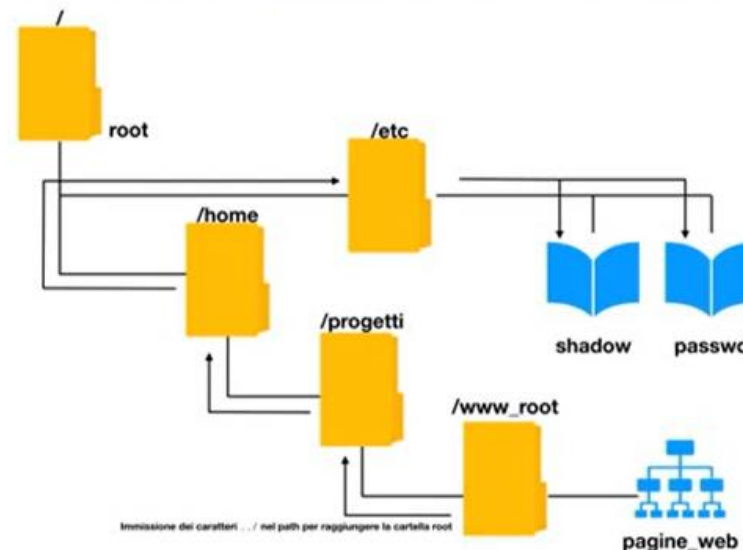


Web Server Attacks: Directory Traversal Attacks

- **DNS Amplification** - uses recursive DNS to DoS a target; amplifies DNS answers to target until it can't do anything.(Botnet)



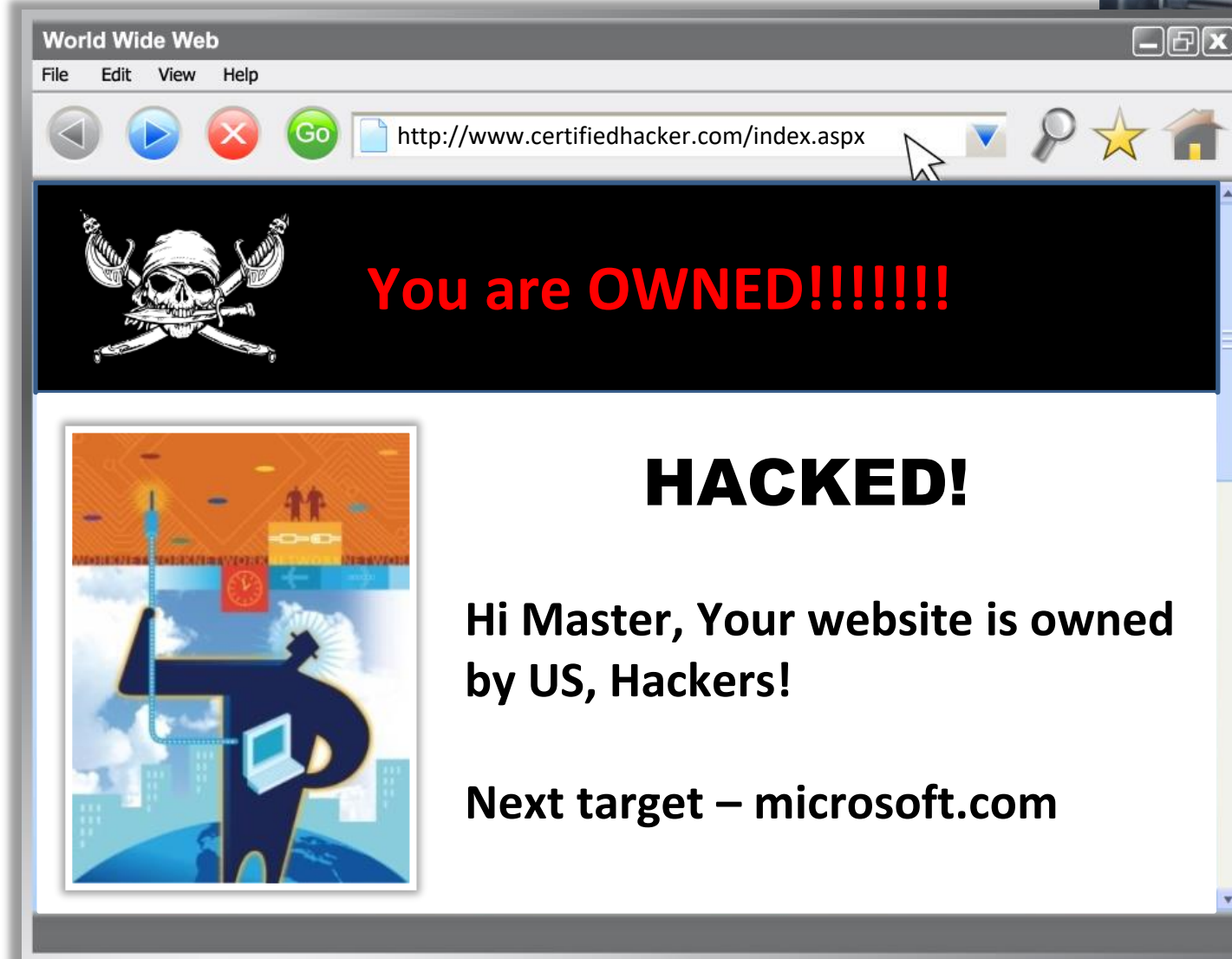
- **Directory Traversal** (../ or dot-dot-slash) - requests file that should not be accessible from web server
 - Example: <http://www.example.com/../../../../etc/passwd>



Web Server
Attacks

Web Server Attacks: Website Defacement

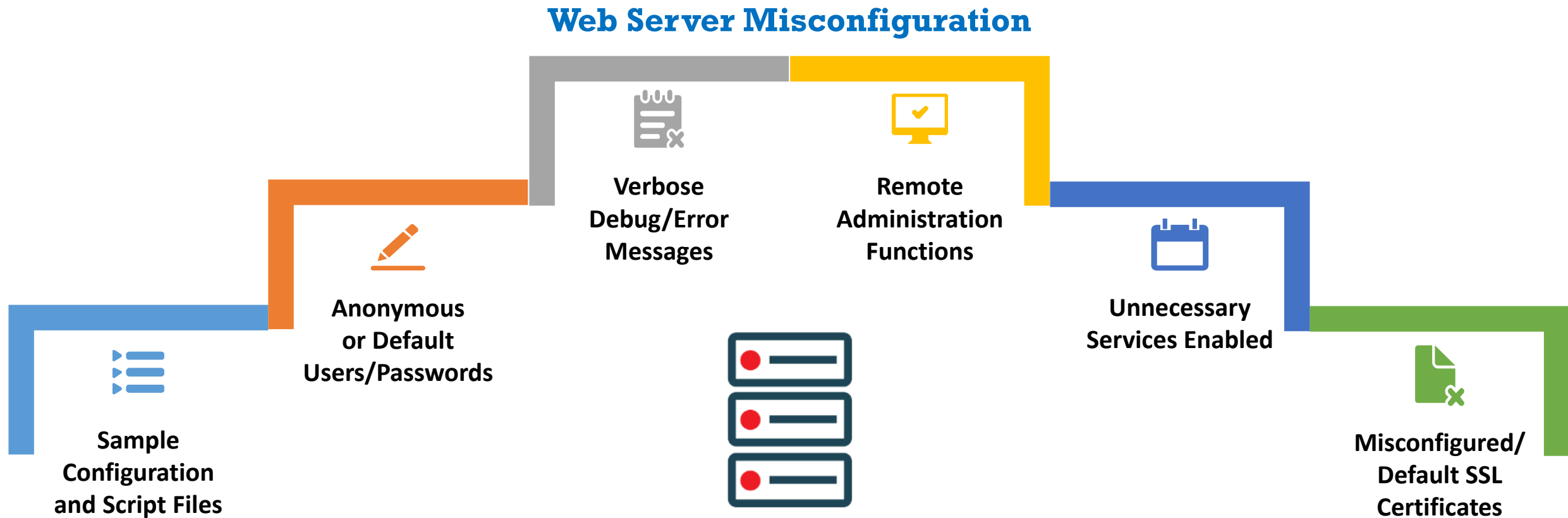
- ❑ Web defacement occurs when an intruder **maliciously alters the visual appearance of a web page** by inserting or substituting provocative, and frequently, offending data
- ❑ Defaced pages **expose visitors to some propaganda** or misleading information until the unauthorized changes are discovered and corrected



Web Server Attacks: Web Server Misconfiguration



Server misconfiguration refers to **configuration weaknesses in web infrastructure** that can be exploited to launch various attacks on web servers such as directory traversal, server intrusion, and data theft



Web Server Attacks: Web Server Misconfiguration (Cont'd)

Web Server Misconfiguration Examples

This configuration allows anyone to view the **server status** page, which contains detailed information about the web server being currently used, including information about the **current hosts** and requests being processed.

```
<Location /server-status>  
SetHandler server-status  
</Location>
```

httpd.conf file
on an **Apache** server

This configuration generates **verbose error messages**



```
display_error = On  
log_errors = On  
error_log = syslog  
ignore_repeated_errors =  
Off
```

php.ini file

Web Server Attacks: HTTP Response-Splitting Attack

01

HTTP response splitting attack involves **adding header response data into the input field** so that the server splits the response into two responses

02

The attacker can **control the first response to redirect the user to a malicious website** whereas the other responses are discarded by the web browser

Server Code

```
String author =  
request.getParameter(AUTHOR_PARAM);  
...  
Cookie cookie = new Cookie("author",  
author);  
cookie.setMaxAge(cookieExpiration);  
response.addCookie(cookie);
```

Input = Jason

```
HTTP/1.1 200 OK  
...  
Set-Cookie: author=Jason  
...
```

Input = JasonTheHacker\r\nHTTP/1.1 200 OK\r\n

First Response (Controlled by Attacker)

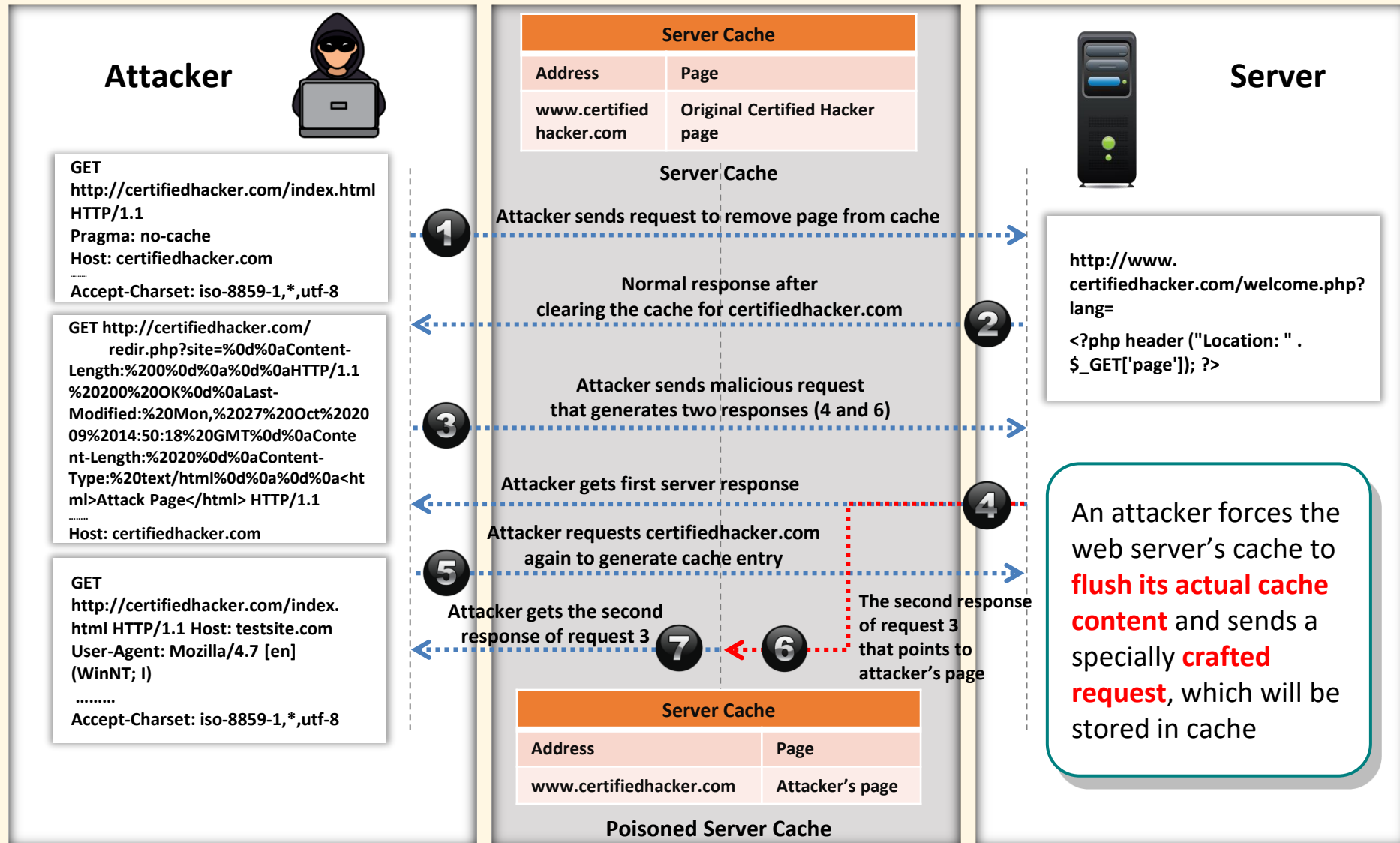
```
Set-Cookie: author=JasonTheHacker  
HTTP/1.1 200 OK  
...
```

Second Response

```
HTTP/1.1 200 OK  
...
```

Web Server Attacks: Web Cache Poisoning Attack

- ❑ Web cache poisoning attacks the **reliability of an intermediate web cache source**
- ❑ In this attack, the attackers **swap cached content** for a random URL with infected content
- ❑ Users of the web cache source can **unknowingly use the poisoned content** instead of the true and secured content when requesting the required URL through the web cache



Web Server Attacks: SSH Brute Force Attack

01

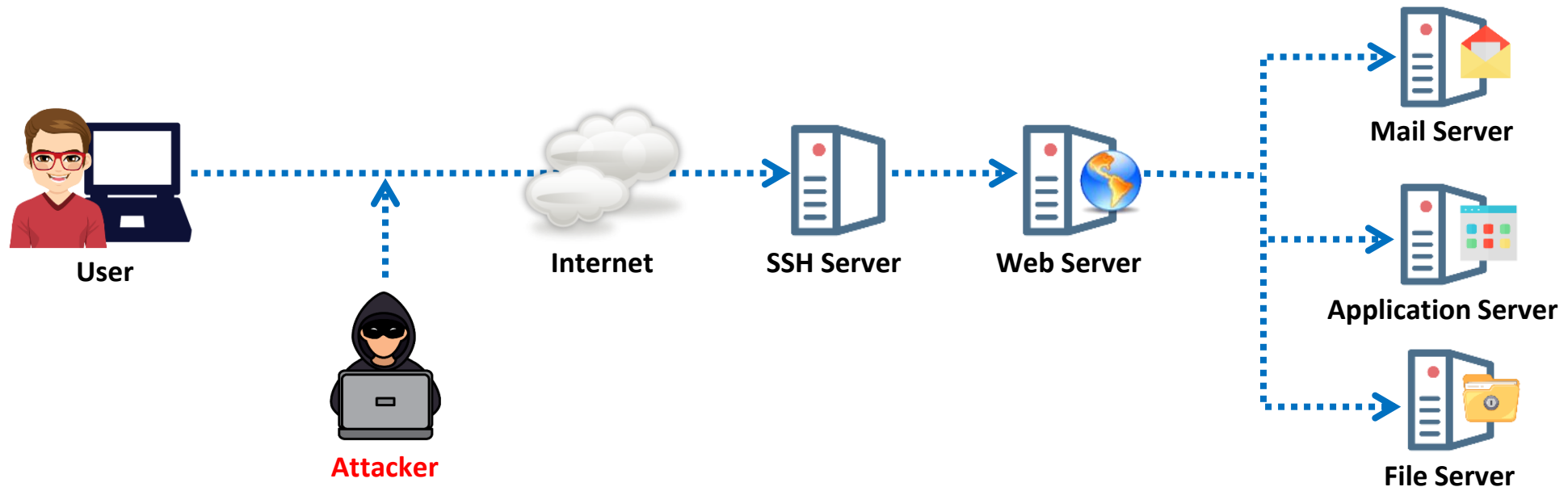
SSH protocols are used to create an **encrypted SSH tunnel** between two hosts to transfer unencrypted data over an insecure network

02

Attackers can brute force SSH login credentials to gain **unauthorized access to an SSH tunnel**

03

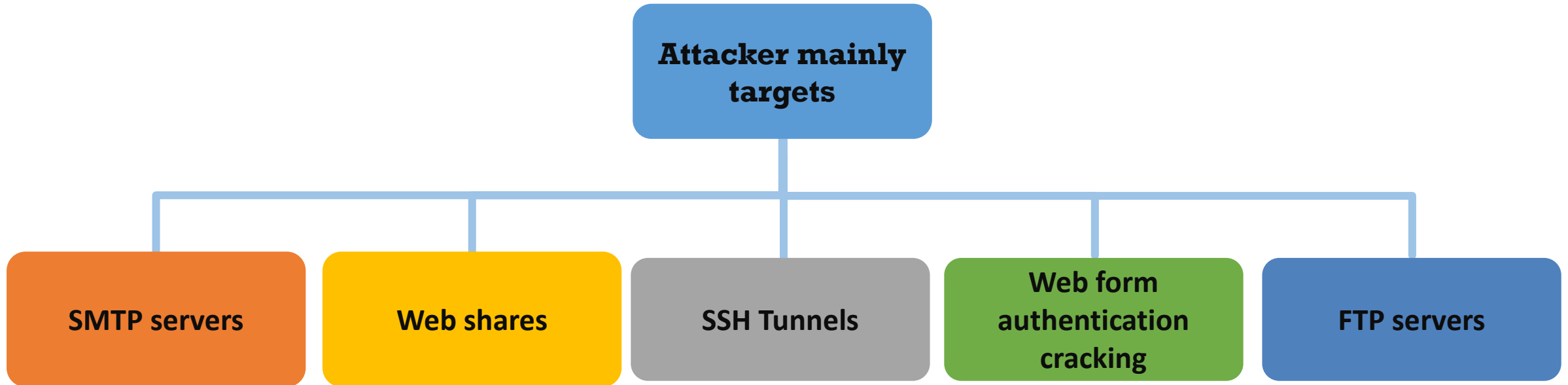
SSH tunnels can be used to **transmit malwares** and other exploits to victims without being detected



