

Cyber Security

Subject Code:-102045607

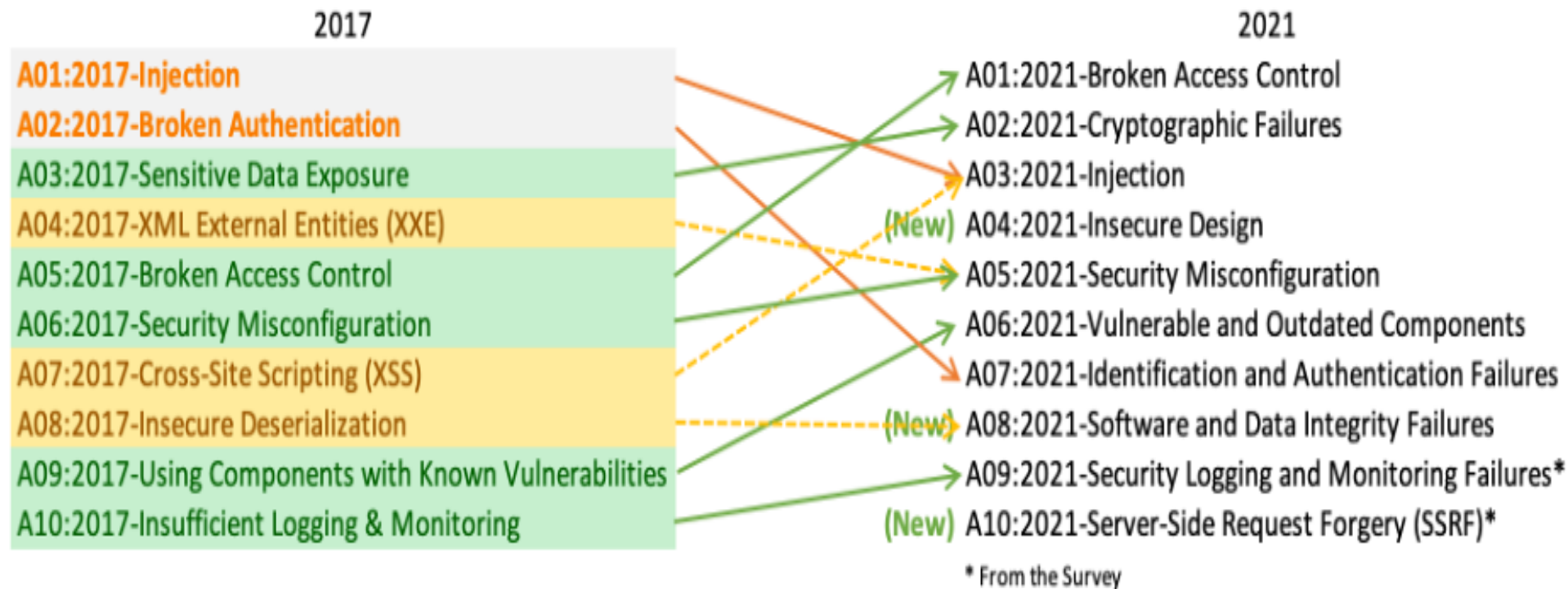
**Unit-5 Web Application
Vulnerabilities**

OWASP-Open Web Application Security Project

- The **Open Web Application Security Project (OWASP)** is an online community that produces freely-available articles, methodologies, documentation, tools, and technologies in the field of web application security.
- The Open Web Application Security Project (OWASP) provides free and open resources. It is led by a non-profit called The OWASP Foundation. The OWASP Top 10 - 2021 is the published result of recent research based on comprehensive data compiled from over 40 partner organizations.

OWASP-Open Web Application Security Project

- **OWASP Top 10 web application security**



OWASP-Open Web Application Security Project

1. Broken Access Control

- Access controls are critical for securing applications against unauthorized access to data and resources. Broken access controls can lead to data compromise, obtaining permissions beyond what's intended for standard users, or account takeover attacks where outsiders hijack user accounts and initiate fraudulent transactions.
- This vulnerability jumped from 5th position in 2017 to 1st in 2021, reflecting that it was found in 94% of tested applications. Common vulnerabilities in this risk category include application logic faults that bypass access control checks by allowing users to change parameter values or force browse to certain URLs.

OWASP-Open Web Application Security Project

1. Broken Access Control

- From a decision-making perspective, it's critical to emphasize the importance of shifting security left in the development cycle. Access controls are harder to implement later, so communicate the importance of implementing proper access controls, such as denying requests by default and rate limiting APIs early on in web app development.