Web Server Attacks: Web Server Password Cracking

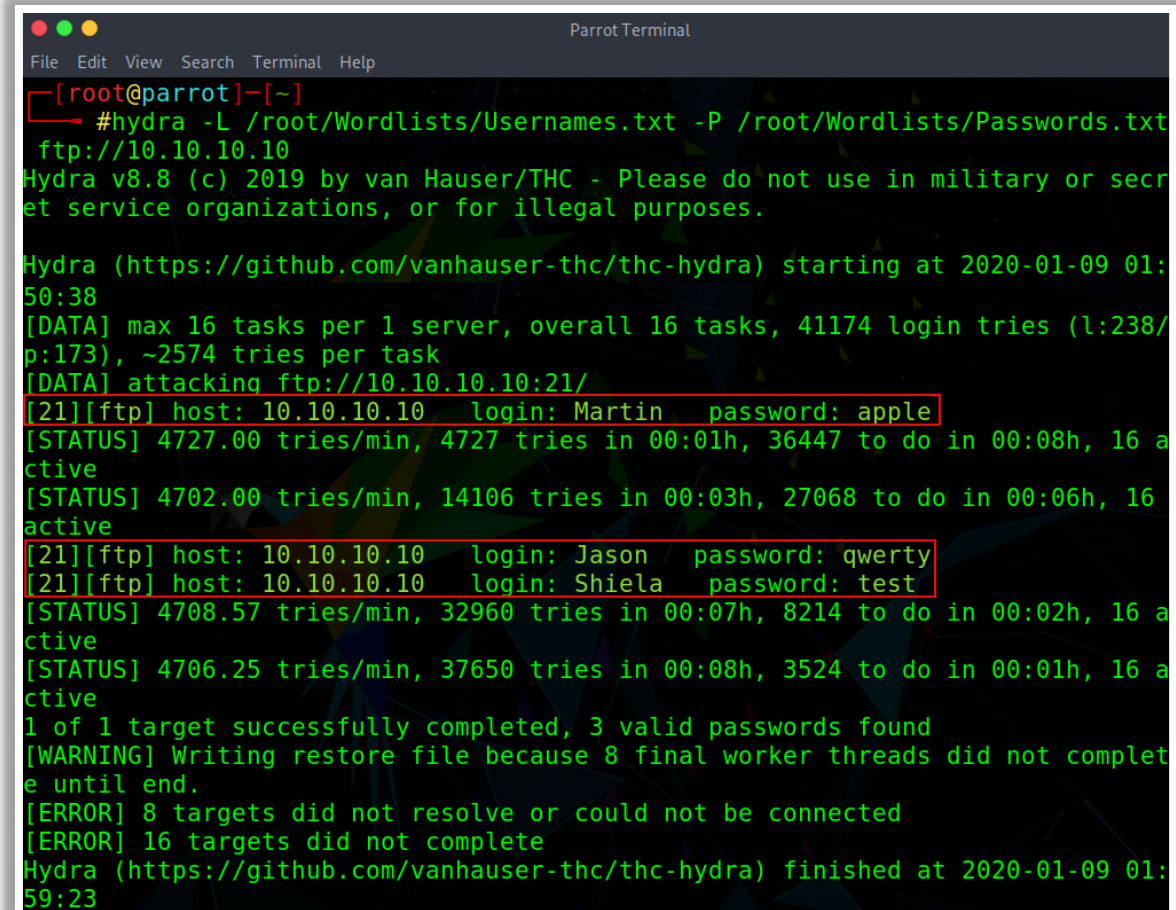


- ☐ An attacker tries to exploit weaknesses to hack **well-chosen passwords**
- ☐ The most **common passwords** found are password, root, administrator, admin, demo, test, guest, qwerty, pet names, etc.



- ☐ Attackers use different methods such as **social engineering, spoofing, phishing**, using a Trojan Horse or virus, wiretapping, and keystroke logging

Web Server Attacks: Web Server Password Cracking (Cont'd)



```
Parrot Terminal
File Edit View Search Terminal Help
[~][root@parrot]~#hydra -L /root/Wordlists/Usernames.txt -P /root/Wordlists/Passwords.txt ftp://10.10.10.10
Hydra v8.8 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

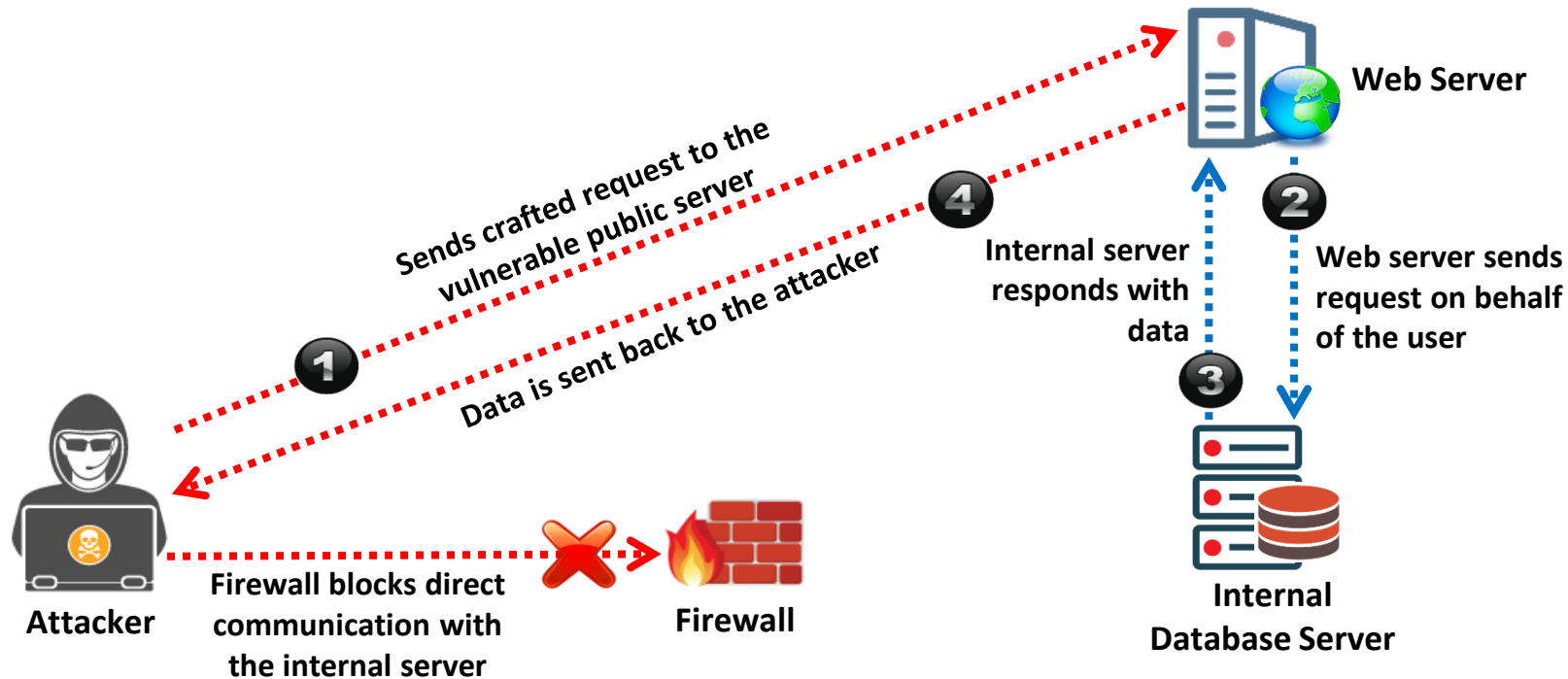
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-01-09 01:50:38
[DATA] max 16 tasks per 1 server, overall 16 tasks, 41174 login tries (l:238/p:173), ~2574 tries per task
[DATA] attacking ftp://10.10.10.10:21/
[21][ftp] host: 10.10.10.10 login: Martin password: apple
[STATUS] 4727.00 tries/min, 4727 tries in 00:01h, 36447 to do in 00:08h, 16 active
[STATUS] 4702.00 tries/min, 14106 tries in 00:03h, 27068 to do in 00:06h, 16 active
[21][ftp] host: 10.10.10.10 login: Jason password: qwerty
[21][ftp] host: 10.10.10.10 login: Shiela password: test
[STATUS] 4708.57 tries/min, 32960 tries in 00:07h, 8214 to do in 00:02h, 16 active
[STATUS] 4706.25 tries/min, 37650 tries in 00:08h, 3524 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
[WARNING] Writing restore file because 8 final worker threads did not complete until end.
[ERROR] 8 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-01-09 01:59:23
```

<https://github.com>

- ❑ Passwords can be cracked **manually** by guessing or by performing dictionary, brute force, and hybrid attacks using **automated tools** such as THC Hydra, and Ncrack

Web Server Attacks: Server-Side Request Forgery (SSRF) Attack

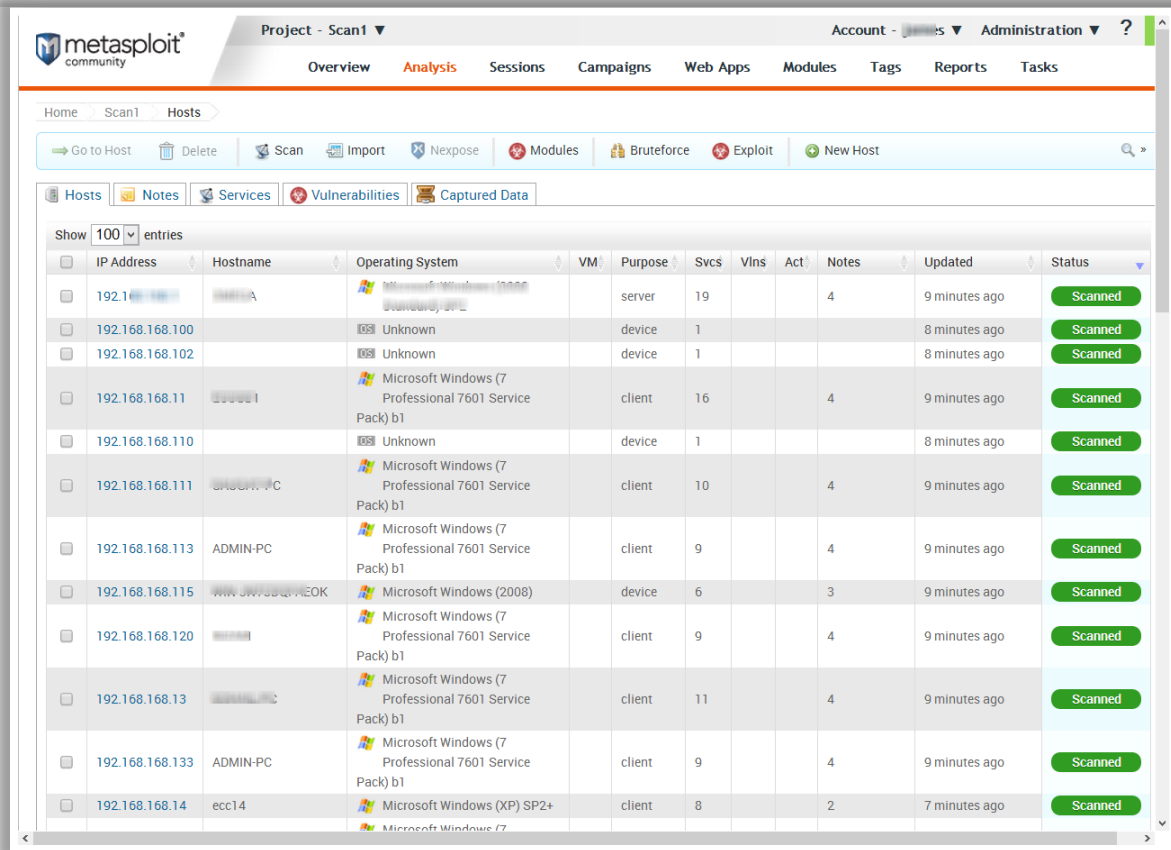
- ❑ Attackers exploit SSRF vulnerabilities in a public web server to **send crafted requests** to the internal or back end servers
- ❑ Once the attack is successfully performed, the attackers can perform various activities such as **port scanning, network scanning, IP address discovery**, reading web server files, and bypassing host-based authentication



Web Server Attack Tools

Metasploit

An exploit development platform that supports fully automated **exploitation of web servers**, by abusing known vulnerabilities and leveraging weak passwords via Telnet, SSH, HTTP, and SNM



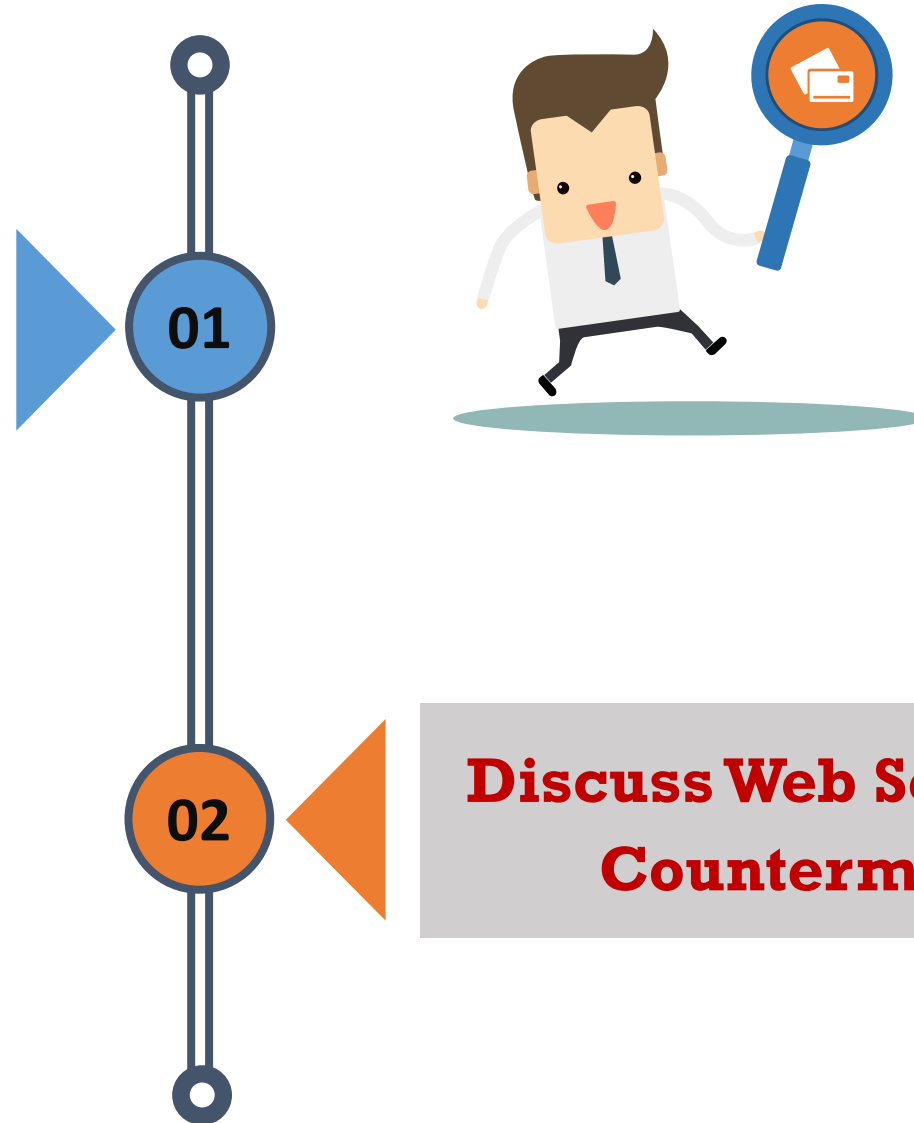
<https://www.metasploit.com>

Web Server Attack Tools

- ✓ Immunity's CANVAS (<https://www.immunityinc.com>)
- ✓ THC Hydra (<https://github.com>)
- ✓ HULK DoS (<https://github.com>)
- ✓ MPack (<https://sourceforge.net>)
- ✓ w3af (<https://w3af.org>)

Module Flow

Discuss Various Web Server Attacks



Discuss Web Server Attack Countermeasures

Web Server Attack Countermeasures

01

- Apply **restricted ACLs** and block remote registry administration
- Secure the **SAM** (Stand-alone Servers Only)

04

- Remove all unnecessary file shares including the **default administration shares** if not required
- Secure the shares with restricted **NTFS permissions**

02

Ensure that security related settings are **configured appropriately** and access to the metabase file is restricted with hardened **NTFS permissions**

05

Relocate sites and virtual directories to **non-system partitions** and use IIS Web permissions to restrict access

03

Remove unnecessary ISAPI filters from the web server

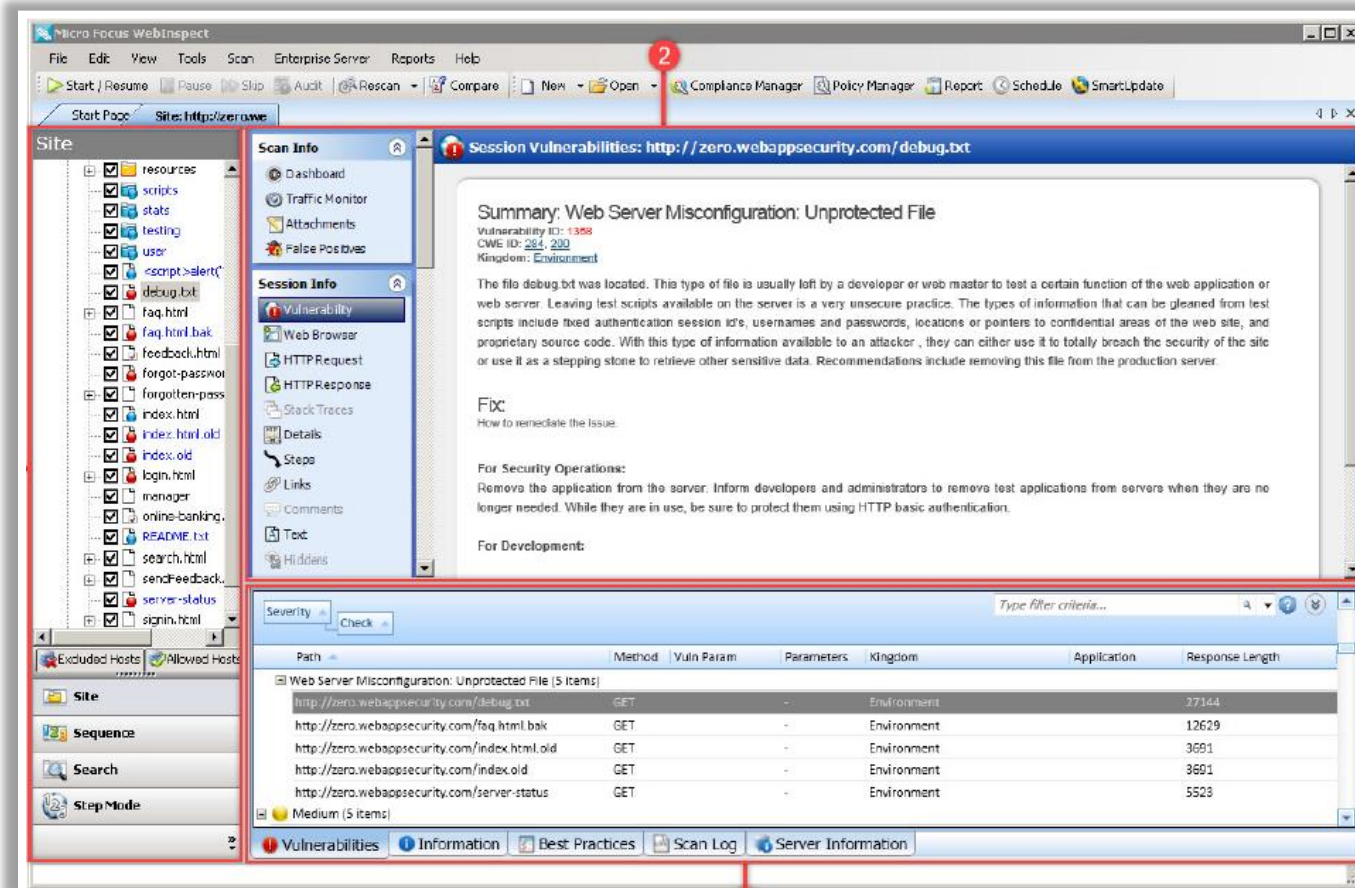
06

Remove all unnecessary **IIS script mappings** for optional file extensions to avoid exploiting any bugs in the ISAPI extensions that handle these types of files

Web Server Security Tools

Fortify WebInspect

Fortify WebInspect is an **automated dynamic testing solution** that discovers configuration issues and identifies and prioritizes security vulnerabilities in running applications



<https://www.microfocus.com>



Acunetix Web Vulnerability Scanner
<https://www.acunetix.com>



Retina Host Security Scanner
<https://www.beyondtrust.com>



NetIQ Secure Configuration Manager
<https://www.netiq.com>



SAINT Security Suite
<https://www.carson-saint.com>



Sophos Intercept X for Server
<https://www.sophos.com>



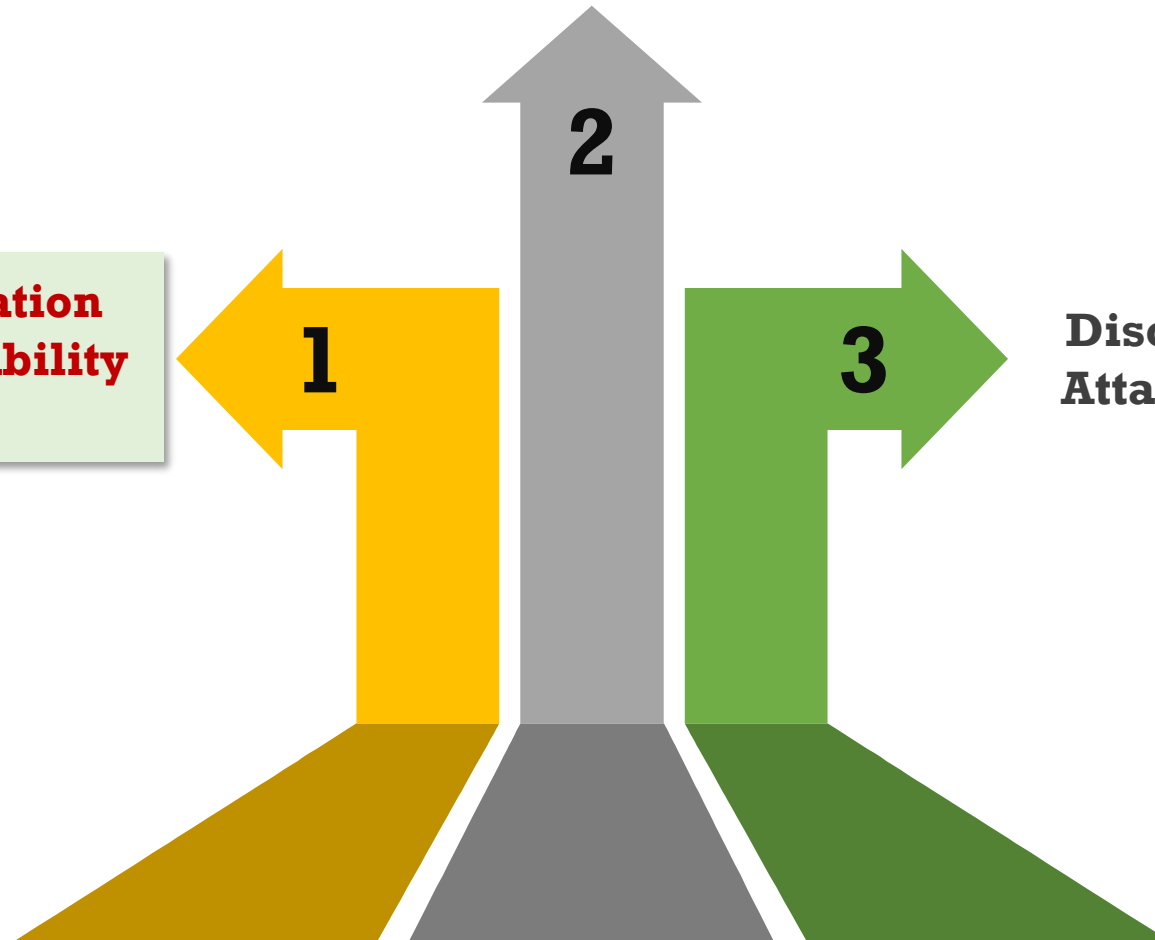
Web Application Attacks

Module Flow

**Discuss Web Application
Threats and Attacks**

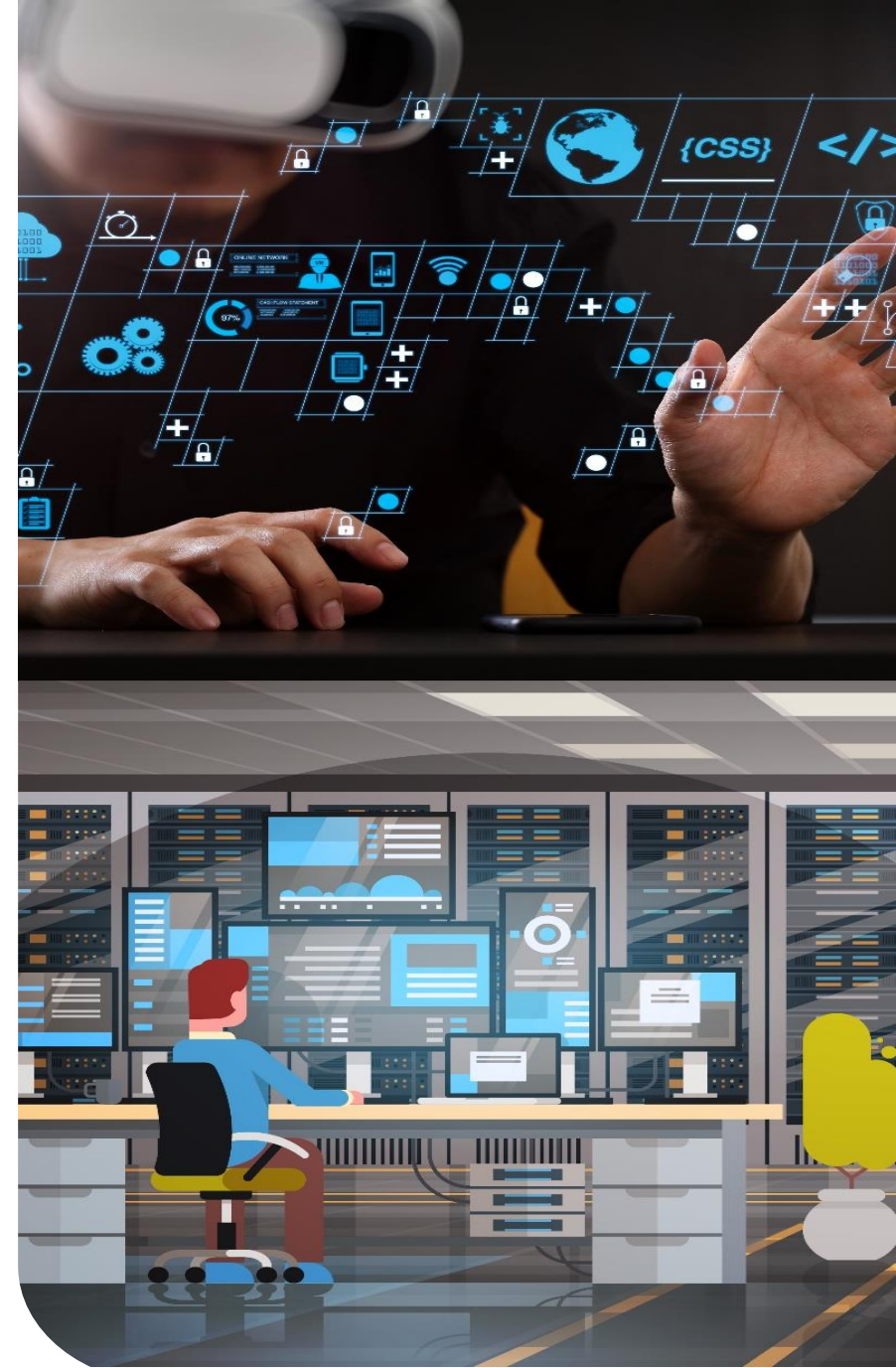
**Understand Web Application
Architecture and Vulnerability
Stack**

**Discuss Web Application
Attack Countermeasures**

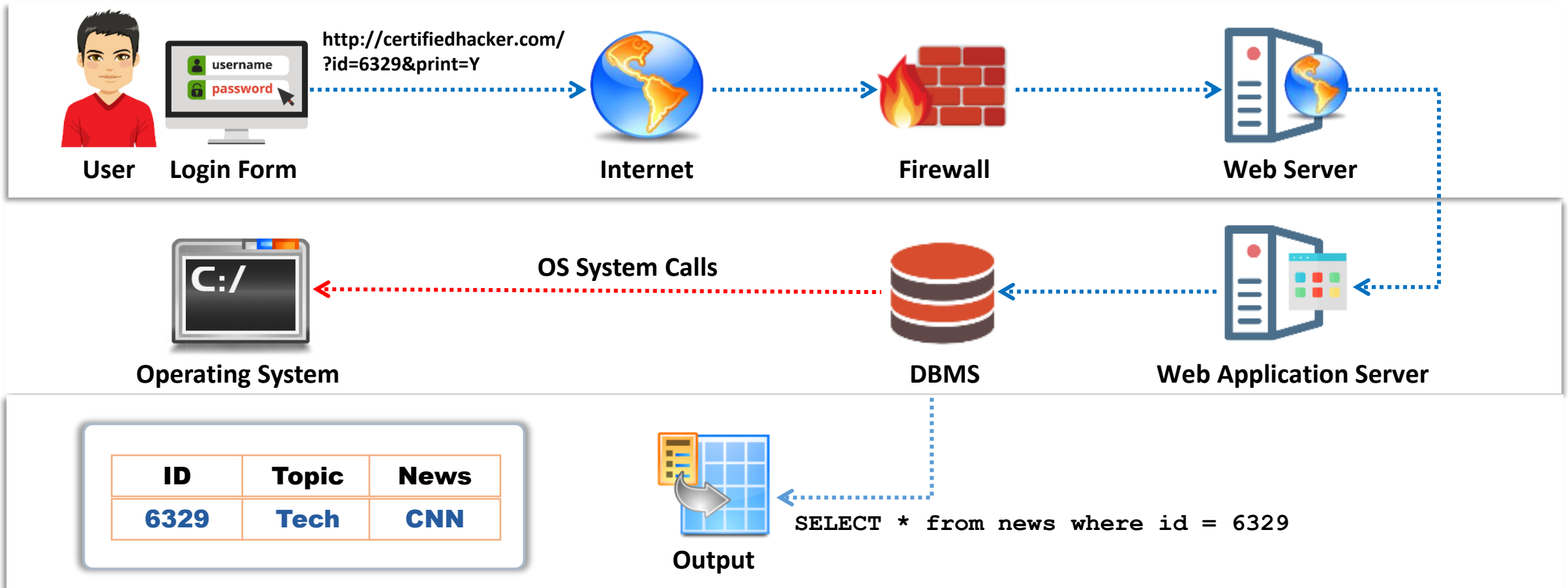


Introduction to Web Applications

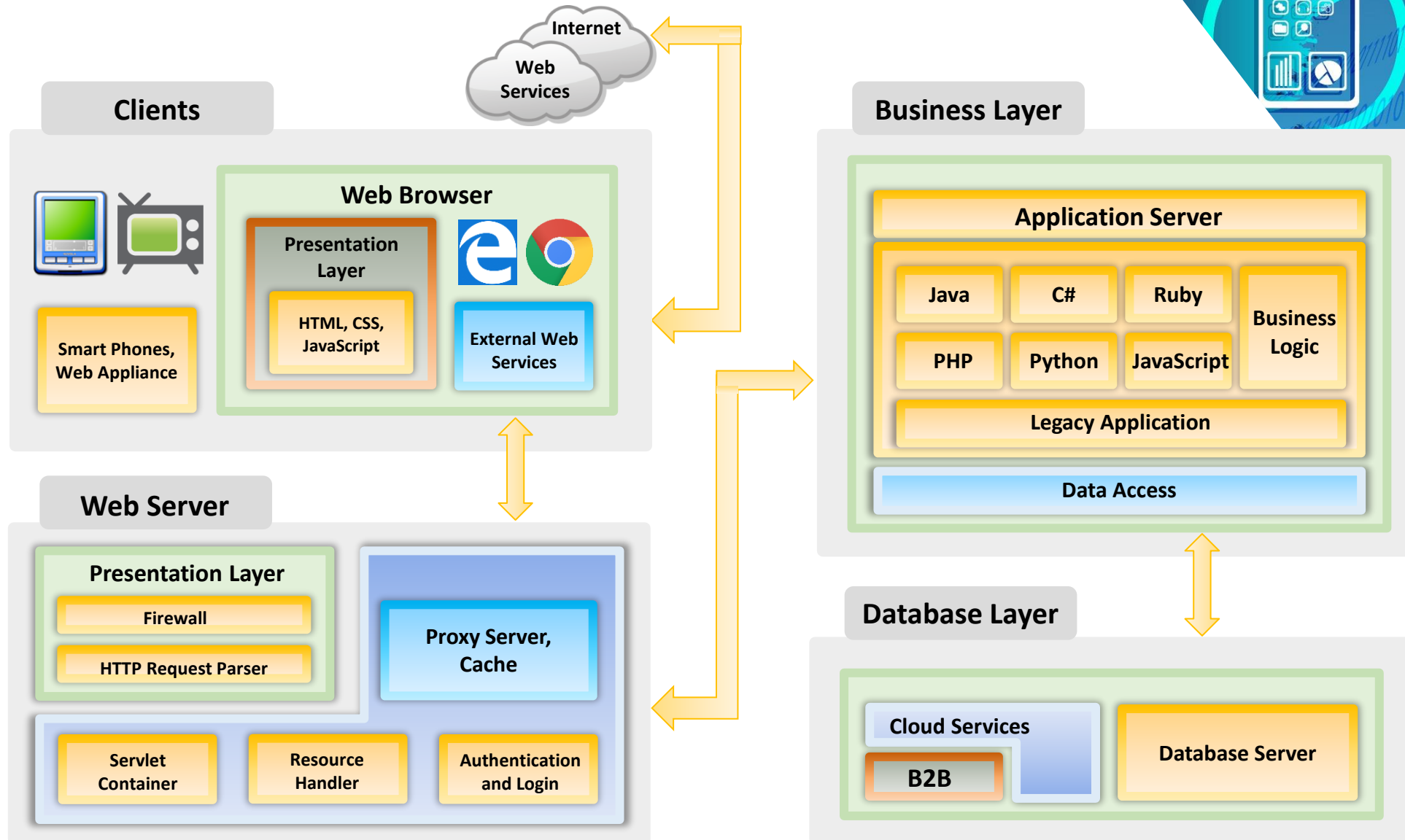
- ❑ Web applications provide an **interface between end users and web servers** through a set of web pages that are generated at the server end or contain script code to be executed dynamically within the client web browser
- ❑ Though web applications enforce certain **security policies**, they are vulnerable to various attacks such as SQL injection, cross-site scripting, and session hijacking