OWASP-Open Web Application Security Project

2. Cryptographic Failures

- Cryptographic failures refer to either a bad implementation of encryption or a complete lack of encryption. The major consequence of a cryptographic failure is that you can potentially expose sensitive data. The exposure of sensitive data can pose compliance, reputational, or competitive business risks depending on what information is not adequately protected by encryption.
- With the average data breach cost at an all-time high of \$4.35 million in 2022, businesses can't afford to slip up with cryptography.

OWASP-Open Web Application Security Project

2. Cryptographic Failures

- Critical to preventing cryptographic failures is first **classifying the data that any web app processes, stores, or transmits**. Then, you can **identify the sensitive data assets** and **ensure they're encrypted both at rest and in transit**.
- A modern encryption solution that uses up-to-date and strong standard algorithms centralizes encryption and encryption key configuration, and manages the encryption key lifecycle is a prudent investment.

OWASP-Open Web Application Security Project

3. Injection

- Injection is a risk category that refers to the ability of threat actors to provide malicious input to web applications that result in the app executing unexpected and unwanted commands. Injection occurs when the app can't distinguish malicious input from its code.
- Common injection attacks include SQL injections that insert malicious SQL queries into input fields or JavaScript injections that load malicious code into the client-side of the web app.

OWASP-Open Web Application Security Project

3. Injection

- Injection attacks can lead to various negative outcomes, including denial of service, privilege elevation, and data breaches. An important strategic element of mitigation is encouraging the use of tools that help to detect injection vulnerabilities in code. Since there are several different injection attacks, you may need more than one tool for thorough testing.

OWASP-Open Web Application Security Project

4. Insecure Design

- This is an entirely new category for the OWASP Top Ten, focusing broadly on application design and architectural flaws that lead to increased security risks.
- When an application is inherently designed in an insecure way, even a perfect implementation of security controls and risks can't compensate for those design weaknesses. Sophisticated threat actors will eventually find and exploit design flaws.

OWASP-Open Web Application Security Project

4. Insecure Design

- At a high level, one of the most important mitigation tips is to **mandate the use of threat modeling for software development teams.**
- Threat modeling should use the structure and data flow inherent to a specific web app to trace out the **key technical threats that could exploit the system.**

OWASP-Open Web Application Security Project

5. Security Misconfiguration

- This category of risks relates to the security components in an application being incorrectly configured.
- **Misconfigurations are increasingly common due to the cloud being used as a development environment and web apps being built with container images.** The infrastructural complexity adds more points at which security misconfigurations can occur.

OWASP-Open Web Application Security Project

5. Security Misconfiguration

- In the data gathered by OWASP current the Top Ten, there were over 200,000 detected instances of security misconfigurations in web apps.
- **The challenge with mitigating security misconfiguration risks from a strategic standpoint is that they cover the whole application stack and the app's infrastructure.** Individual errors are often at play here, such as opening unnecessary ports, not changing default passwords, or leaving cloud storage buckets open.

OWASP-Open Web Application Security Project

6. Vulnerable and Outdated Components

- Web apps comprise many components or building blocks from external sources (libraries, frameworks, etc.). These components handle both back-end and front-end functionality.
- **When threat actors try to compromise an application, they look at its component parts and attempt to exploit any vulnerabilities.** Often, these vulnerabilities come from using out-of-date frameworks or libraries that are easy to exploit.

OWASP-Open Web Application Security Project

6. Vulnerable and Outdated Components

- The overall strategic mitigation here is to **ensure an effective patch management strategy is in place.**
- Part of that strategy entails maintaining an inventory of all the components in your apps and the respective versions of those components the app is running. Ideally, **you'll be able to automate the inventory step with a digital inventory solution.**

OWASP-Open Web Application Security Project

7. Identification and Authentication Failures

- Failures in authentication and identity management make applications vulnerable to threat actors masked as legitimate users.
- Some examples of vulnerabilities include not setting validity periods for session IDs, permitting weak passwords that are easy to guess, and not rate limiting login attempts against automated attacks.
- The solutions include implementing multi-factor authentication in apps and communicating the importance of complying with recommended password length, complexity, and rotation policies to developers.

OWASP-Open Web Application Security Project

8. Software and Data Integrity Failures

- This is another new risk category in the OWASP Top Ten, and it's all about making faulty default assumptions within development pipelines about the integrity of software or data.
- Since web apps regularly rely on plugins and libraries from external sources, a lack of verification of the integrity of these sources introduces the risk of malicious code, unauthorized access, and compromise.
- The main mitigation strategy is ensuring external code or data hasn't been tampered with by requiring digital signatures.