How Web Applications Work



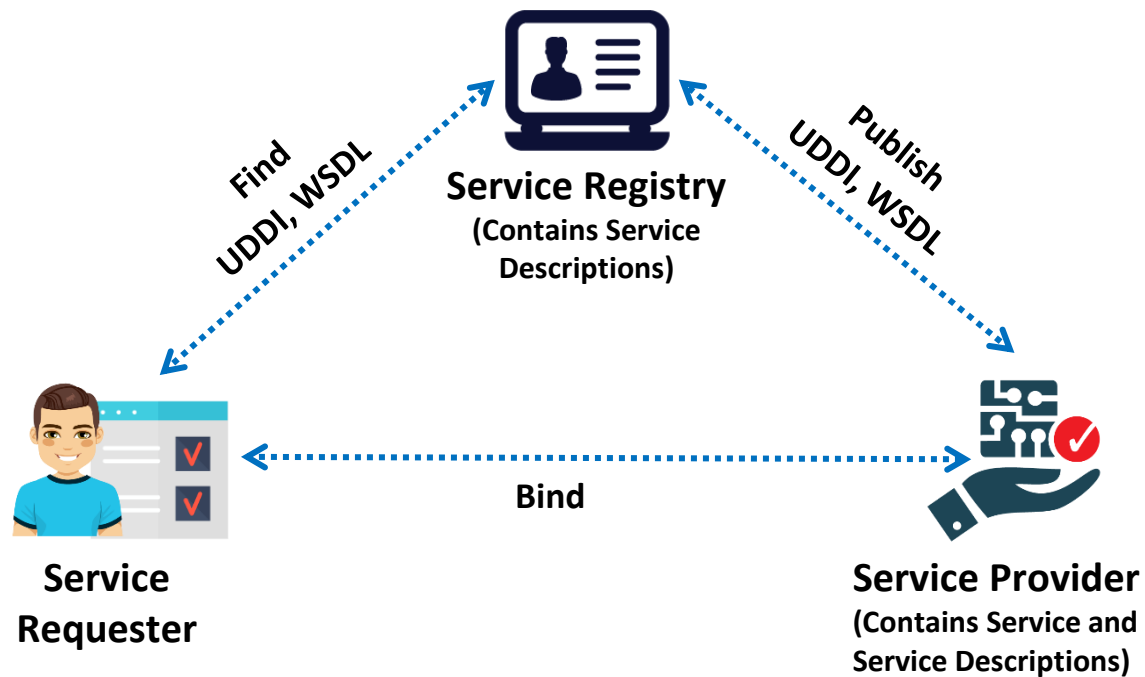
Web Application Architecture



Web Services

- ❑ A web service is an application or software that is deployed over the Internet and uses standard messaging protocols such as **SOAP**, **UDDI**, **WSDL**, and **REST to enable communication** between applications developed for different platforms

Web Service Architecture



Web Applications vs Web Services

OpenWeather

Search city Search

1:51pm, Mar 27
Zurich, CH

10°C

Feels like 3°C. Scattered clouds. Moderate breeze
6.2m/s WNW 1027hPa Humidity: 50%
UV: 3 Dew point: 0°C Visibility: 10.0km

No precipitation within an hour

Minute forecast

now	15min	30min	45min	60min
1:51pm	2:06pm	2:21pm	2:36pm	2:51pm
0mm/h				

Hourly forecast

now	2pm	3pm	4pm	5pm	6pm	7pm	8pm	9pm
10°C	9°C	9°C	10°C	9°C	8°C	5°C	4°C	4°C
scattered clouds	broken clouds	overcast clouds	overcast clouds	overcast clouds	overcast clouds	overcast clouds	broken clouds	scattered clouds
3.5m/s	3.5m/s	3.4m/s	3.2m/s	2.9m/s	2.2m/s	0.8m/s	1.0m/s	1.0m/s

8-day forecast

Sat, Mar 27	10 / 3°C	moderate rain
Sun, Mar 28	14 / 0°C	overcast clouds
Mon, Mar 29	19 / 3°C	scattered clouds
Tue, Mar 30	19 / 4°C	broken clouds
Wed, Mar 31	21 / 7°C	broken clouds
Thu, Apr 01	19 / 8°C	broken clouds
Fri, Apr 02	8 / 3°C	rain and snow
Sat, Apr 03	5 / 0°C	rain and snow

api.openweathermap.org/data/2.5/weather?lang=en&id=2

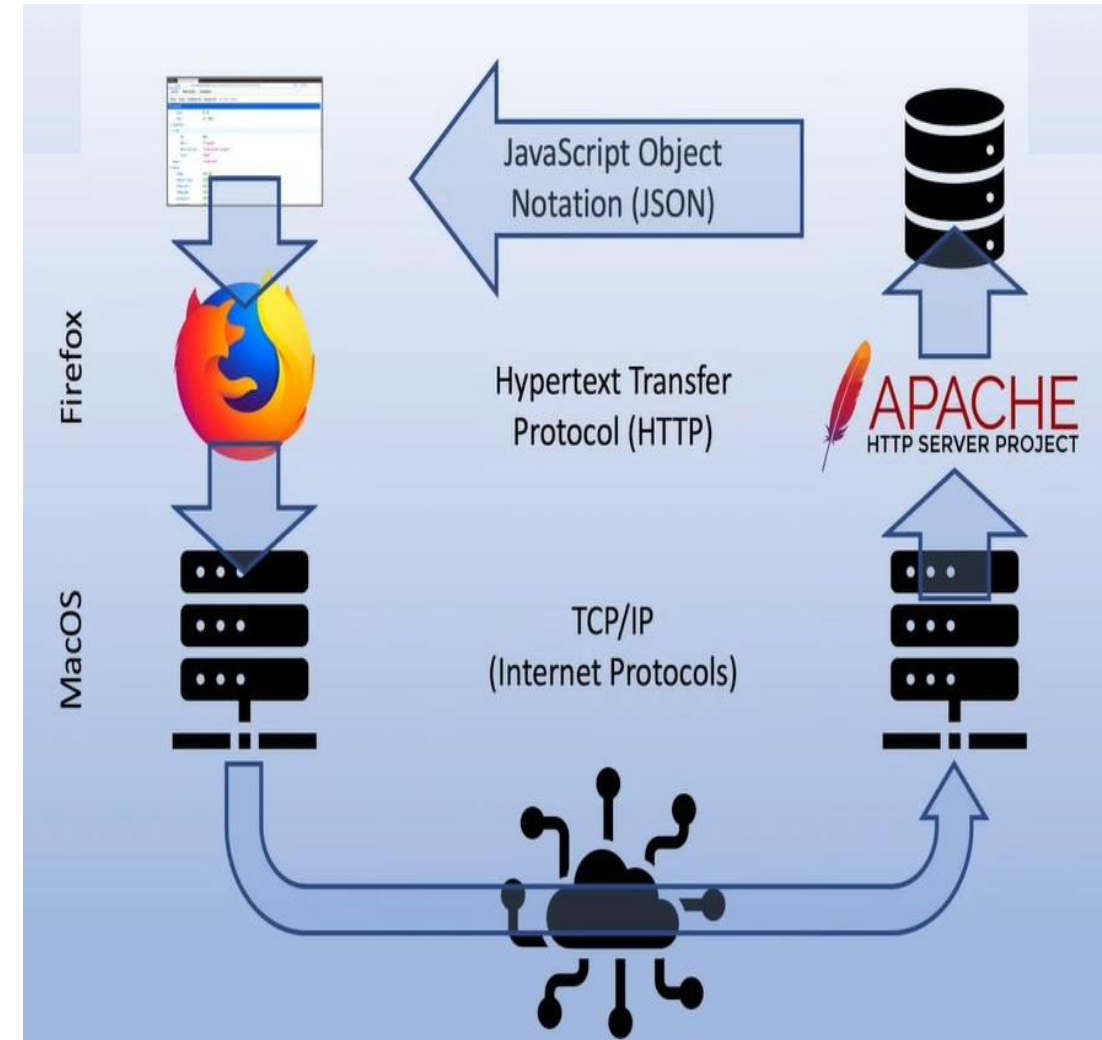
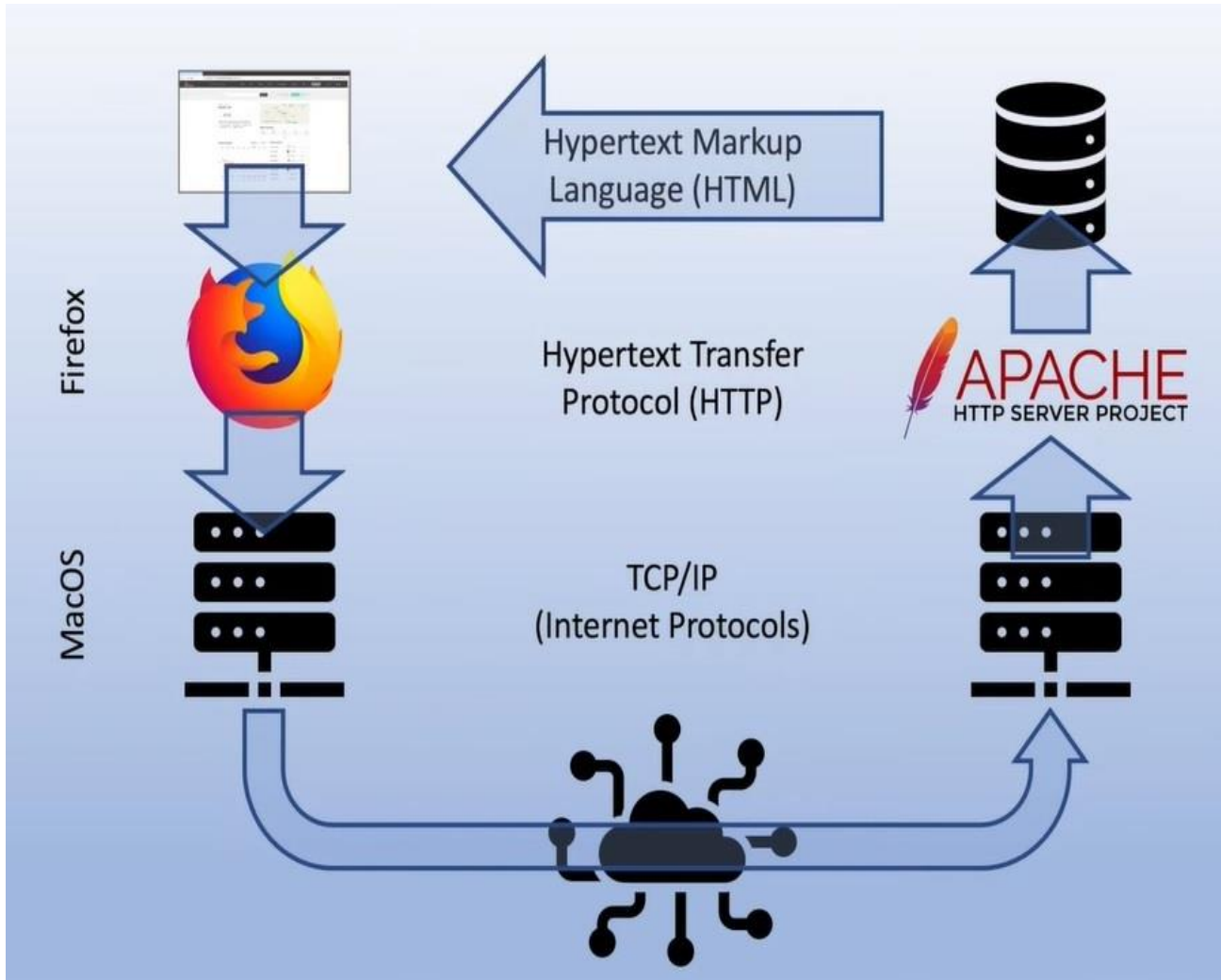
JSON Raw Data Headers

Save Copy Collapse All Expand All Filter JSON

```
▼ coord:  
  lon: 8.55  
  lat: 47.3667  
▼ weather:  
  ▼ 0:  
    id: 802  
    main: "Clouds"  
    description: "scattered clouds"  
    icon: "03d"  
  base: "stations"  
▼ main:  
  temp: 282.69  
  feels_like: 276.33  
  temp_min: 281.48  
  temp_max: 283.71  
  pressure: 1027
```

Getting APIs to Work

Web Applications vs Web Services



Types of Web Services

SOAP web services

It is based on the **XML format** and is used to transfer data between a service provider and requestor

RESTful web services

It is based on a **set of constraints** using underlying HTTP concepts to improve performance

Vulnerability Stack

Custom Web Applications



Layer 7



Business Logic Flaws
Technical Vulnerabilities

Third-party Components



Layer 6



Open Source / Commercial

Web Server



Layer 5



Apache / Microsoft IIS

Database



Layer 4



Oracle / MySQL / MS SQL

Operating System



Layer 3



Windows / Linux / macOS

Network



Layer 2



Router / Switch

Security

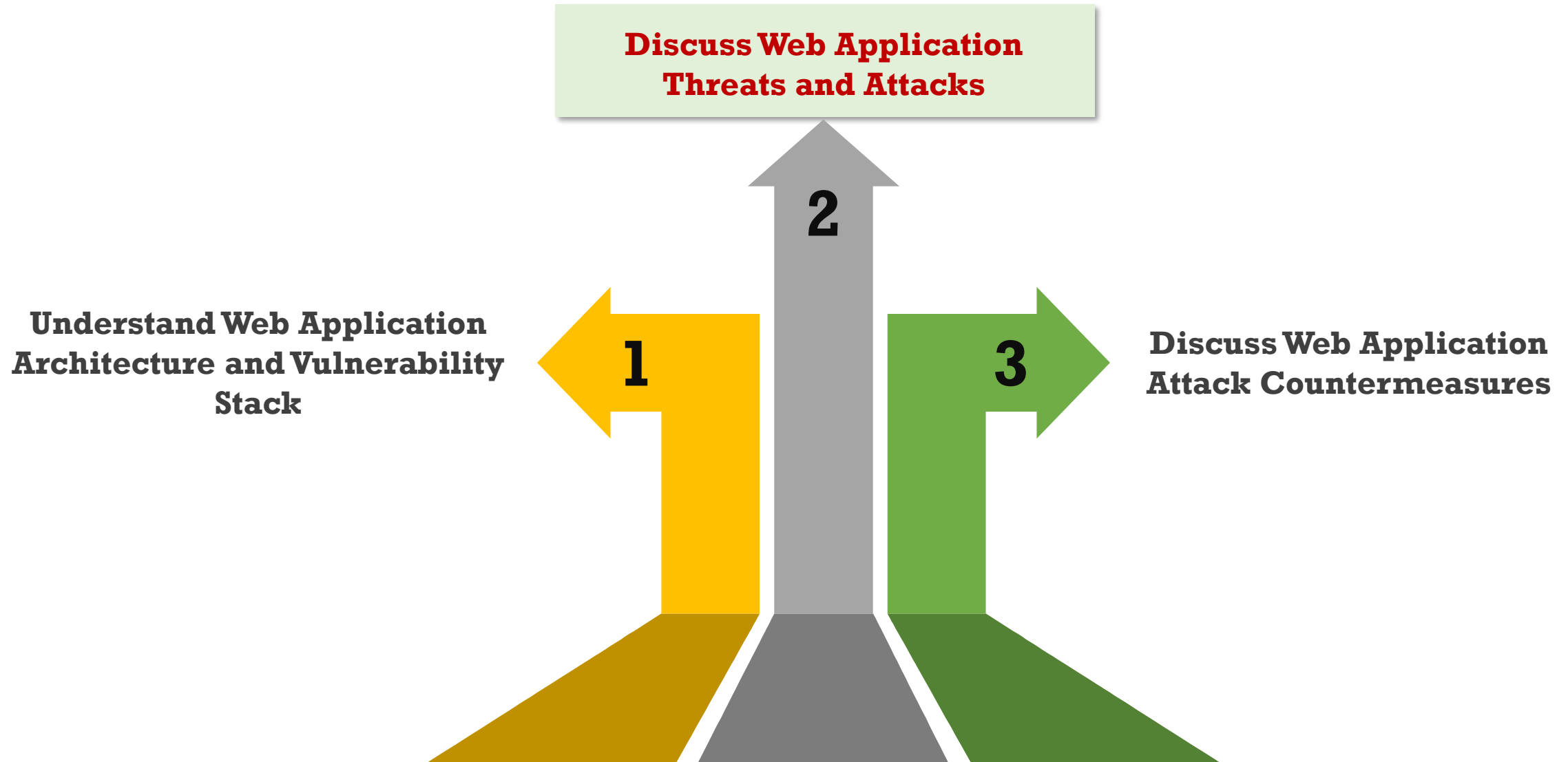


Layer 1



IPS / IDS

Module Flow



OWASP Top 10 Application Security Risks - 2017



Injection



**Broken
Authentication**



**Sensitive Data
Exposure**



**XML External
Entity (XXE)**



**Broken Access
Control**



**Security
Misconfiguration**



**Cross-Site
Scripting (XSS)**



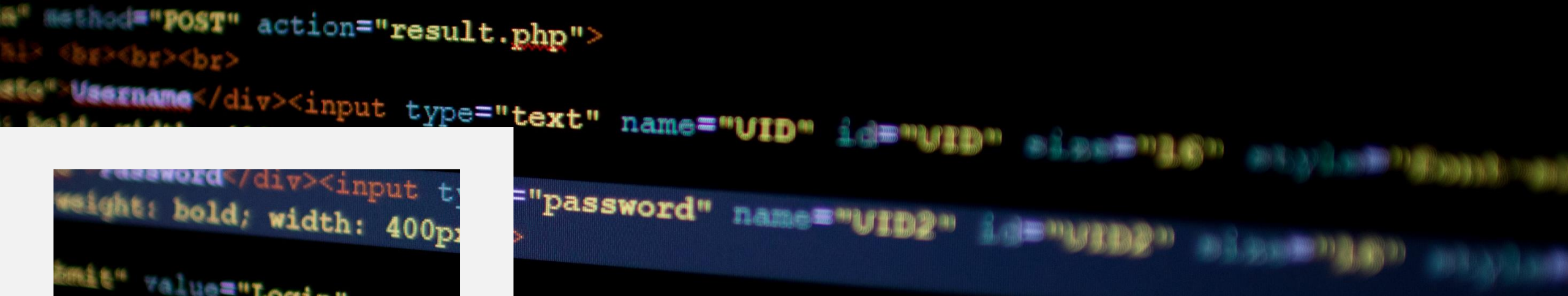
**Insecure
Deserialization**



**Using Components
with Known
Vulnerabilities**



**Insufficient Logging
and Monitoring**



A1 - Injection Flaws

- ❑ Injection flaws are web application vulnerabilities that allow **untrusted data** to be interpreted and executed as part of a command or query
- ❑ Attackers exploit injection flaws by **constructing malicious commands or queries** that result in data loss or corruption, lack of accountability, or denial of access

SQL Injection

It involves the injection of malicious SQL queries into user input forms

Command Injection

It involves the injection of malicious code through a web application

LDAP Injection

It involves the injection of malicious LDAP statements

A2 - Broken Authentication



- ❑ Attackers can exploit vulnerabilities in **authentication** or **session management functions** such as exposed accounts, session IDs, logout, password management, timeouts, etc. to impersonate users



Session ID in URLs

<http://www.certifiedhackershop.com/sale/saleitems=304;jsessionid=12OMTOIDPXM0OQSABGCKLHCJUN2JV?dest=NewMexico>

- Attackers **sniff the network traffic** or trick users to get session IDs and then reuse those session IDs for malicious purposes



Password Exploitation

- Attackers can gain access to a **web application's password database**. If user passwords are not encrypted, an attacker can exploit any user's password



Timeout Exploitation

- If an application's timeouts are not set properly and a user closes their browser without logging out from sites accessed through a public computer, an attacker can use the same browser later and **exploit that user's privileges**

A3 - Sensitive Data Exposure

- ❑ Sensitive data exposure occurs due to flaws like insecure cryptographic storage and information leakage
- ❑ When an **application uses poorly written encryption code** to securely encrypt and store sensitive data in the database, an attacker can exploit this flaw and **steal or modify weakly protected sensitive data** such as credit cards numbers, SSNs, and other authentication credentials

Vulnerable Code

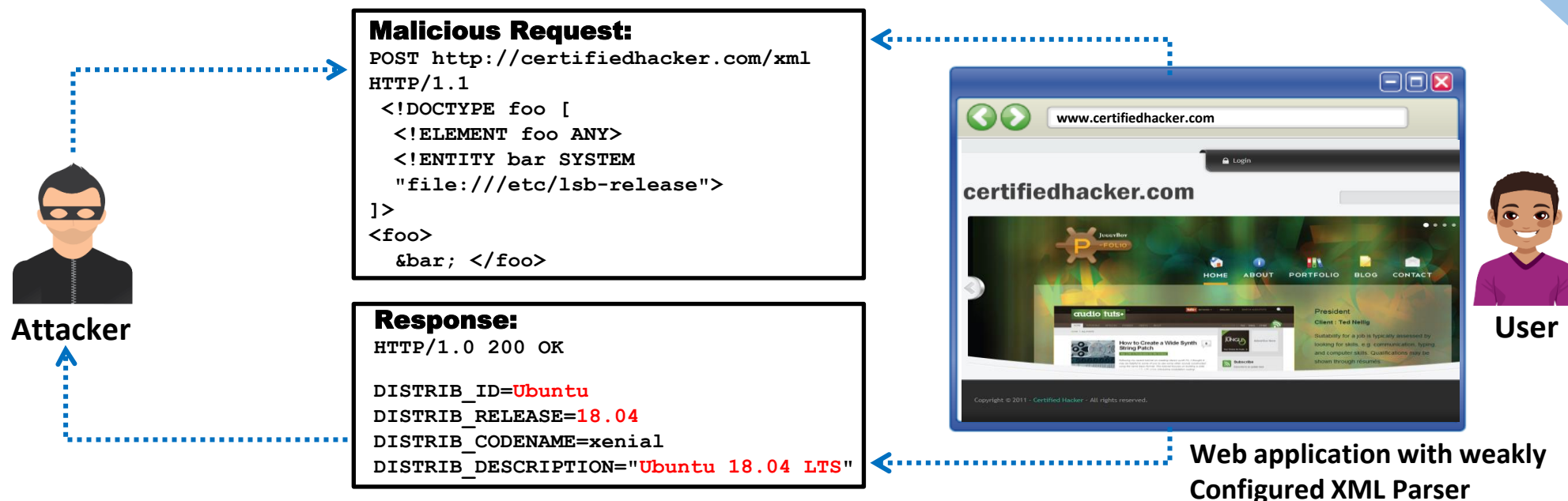
```
public String encrypt(String plainText) {  
    plainText = plainText.replace("a","z");  
    plainText = plainText.replace("b","y");  
    -----  
    return Base64Encoder.encode(plainText); }
```

Secure Code

```
private static String sKey = "zooooooooooom!!!!";  
private static String salt = "ooohhhhhhhhhhh!!!!";  
  
public static String encrypt(String plainText) {  
    byte[] iv = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };  
    IvParameterSpec ivspec = new IvParameterSpec(iv);  
  
    SecretKeyFactory factory = new  
        SecretKeyFactory.getInstance("PBKDF2WithHmacSHA256");  
  
    KeySpec = new PBEKeySpec(sKey.toCharArray(), salt.getBytes(),  
        65536, 256);  
  
    SecretKey key = factory.generateSecret(keySpec);  
  
    SecretKeySpec secretKey = new SecretKeySpec(key.getEncoded(),  
        "AES");  
  
    Cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");  
    cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivspec);  
  
    byte[] utf8text = plainText.getBytes("UTF-8");  
    byte[] encryptedText = cipher.doFinal(utf8text);  
  
    return Base64Encoder.encodeToString(encryptedText); }
```

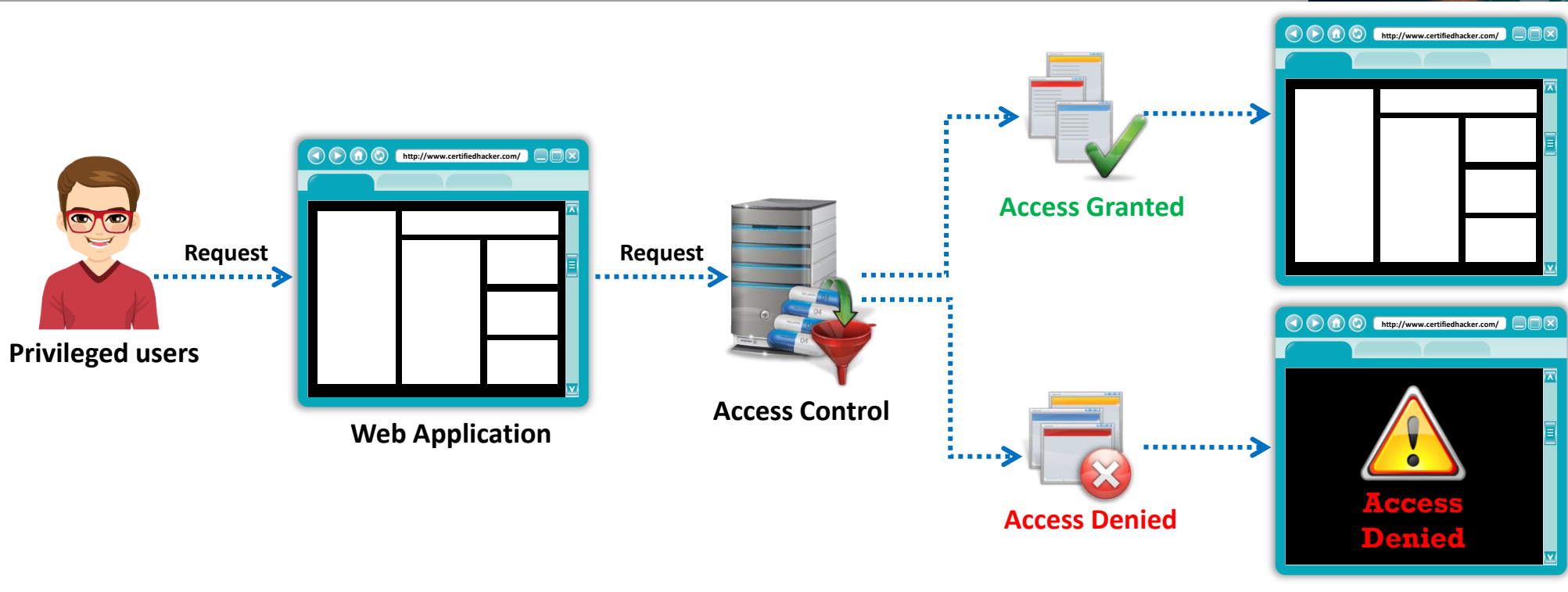
A4 - XML External Entity (XXE)

- ❑ XML External Entity attack is a server-side request forgery (SSRF) attack that can occur when a misconfigured XML parser allows **applications to parse XML input** from an unreliable source
- ❑ Attackers can refer a victim's web application to an external entity by including the reference in the **malicious XML input**
- ❑ When this malicious input is processed by the weakly configured XML parser of a target web application, it enables the attacker to **access protected files and services** from servers or connected networks



A5 - Broken Access Control

- ❑ Broken access control is a method in which an attacker identifies a flaw related to access control and bypasses the authentication, which allows them to compromise the network
- ❑ It allows an attacker to **act as users or administrators** with privileged functions and create, access, update or delete **every record**



A6 - Security Misconfiguration



Unvalidated Inputs

It refers to a web application vulnerability where input from a **client is not validated** before being processed by web applications and backend servers



Parameter/Form Tampering

It involves the **manipulation of parameters** exchanged between client and server to modify application data



Improper Error Handling

It gives **insight into source code** such as logic flaws, and default accounts. Using the information received from an error message, an attacker identifies vulnerabilities to launch various web application attacks



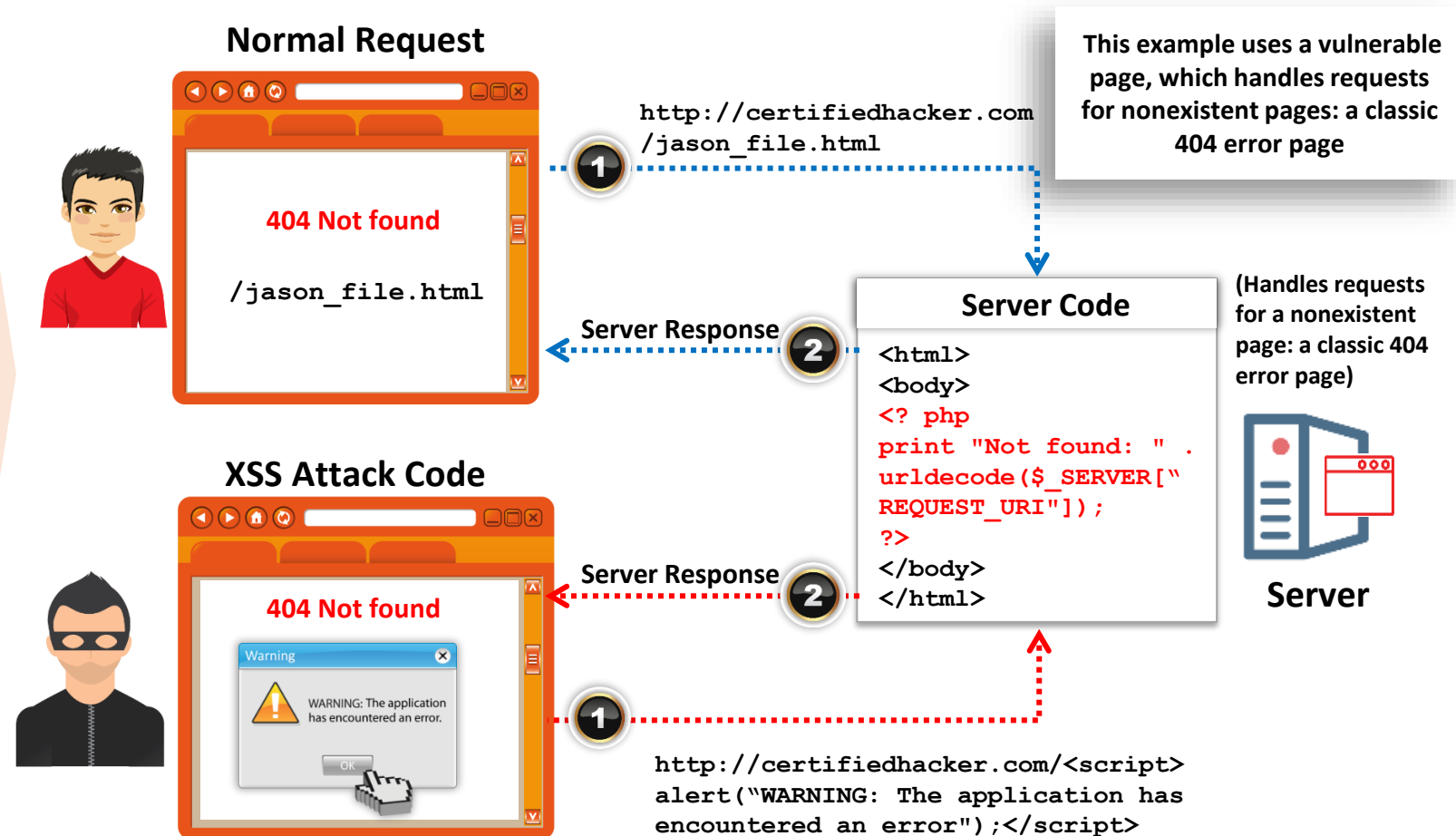
Insufficient Transport Layer Protection

It **supports weak algorithms** and uses expired or invalid certificates. Using insufficient transport layer protection exposes user data to untrusted third parties and can lead to account theft

A7 - Cross-Site Scripting (XSS) Attacks

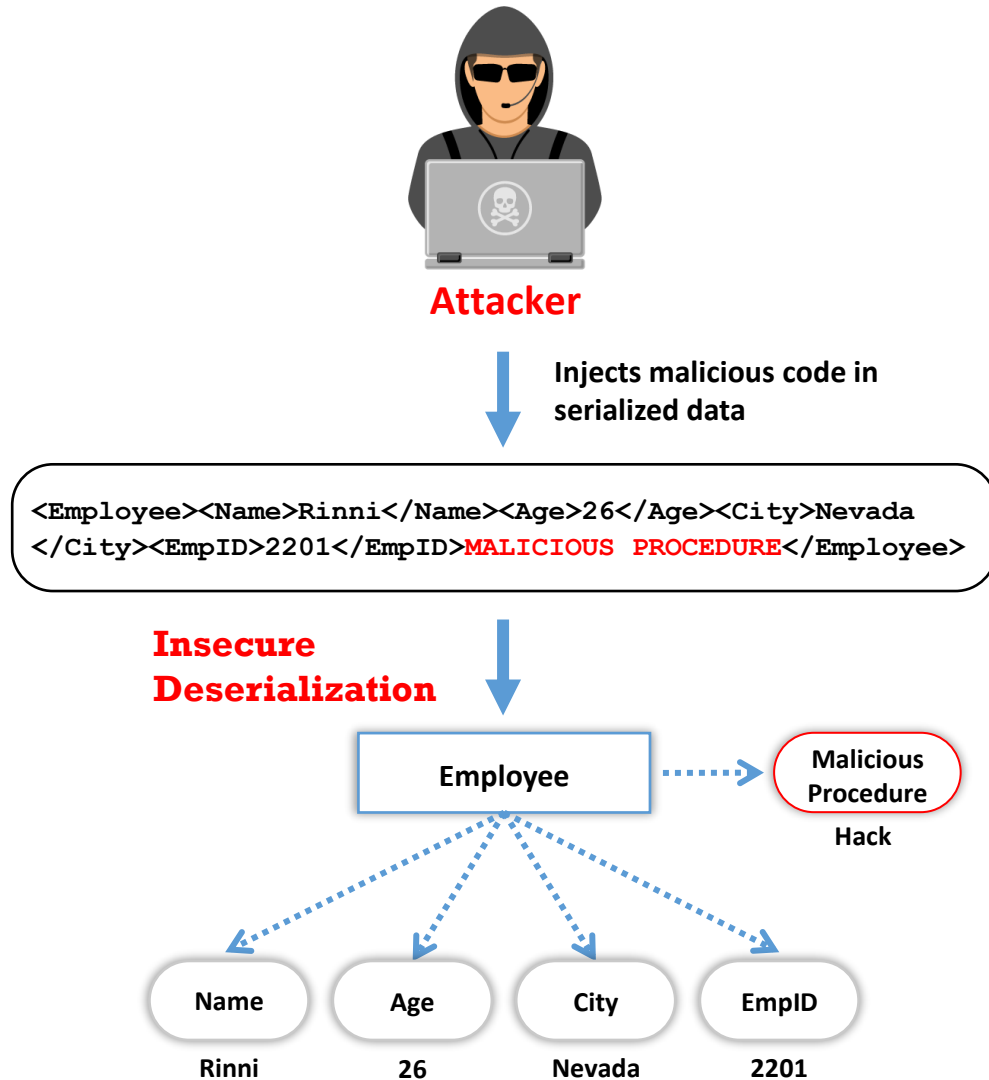
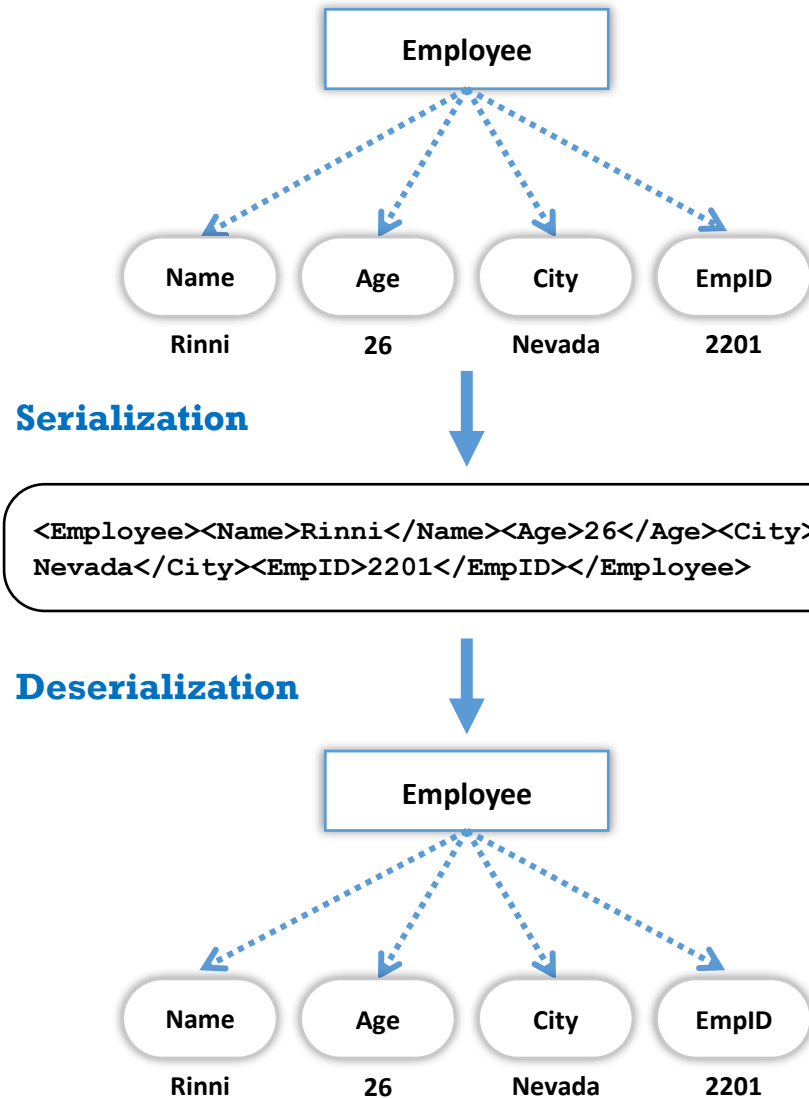
- ❑ Cross-site scripting ('XSS' or 'CSS') attacks **exploit vulnerabilities in dynamically generated web pages**, enabling malicious attackers to inject client-side scripts into web pages viewed by other users
- ❑ It occurs when **unvalidated input data** is included in dynamic content that is sent to a user's web browser for rendering