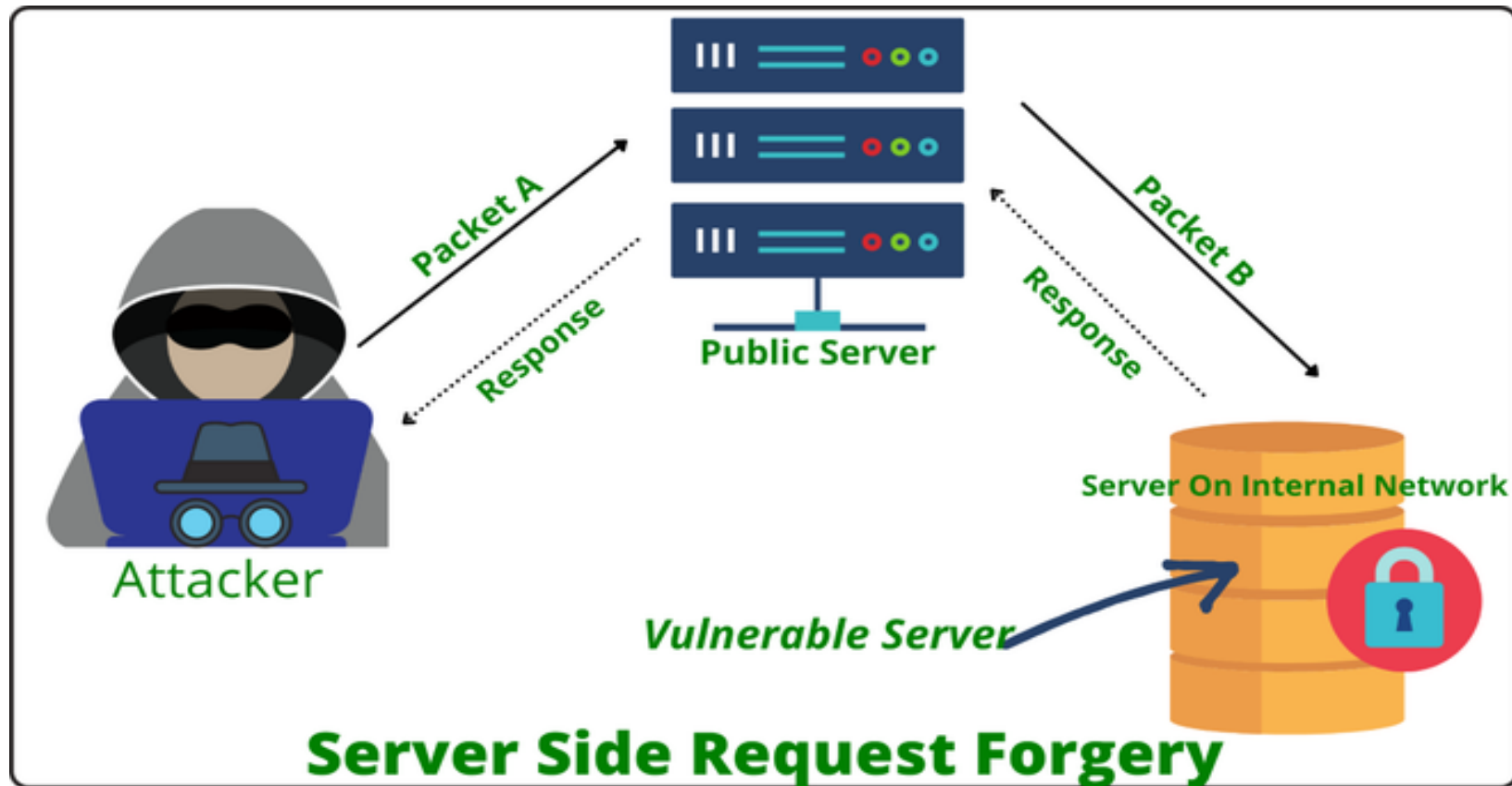
OWASP-Open Web Application Security Project

9. Security Logging and Monitoring Failures

- Logging and monitoring help to provide security accountability, visibility into events, incident alerting, and forensics. When there are failures in these capabilities, your company's ability to detect and respond to application breaches becomes severely compromised.
- To mitigate, use open source or proprietary tools to correlate logs, implement monitoring and alerting, and create an incident recovery and response strategy using established guidelines, such as NIST 800-61r2.

OWASP-Open Web Application Security Project

10. Server-Side Request Forgery (SSRF)



OWASP-Open Web Application Security Project

10. Server-Side Request Forgery (SSRF)

- **SSRF is one of the two OWASP Top Ten risks added based on the community survey rather than data from web apps.** Most web apps today **require external resources for their functionality, which are usually accessed at URLs.**
- **SSRF occurs when hackers can get servers to make requests that they control.** The typical vulnerability is that the web application **doesn't validate the user-supplied URL, potentially allowing access to internal services or resources by bypassing access controls.**

OWASP-Open Web Application Security Project

10. Server-Side Request Forgery (SSRF)

- The strategic concept of defence in depth is important here; **multiple controls at the application and network layers can help to prevent SSRF.** Client-supplied input data should be validated and sanitized, while network segmentation can also help.

Application Inspection tools -Zed Attack Proxy

- Zed Attack Proxy is an open-source security software written in Java programming language and released in 2010.
- It is used to scan web applications and find vulnerabilities in it. It was started as a small project by the Open Web Application Security Project (OWASP) and now it is the most active project maintained by thousands of individuals around the globe.
- It is available for Linux, Windows, and mac in 29 languages. It can also be used as a proxy server like a burp suite to manipulate the request including the HTTPS request.
- Daemon mode is also present in it which can later be controlled by REST API.

Application Inspection tools -Zed Attack Proxy

Features:

- Passive Scanner
- Automated Scanner
- Proxy Server
- Port Identification
- Directory Searching
- Brute Force Attack
- Web Crawler
- Fuzzer

Application Inspection tools -Zed Attack Proxy

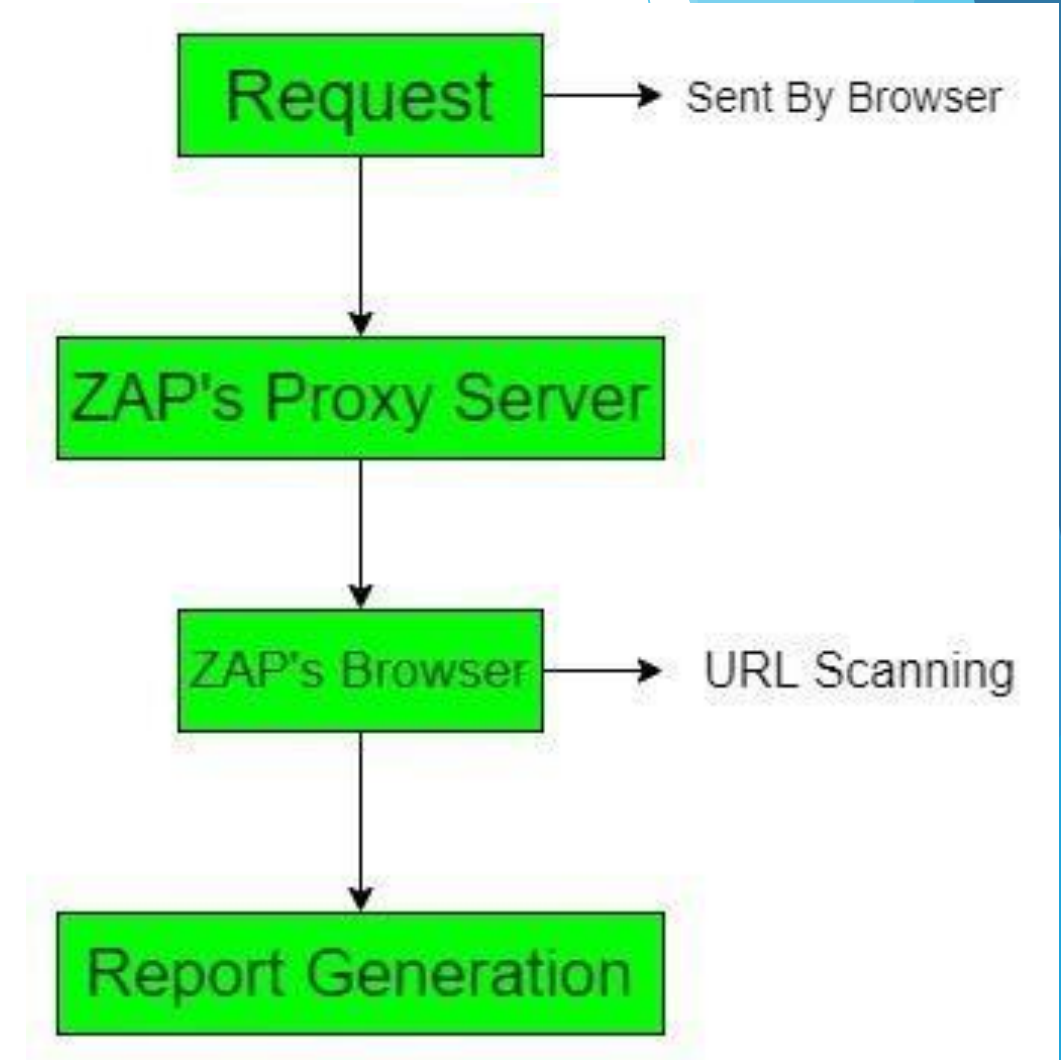
Why do we use Zed Attack Proxy?