How XSS Attacks Work



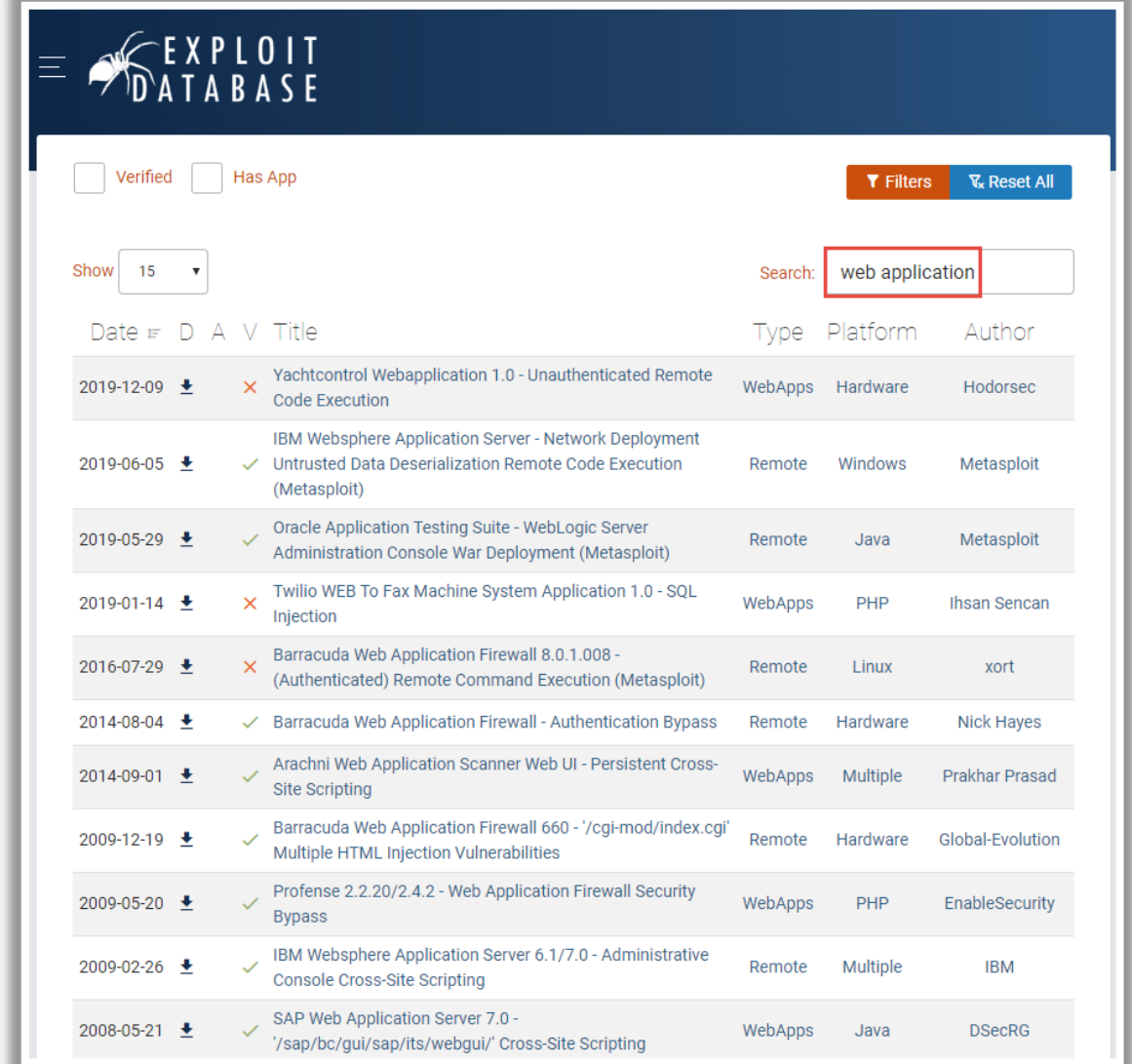
A8 - Insecure Deserialization

- ❑ Data serialization and deserialization is an effective process of **linearizing** and **de-linearizing data objects** for transmission to other networks or systems
- ❑ Attackers **inject malicious code** into **serialized data** and forward the malicious serialized data to the victim
- ❑ Insecure deserialization deserializes the malicious serialized content **along with the injected malicious code**, compromising the system or network



A9 - Using Components with Known Vulnerabilities

- ❑ Most web applications that use components such as **libraries** and **frameworks** always **execute them with full privileges**, and flaws in any component can result in serious impact
- ❑ Attackers can **identify weak components** or dependencies **by scanning** or by performing manual analysis
- ❑ Attackers search for any vulnerabilities on exploit sites such as **Exploit Database** (<https://www.exploit-db.com>), and **SecurityFocus** (<https://www.securityfocus.com>)
- ❑ If a vulnerable component is identified, the attacker customizes the exploit as required and execute the attack



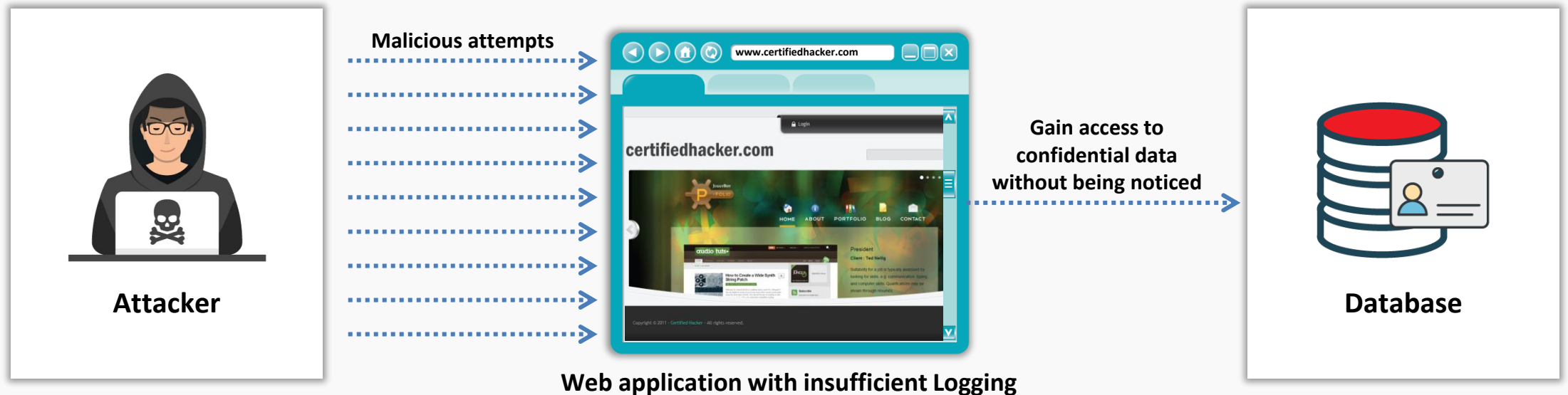
The screenshot shows the Exploit Database website interface. At the top, there's a navigation bar with the 'EXPLOIT DATABASE' logo. Below it, there are filters for 'Verified' and 'Has App', a 'Filters' button, and a 'Reset All' button. A search bar contains the text 'web application'. Below the search bar, there's a table of search results. The table has columns for 'Date', 'D', 'A', 'V', 'Title', 'Type', 'Platform', and 'Author'. The results list various vulnerabilities, including 'Yachtcontrol Webapplication 1.0 - Unauthenticated Remote Code Execution', 'IBM Websphere Application Server - Network Deployment Untrusted Data Deserialization Remote Code Execution (Metasploit)', 'Oracle Application Testing Suite - WebLogic Server Administration Console War Deployment (Metasploit)', 'Twilio WEB To Fax Machine System Application 1.0 - SQL Injection', 'Barracuda Web Application Firewall 8.0.1.008 - (Authenticated) Remote Command Execution (Metasploit)', 'Barracuda Web Application Firewall - Authentication Bypass', 'Arachni Web Application Scanner Web UI - Persistent Cross-Site Scripting', 'Barracuda Web Application Firewall 660 - '/cgi-mod/index.cgi' Multiple HTML Injection Vulnerabilities', 'Profense 2.2.20/2.4.2 - Web Application Firewall Security Bypass', 'IBM Websphere Application Server 6.1/7.0 - Administrative Console Cross-Site Scripting', and 'SAP Web Application Server 7.0 - '/sap/bc/gui/sap/its/webgui/' Cross-Site Scripting'.

Date	D	A	V	Title	Type	Platform	Author
2019-12-09				Yachtcontrol Webapplication 1.0 - Unauthenticated Remote Code Execution	WebApps	Hardware	Hodorsec
2019-06-05				IBM Websphere Application Server - Network Deployment Untrusted Data Deserialization Remote Code Execution (Metasploit)	Remote	Windows	Metasploit
2019-05-29				Oracle Application Testing Suite - WebLogic Server Administration Console War Deployment (Metasploit)	Remote	Java	Metasploit
2019-01-14				Twilio WEB To Fax Machine System Application 1.0 - SQL Injection	WebApps	PHP	Ihsan Sencan
2016-07-29				Barracuda Web Application Firewall 8.0.1.008 - (Authenticated) Remote Command Execution (Metasploit)	Remote	Linux	xort
2014-08-04				Barracuda Web Application Firewall - Authentication Bypass	Remote	Hardware	Nick Hayes
2014-09-01				Arachni Web Application Scanner Web UI - Persistent Cross-Site Scripting	WebApps	Multiple	Prakhar Prasad
2009-12-19				Barracuda Web Application Firewall 660 - '/cgi-mod/index.cgi' Multiple HTML Injection Vulnerabilities	Remote	Hardware	Global-Evolution
2009-05-20				Profense 2.2.20/2.4.2 - Web Application Firewall Security Bypass	WebApps	PHP	EnableSecurity
2009-02-26				IBM Websphere Application Server 6.1/7.0 - Administrative Console Cross-Site Scripting	Remote	Multiple	IBM
2008-05-21				SAP Web Application Server 7.0 - '/sap/bc/gui/sap/its/webgui/' Cross-Site Scripting	WebApps	Java	DSecRG

<https://www.exploit-db.com>

A10 - Insufficient Logging and Monitoring

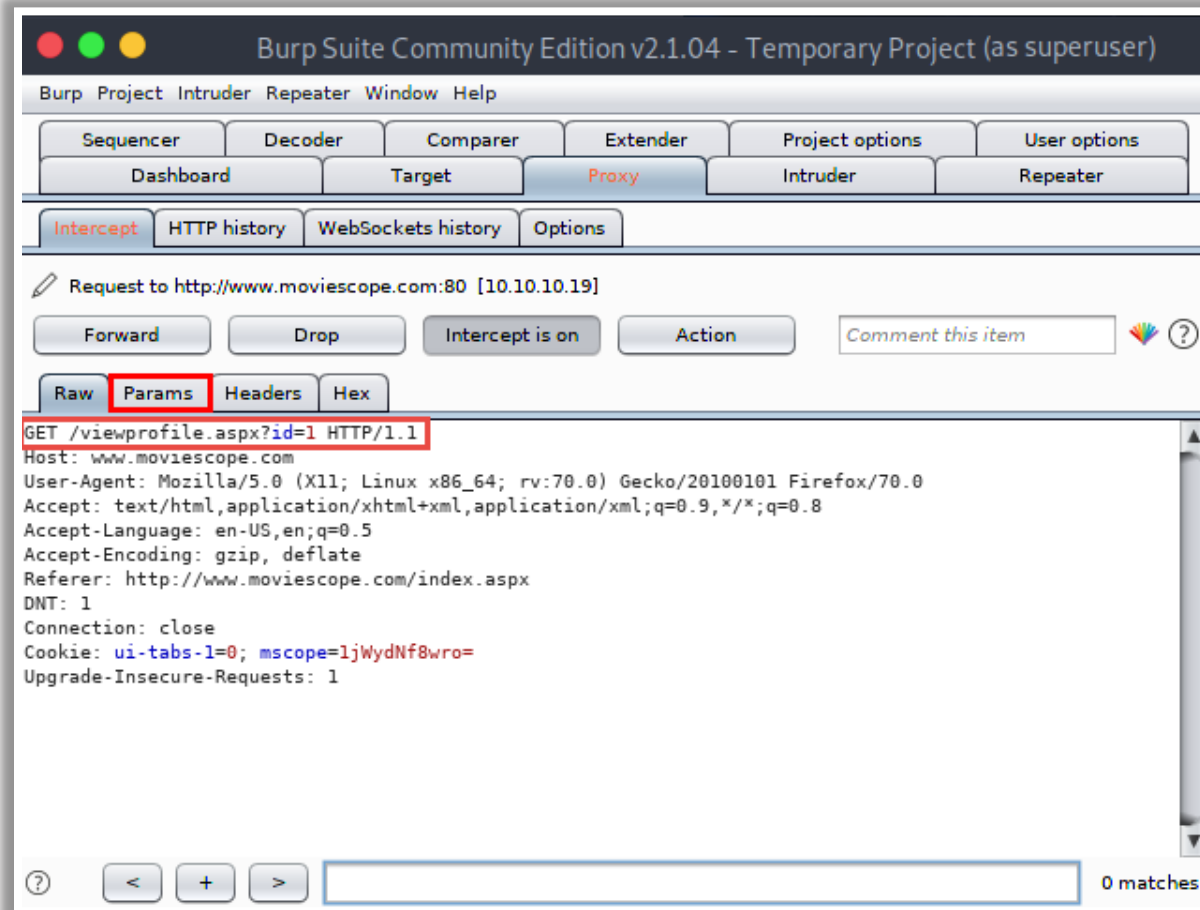
- ❑ Web applications maintain logs to track usage patterns, such as **user login credentials** and **admin login credentials**
- ❑ Insufficient logging and monitoring refer to the scenario where the detection software either does not **record the malicious event** or ignores important details about the event
- ❑ Attackers usually inject, delete, or tamper the web application logs to engage in **malicious activities** or **hide their identities**



Web Application Attack Tools

Burp Suite

Support the entire web application testing process, from initial mapping and analysis of an application's attack surface to finding and **exploiting security vulnerabilities**



<https://portswigger.net>



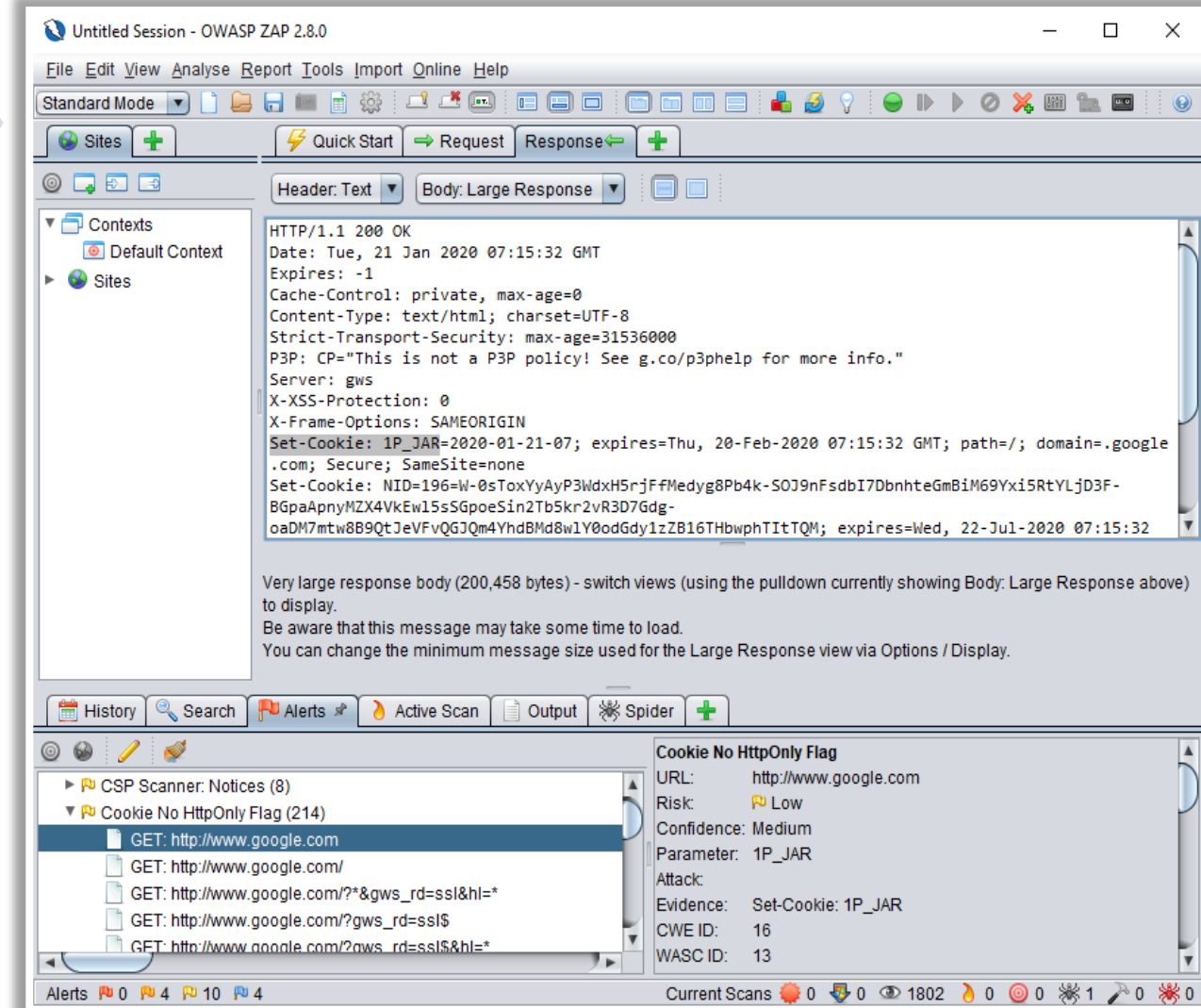
Web Application Attack Tools (Cont'd)