- Zed Attack Proxy is used to detect vulnerabilities present on any web server and try to remove them. Here is some big vulnerability that could be present in the web server:
 - SQL injection
 - Cross-site scripting (XSS)
 - Broken access control
 - Security miss-configuration
 - Broken authentication
 - Sensitive data exposure
 - Cross-site request forgery (CSRF)
 - Using components with known vulnerabilities.

Application Inspection tools -Zed Attack Proxy

Working Process:

- First we set up the proxy server with any browser.
- The browser sends website data to the proxy server and then the browser inside the ZAP process the request and perform attacks and generates the report.



Application Inspection tools -Zed Attack Proxy

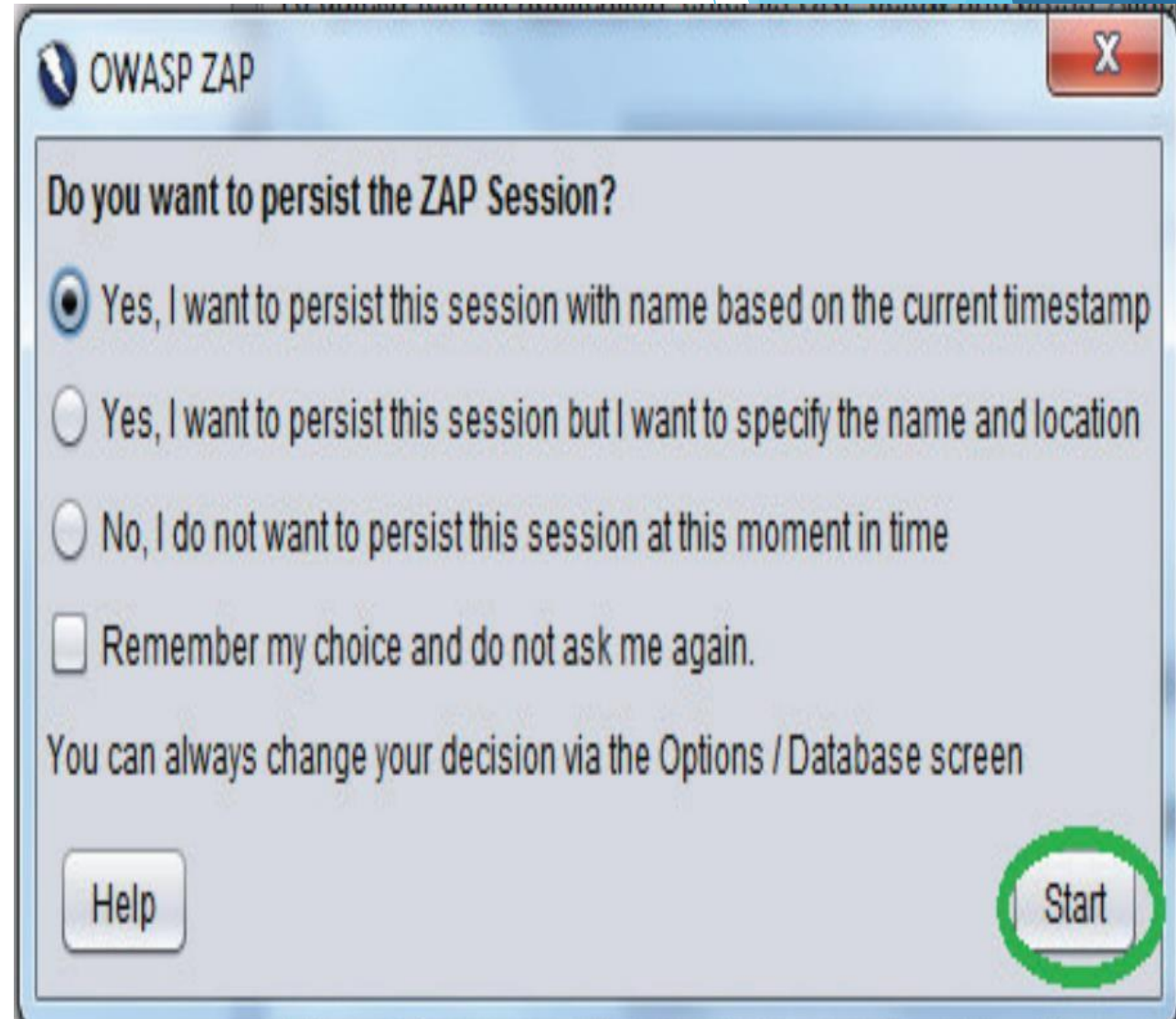
Configuration Steps:

- **Step 1:** Download ZAP from <https://www.zaproxy.org/download/> by selecting the proper operating system.
- **Step 2:** Run the file and follow the instruction until the installation is complete.