OWASP Zed Attack Proxy

Provides automated scanners and tools that allow you to find security vulnerabilities manually

Web Application Attack Tools

- Metasploit (<https://www.metasploit.com>)
- w3af (<http://w3af.org>)
- Nikto (<https://cirt.net>)
- Sn1per (<https://github.com>)
- WSSiP (<https://github.com>)

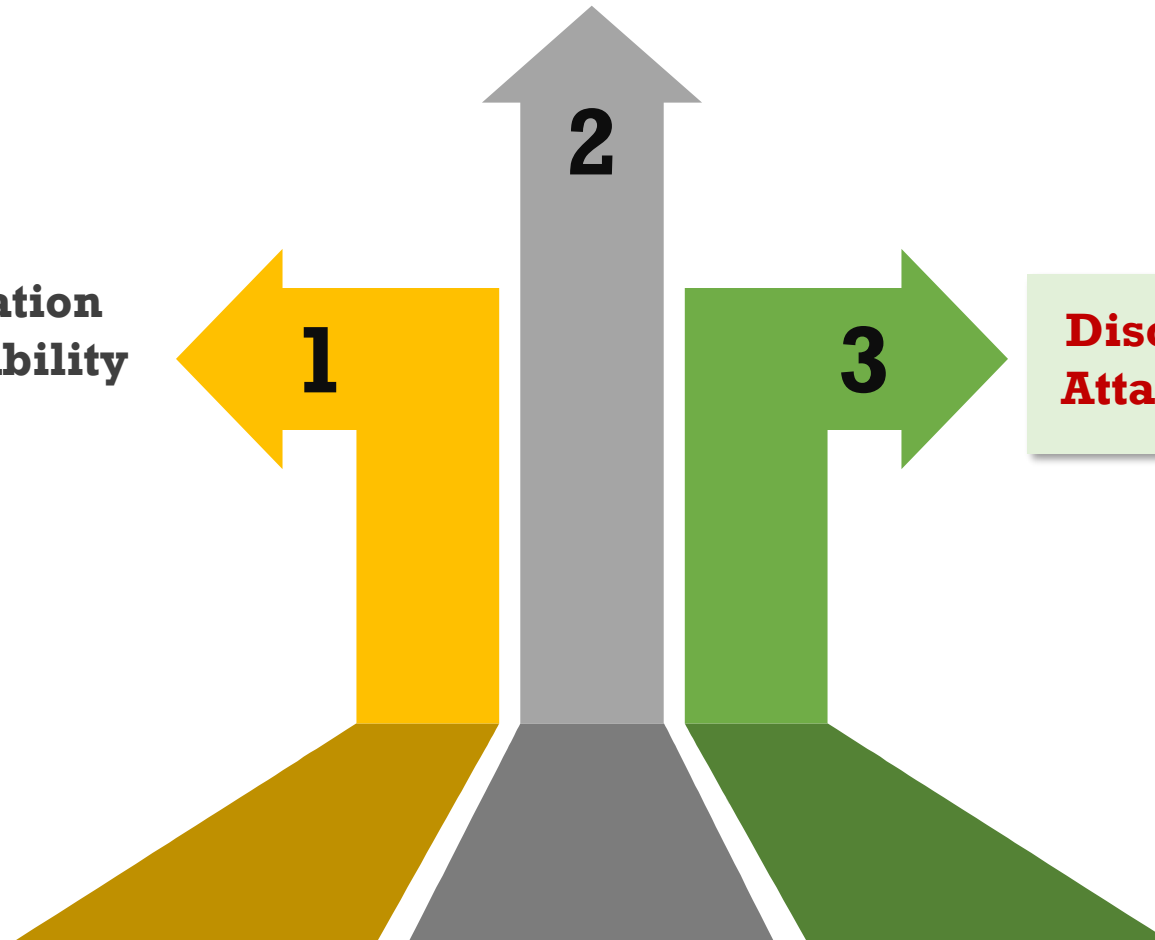


Module Flow

**Discuss Web Application
Threats and Attacks**

**Understand Web Application
Architecture and Vulnerability
Stack**

**Discuss Web Application
Attack Countermeasures**



Web Application Attack Countermeasures



SQL Injection Attacks

- ☐ Limit the **length** of user input
- ☐ Use custom **error messages**
- ☐ Monitor **DB traffic** using an IDS, WAF



Command Injection Flaws

- ☐ Perform **input validation**
- ☐ Escape **dangerous characters**
- ☐ Use **language-specific** libraries that avoid problems due to shell commands



LDAP Injection Attacks

- ☐ Perform type, pattern, and **domain value validation** on all input data
- ☐ Make the **LDAP filter** as specific as possible
- ☐ Validate and restrict the **amount of data returned** to the user

Web Application Attack Countermeasures (Cont'd)

Broken Authentication and Session Management

- Use **SSL** for authenticated parts of the application
- Verify whether all the users' identities and credentials are stored in a **hashed form**
- Never submit session data as part of a **GET, POST**

Sensitive Data Exposure

- Do not create or use **weak cryptographic algorithms**
- **Generate encryption keys** offline and store them securely
- Ensure that encrypted data stored on disk is not easy to **decrypt**

XML External Entity

- **Avoid processing XML input** containing reference to external entity by weakly configured XML parser
- **XML unmarshaller** should be configured securely
- **Parse the document** with a securely configured parser

Broken Access Control

- Perform **access control checks** before redirecting the authorized user to the requested resource
- Avoid using **insecure IDs** to prevent attackers guessing them
- Provide a session timeout mechanism

Web Application Attack Countermeasures (Cont'd)



Security Misconfiguration

- ☐ Configure all **security mechanisms** and disable all unused services
- ☐ Setup roles, permissions, and accounts and **disable all default accounts** or change their default passwords
- ☐ Scan for the **latest security vulnerabilities** and apply the latest security patches
- ☐ Non-SSL requests to web pages should be redirected to the **SSL page**



XSS Attacks

- ☐ Validate all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification
- ☐ Use testing tools extensively during the design phase to eliminate such XSS holes in the application
- ☐ Use a web application firewall to block the **execution of malicious scripts**
- ☐ Convert all **non-alphanumeric characters** to HTML character entities before displaying the user input in search engines

Web Application Attack Countermeasures (Cont'd)



Insecure Deserialization

- ❑ Validate untrusted input which is to be serialized to ensure serialized data contain only **trusted classes**
- ❑ Deserialization of trusted data must cross a trust boundary
- ❑ Developers must re-architect their applications



Using Components with Known Vulnerabilities

- ❑ Regularly check the versions of both client-side and server-side components and their dependencies
- ❑ Continuously monitor sources like the **national vulnerability database** (NVB) for vulnerabilities in your components
- ❑ Regularly apply security patches



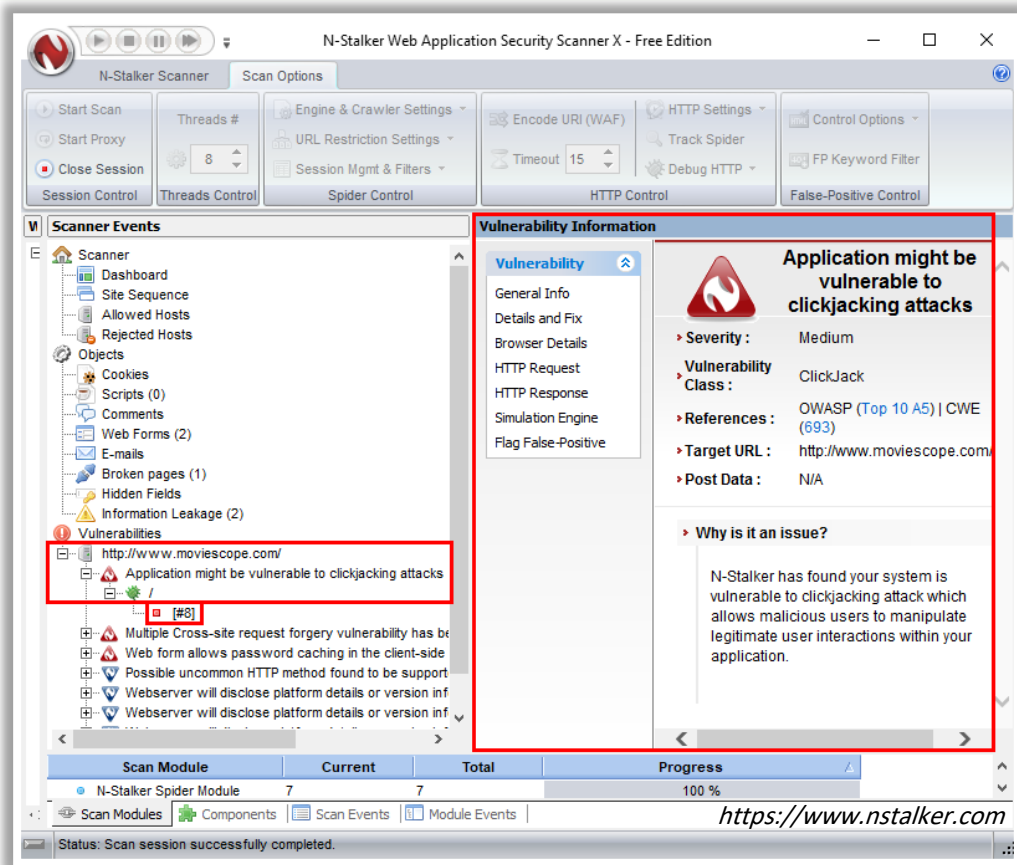
Insufficient Logging & Monitoring

- ❑ Define the scope of assets covered in **log monitoring** to include business critical areas
- ❑ Setup a minimum baseline for logging and ensure that it is followed for all assets
- ❑ Ensure that logs are logged with user context, so that the **logs are traceable** to specific users

Web Application Security Testing Tools

N-Stalker Web App Security Scanner

N-Stalker web app security scanner checks for vulnerabilities such as SQL injection, XSS, and other known attacks



Acunetix WVS

<https://www.acunetix.com>



Browser Exploitation Framework (BeEF)

<http://beefproject.com>



Metasploit

<https://www.metasploit.com>



PowerSploit

<https://github.com>

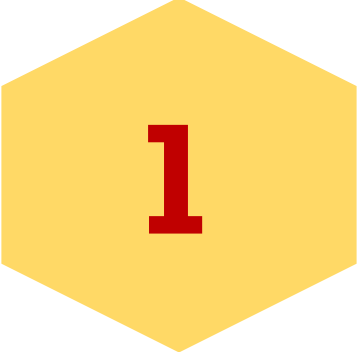


Watcher

<https://www.casaba.com>

SQL Injection Attacks

Module Flow



1

**Discuss Types of SQL
Injection Attacks**



2

**Discuss SQL Injection Attack
Countermeasures**

What is SQL Injection?

- ❑ SQL injection is a technique used to take advantage of **un-sanitized input vulnerabilities** to pass SQL commands through a web application for execution by a **backend database**
- ❑ It is a basic attack used to either **gain unauthorized access** to a database or **retrieve information** directly from the database



Why Bother about SQL Injection?



Based on the use of **applications** and the way they **process user supplied data**, SQL injections can be used to implement the following types of attacks:

1 Authentication Bypass

Compromised Data Integrity **4**

2 Authorization Bypass

Compromised Availability of Data **5**

3 Information Disclosure

Remote Code Execution **6**

SQL Injection and Server-side Technologies



Server-side Technology

Powerful server-side technologies like ASP.NET and database servers allow developers to **create dynamic, data-driven websites**, and **web apps** with incredible ease



Exploit

The power of ASP.NET and SQL can easily be **exploited by hackers** using SQL injection attacks



Susceptible Databases

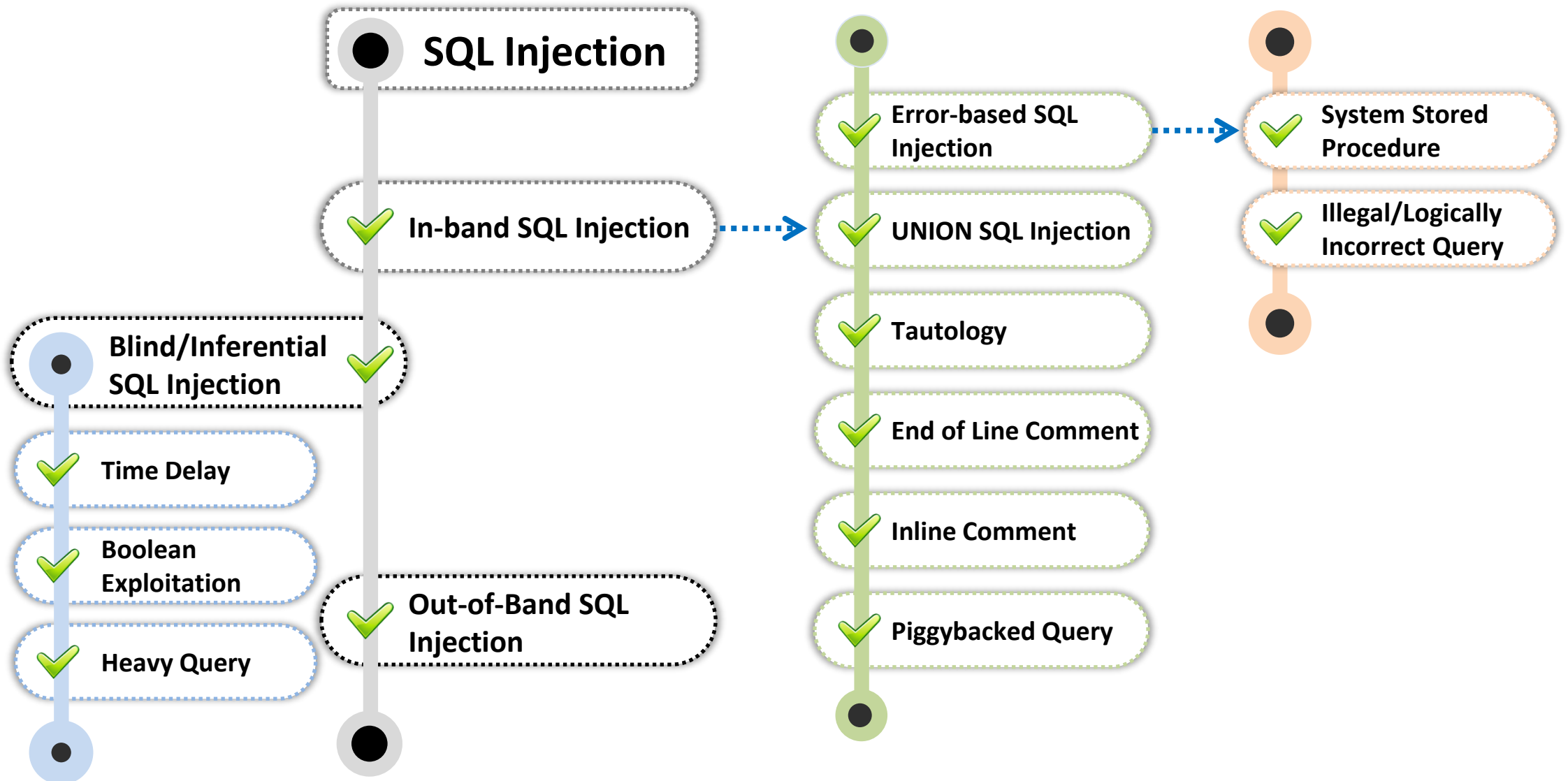
All relational databases, SQL Server, Oracle, IBM DB2, and MySQL, are susceptible to **SQL-injection attacks**



Attack

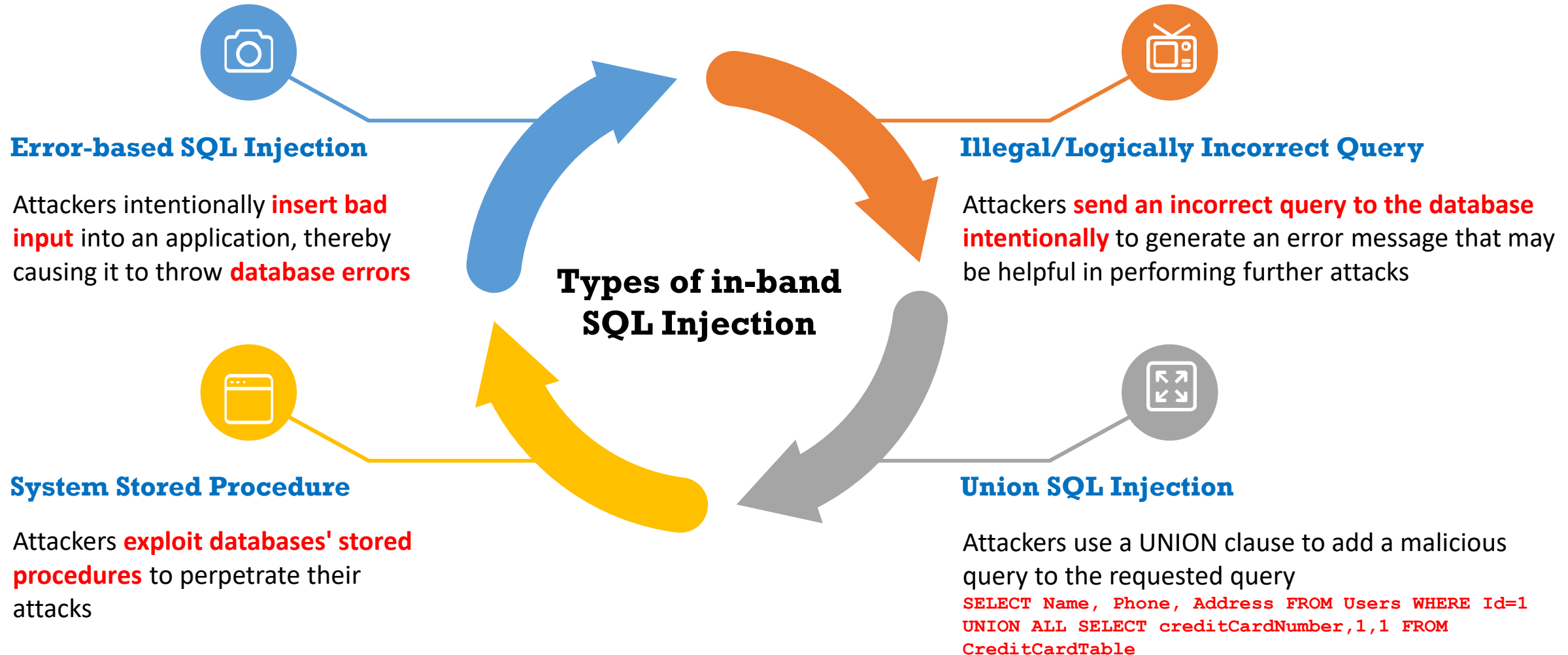
SQL injection attacks do not exploit a specific software vulnerability, instead they **target websites and web apps** that do not follow **secure coding practices** for accessing and manipulating data stored in a relational database