Application Inspection tools -Zed Attack Proxy

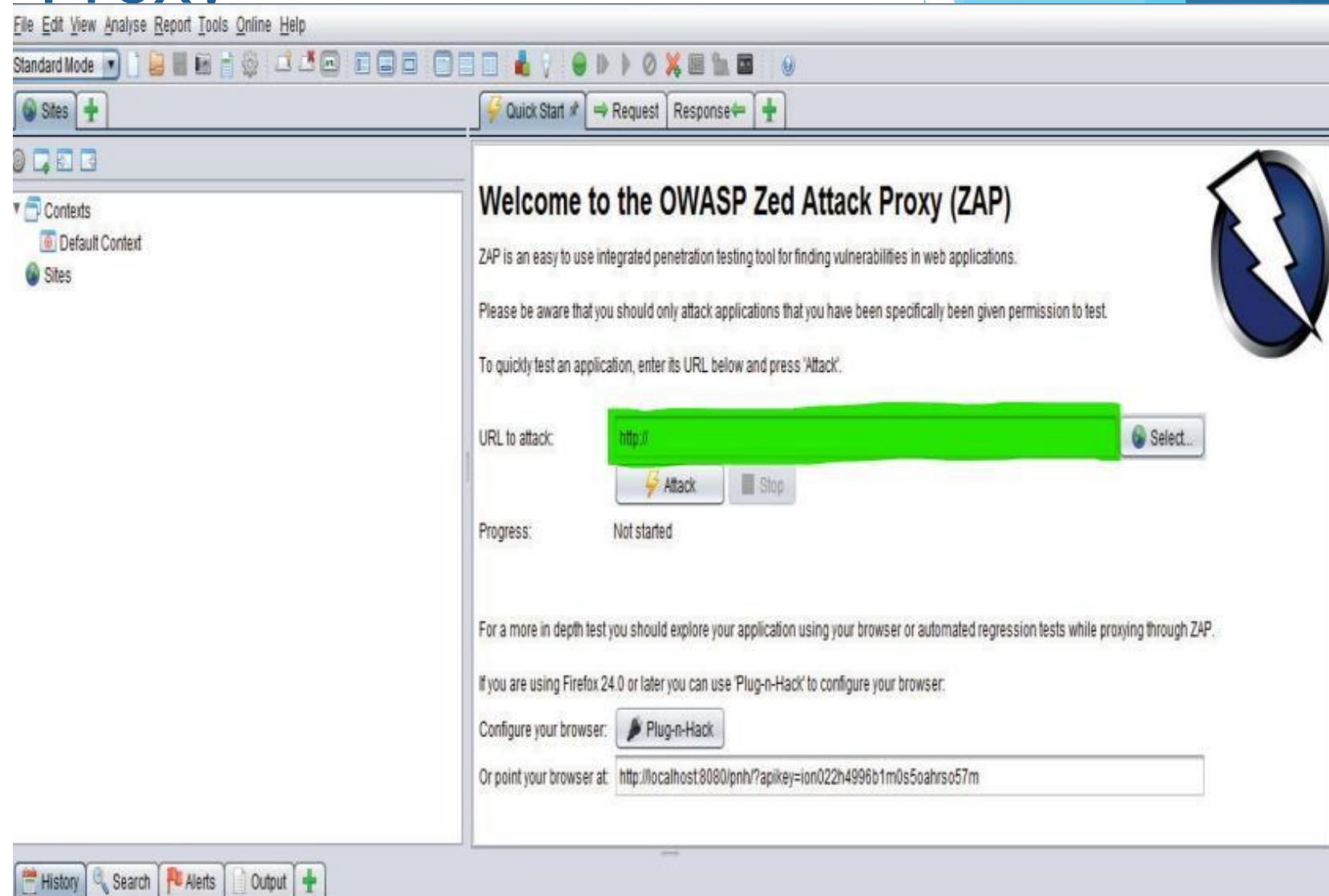
Steps to Run:

- **Step 1:** Open the application through the terminal or by clicking on the icon.
- **Step 2:** In the next step, select the first option and click start.