Types of SQL Injection



In-Band SQL Injection

- ❑ Attackers use the **same communication channel** to perform the attack and **retrieve** the results



In-Band SQL Injection (Cont'd)

Types of in-band SQL Injection

Tautology

Attackers inject statements that are always true so that the queries always return results after evaluating the WHERE condition

```
SELECT * FROM users WHERE name =  
    '' OR '1'='1';
```

End of Line Comment

After injecting the code into a specific field, legitimate code that follows is nullified using end of line comments

```
SELECT * FROM user WHERE name =  
    'x' AND userid IS NULL; --';
```

In-line Comments

Attackers integrate multiple vulnerable inputs into a single query using inline comments

```
INSERT INTO Users (UserName,  
isAdmin, Password)  
VALUES ('Attacker', 1, /*', 0,  
    '*/'mypwd')
```

Piggybacked Query

Attackers inject additional malicious query into the original query. Consequently, the DBMS executes multiple SQL queries

```
SELECT * FROM EMP WHERE EMP.EID =  
1001 AND EMP.ENAME = 'Bob'; DROP  
TABLE DEPT;
```



Error Based SQL Injection



- ❑ Error based SQL Injection **forces the database** to perform some operation in which the **result will be an error**
- ❑ This exploitation may differ depending on the DBMS

- ✓ Consider the SQL query shown below:

```
SELECT * FROM products WHERE  
id_product=$id_product
```

- ✓ Consider the following request to a script that executes the query above:

```
http://www.example.com/product.php?id=10
```

- ✓ The malicious request would be (e.g., Oracle 10g):

```
http://www.example.com/product.php?  
id=10||UTL_INADDR.GET_HOST_NAME( (SELECT user  
FROM DUAL) )-
```



Union SQL Injection



- ❑ This technique involves **joining a forged query** to the **original query**
- ❑ The result of a forged query will be joined to the result of the original query, thereby allowing it to obtain the **values of fields of other tables**

Example:

• `SELECT Name, Phone, Address FROM Users WHERE Id=$id`

Now set the following Id value:

• `$id=1 UNION ALL SELECT creditCardNumber,1,1 FROM CreditCardTable`

The final query is as shown below:

• `SELECT Name, Phone, Address FROM Users WHERE Id=1 UNION ALL SELECT creditCardNumber,1,1 FROM CreditCardTable`

The above query joins the result of the original query with all the credit card users

Blind/Inferential SQL Injection



No Error Message

Blind SQL Injection is used when a **web application is vulnerable** to an SQL injection, but the results of the injection are not visible to the attacker



Generic Page

Blind SQL injection is identical to a normal SQL Injection, except that a generic custom page is displayed when an attacker attempts to exploit an application rather than seeing a **useful error message**

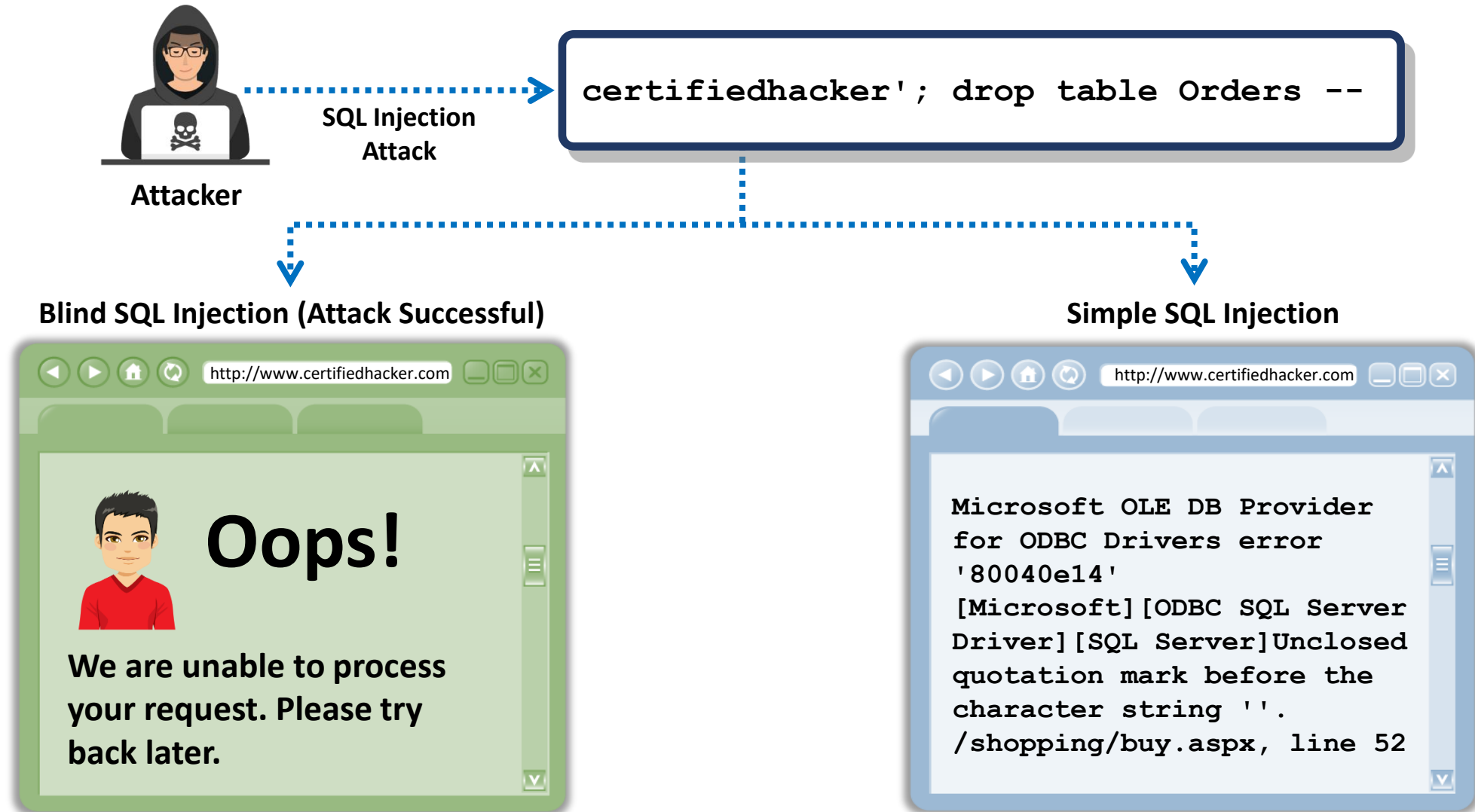


Time- intensive

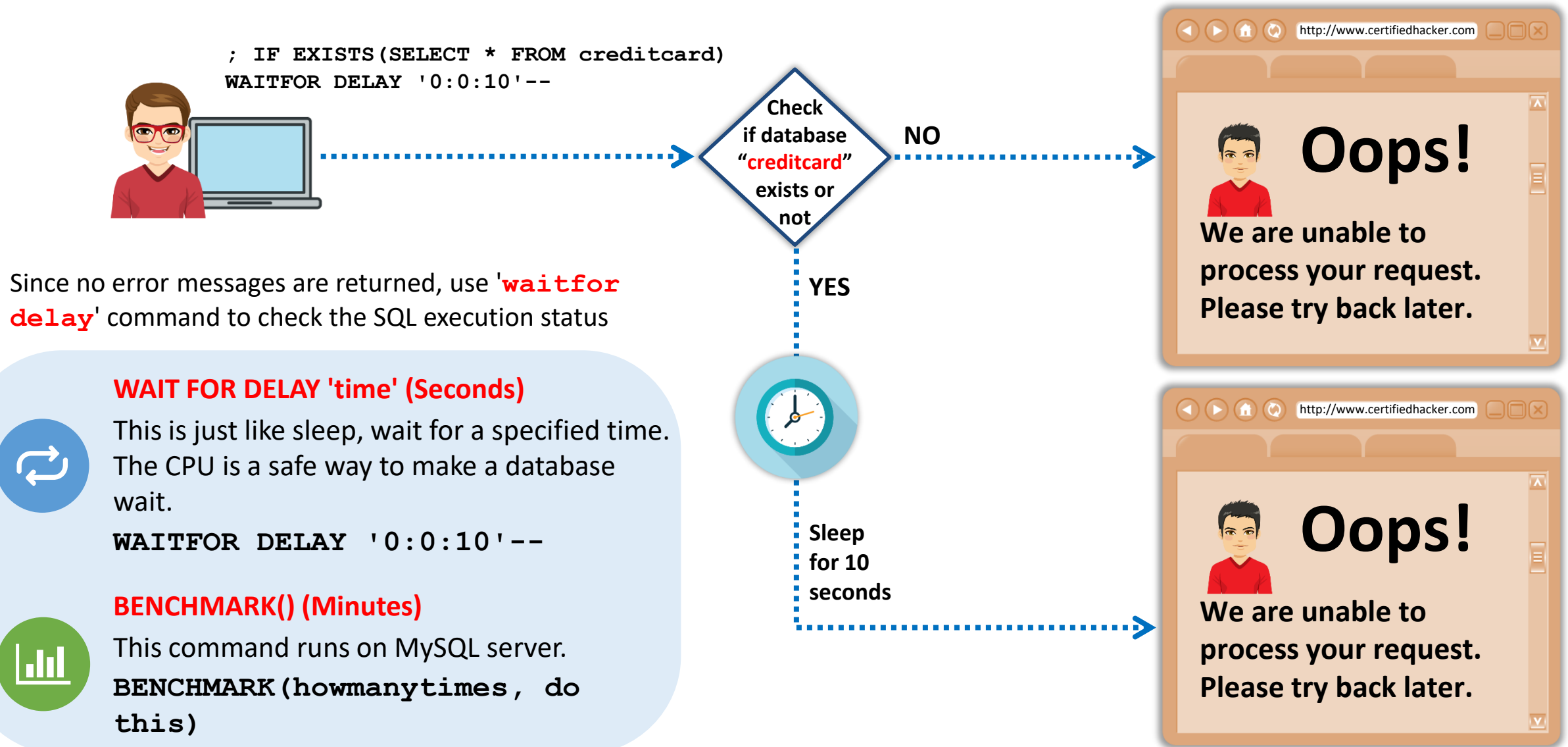
This type of attack can become **time-intensive because a new statement** must be crafted for each bit recovered

Note: An attacker can still steal data by asking a series of True and False questions through SQL statements

Blind SQL Injection: No Error Message Returned



Blind SQL Injection: WAITFOR DELAY (YES or NO Response)



Blind SQL Injection: Boolean Exploitation



Multiple valid statements that evaluate **true** and **false** are supplied in the affected parameter in the **HTTP request**



By comparing the response page between both conditions, the attackers can infer whether or not the **injection was successful**



For example, consider the following URL:

`http://www.myshop.com/item.aspx?id=67`

An attacker may manipulate the above request to

`http://www.myshop.com/item.aspx?id=67 and 1=2`

SQL Query Executed

```
SELECT Name, Price, Description FROM ITEM_DATA WHERE ITEM_ID = 67  
AND 1 = 2
```

Blind SQL Injection: Heavy Query

Attackers use heavy queries to perform a time delay SQL injection attack without using **time delay functions**

A heavy query retrieves a significant amount of data and takes a long time to execute in the **database engine**

Attackers generate heavy queries using **multiple joins on system tables**

For example,

```
SELECT * FROM products WHERE id=1 AND 1 <  
SELECT count(*) FROM all_users A,  
all_users B, all_users C
```



Out-of-Band SQL Injection

01

In Out-of-Band SQL injection, the attacker needs to **communicate with the server** and acquire features of the **database server** used by the web application

02

Attackers use different **communication channels** to perform the attack and obtain the results



03

Attackers use **DNS** and **HTTP requests** to retrieve data from the database server

04

For example, in a Microsoft SQL Server, an attacker exploits the **xp_dirtree** **command** to send DNS requests to a server controlled by the attacker

sqlmap

sqlmap

<http://sqlmap.org>



Blisqy

blind-sql-bitshifting

NoSQLMap

SQL Power Injector

<http://www.sqlpowerinjector.com>

Module Flow



1

**Discuss Types of SQL
Injection Attacks**



2

**Discuss SQL Injection Attack
Countermeasures**

SQL Injection Attack Countermeasures

1

Make no assumptions about the **size**, **type**, or **content** of the data that is received by your application

2

Test the **size** and **data type of input** and enforce appropriate limits to prevent buffer overruns

3

Test the content of **string variables** and accept only **expected values**

4

Reject entries that contain **binary data**, **escape sequences**, and **comment** characters

5

Never build **Transact-SQL** statements directly from user input and use stored procedures to validate user input

6

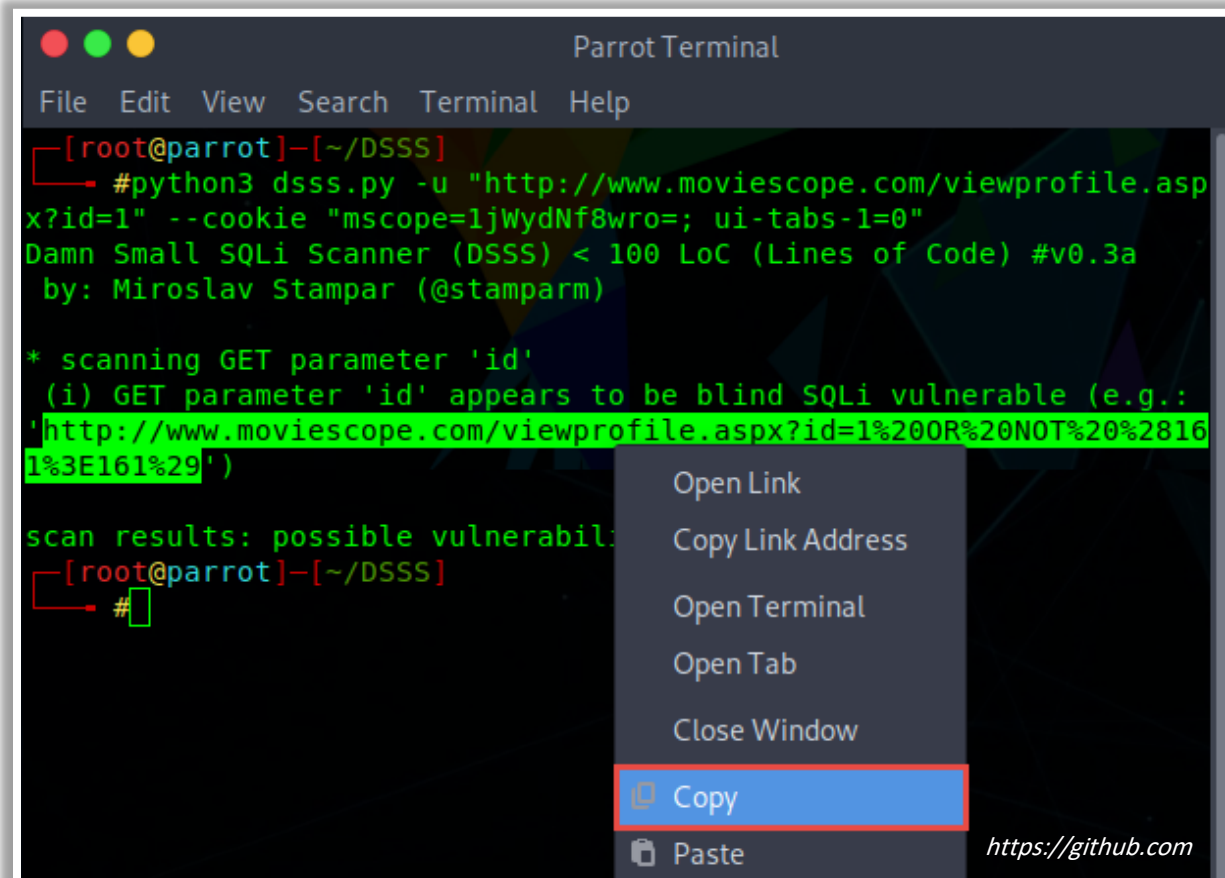
Implement **multiple layers of validation** and never concatenate user input that is not validated



SQL Injection Detection Tools

Damn Small SQLi Scanner (DSSS)

DSSS is an **SQL injection vulnerability scanner** that scans the web application for various SQL injection vulnerabilities



```
Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]--[~/DSSS]
#python3 dsss.py -u "http://www.moviescope.com/viewprofile.aspx?id=1" --cookie "mscope=1jWydNf8wro=; ui-tabs-1=0"
Damn Small SQLi Scanner (DSSS) < 100 LoC (Lines of Code) #v0.3a
by: Miroslav Stampar (@stamparm)

* scanning GET parameter 'id'
(i) GET parameter 'id' appears to be blind SQLi vulnerable (e.g.:
'http://www.moviescope.com/viewprofile.aspx?id=1%20OR%20NOT%20%2816
1%3E161%29')

scan results: possible vulnerabil
[root@parrot]--[~/DSSS]
#
```



OWASP ZAP

<https://www.owasp.org>



Snort

<https://www.snort.org>



Burp Suite

<https://portswigger.net>



HCL AppScan

<https://www.hcltech.com>



w3af

<https://w3af.org>

Module Summary



This module has discussed web server concepts and attacks



It has covered various web server attack tools and countermeasures



It discussed web application architecture and vulnerability stack



It also discussed various web application threats and attacks



It demonstrated different web application attack tools



It discussed various countermeasures against web application attacks



This module also discussed different types of SQL injection attacks and SQL injection tools



Finally, this module ended with a detailed discussion on various countermeasures against SQL injection attacks



In the next module, we will discuss in detail on various wireless attacks and countermeasures