Application Inspection tools -Zed Attack Proxy

- **Step 3:** Now choose a target to scan and enter its web address in the green highlighted box and click attack.
- **Step 4:** Now you will have to wait for a few minutes to get the result.



sqlmap

- SQLmap is a SQL Injection Tool used to performing Automated Injection in Database and try to fetch tables out of it.
- SQLmap used by WhiteHat and BlackHat hackers. BlackHat try to Exploit random or targeted sites using this tool as a challenge or harming sites.
- But White-hat Hackers (Security Researchers) use that tool for scanning their clients' website for any injectable query if they found they report to Administrator and get bug bounty or Earn Reward from it!

sqlmap


- sqlmap is an open source penetration testing tool that automates the process of detecting and exploiting SQL injection flaws and taking over of database servers.
- It comes with a powerful detection engine, many niche features for the ultimate penetration tester and a broad range of switches lasting from database fingerprinting, over data fetching from the database, to accessing the underlying file system and executing commands on the operating system via out-of-band connections.

```
batch

mutual consent i
al, state and fed
misuse or damage

/IPS
```

```
$ python sqlmap.py -u "http://debiandev/sqlmap/mysql/get_int.php?id=1" --batch
```



```
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program
```

```
[*] starting @ 10:44:53 /2019-04-30/
```

```
[10:44:54] [INFO] testing connection to the target URL
[10:44:54] [INFO] heuristics detected web page charset 'ascii'
[10:44:54] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:44:54] [INFO] testing if the target URL content is stable
[10:44:55] [INFO] target URL content is stable
[10:44:55] [INFO] testing if GET parameter 'id' is dynamic
[10:44:55] [INFO] GET parameter 'id' appears to be dynamic
[10:44:55] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable
(possible DBMS: 'MySQL')
```


sqlmap

Feature:

- Full support for MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, MariaDB, MemSQL, TiDB, CockroachDB, HSQLDB, H2, MonetDB, Apache Derby, Amazon Redshift, Vertica, Mckoi, Presto, Altibase, MimerSQL, CrateDB, Greenplum, Drizzle, Apache Ignite, Cubrid, InterSystems Cache, IRIS, eXtremeDB, FrontBase, Raima Database Manager, YugabyteDB and Virtuoso database management systems.
- Full support for six SQL injection techniques: boolean-based blind, time-based blind, error-based, UNION query-based, stacked queries and out-of-band.

sqlmap

- Support to directly connect to the database without passing via a SQL injection, by providing DBMS credentials, IP address, port and database name.
- Support to enumerate users, password hashes, privileges, roles, databases, tables and columns.
- Automatic recognition of password hash formats and support for cracking them using a dictionary-based attack.
- Support to **dump database tables** entirely, a range of entries or specific columns as per user's choice. The user can also choose to dump only a range of characters from each column's entry.

sqlmap

- Support to **search for specific database names, specific tables across all databases or specific columns across all databases' tables**. This is useful, for instance, to identify tables containing custom application credentials where relevant columns' names contain string like name and pass.
- Support to **download and upload any file** from the database server underlying file system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.
- Support to **execute arbitrary commands and retrieve their standard output** on the database server underlying operating system when the database software is MySQL, PostgreSQL or Microsoft SQL Server.

DVWA(Damn Vulnerable Web App)

- Damn Vulnerable Web Application, shorter DVWA, is a PHP/MySQL web application that is damn vulnerable.
- The main goal of this pentesting playground is to aid penetration testers and security professionals to test their skills and tools.



DVWA(Damn Vulnerable Web App)

- DVWA is a damn vulnerable web application coded in PHP that uses MySQL database.
- With this amazing pen testing web app you can practice some of the most common web vulnerabilities (different levels of difficulty) using its very simple GUI.
- You can play around and try to discover as many issues as possible in order to deepen your knowledge/skill set.

DVWA(Damn Vulnerable Web App)

DVWAAttacks:

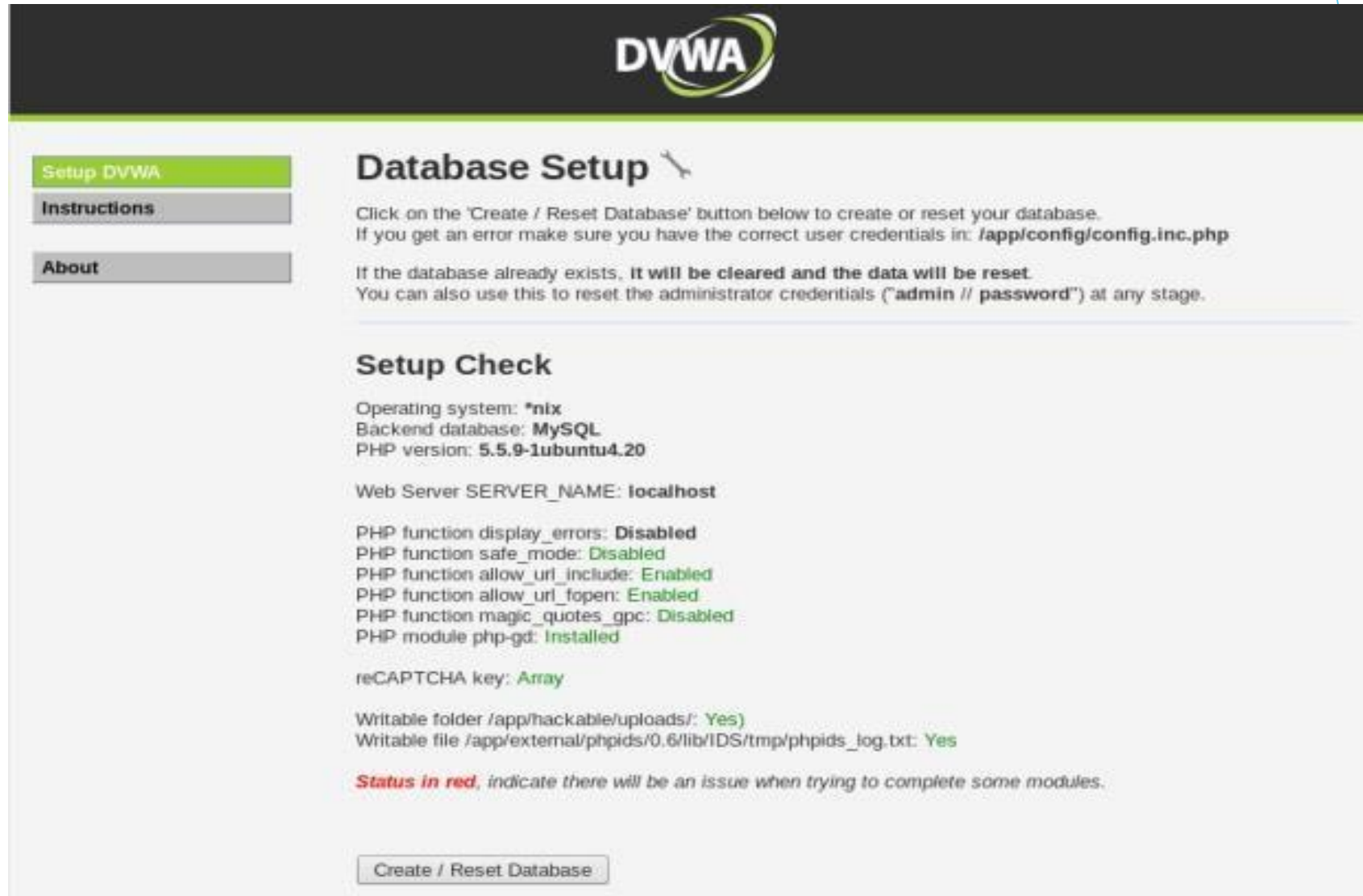
- Brute-force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection / SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass

DVWA(Damn Vulnerable Web App)

Requirements:

- web server (XAMPP as an alternative)
- PHP
- MySQL
- Other possible dependencies (depending on the OS)

DVWA(Damn Vulnerable Web App)



The screenshot shows the DVWA web application interface. At the top is a black header with the DVWA logo. Below the header is a left sidebar with three menu items: 'Setup DVWA' (highlighted in green), 'Instructions', and 'About'. The main content area is titled 'Database Setup' with a wrench icon. It contains instructions on how to create or reset the database, mentioning the file path `/app/config/config.inc.php` and the default administrator credentials 'admin // password'. Below this is a 'Setup Check' section that lists various system and configuration details. Most items are in green, indicating they are correct, but 'display_errors' is 'Disabled' in red. At the bottom of the main area is a 'Create / Reset Database' button.

DVWA

Setup DVWA

Instructions

About

Database Setup

Click on the 'Create / Reset Database' button below to create or reset your database.
If you get an error make sure you have the correct user credentials in: `/app/config/config.inc.php`

If the database already exists, **it will be cleared and the data will be reset**.
You can also use this to reset the administrator credentials ("**admin // password**") at any stage.

Setup Check

Operating system: ***nix**
Backend database: **MySQL**
PHP version: **5.5.9-1ubuntu4.20**

Web Server SERVER_NAME: **localhost**

PHP function display_errors: **Disabled**
PHP function safe_mode: **Disabled**
PHP function allow_url_include: **Enabled**
PHP function allow_url_fopen: **Enabled**
PHP function magic_quotes_gpc: **Disabled**
PHP module php-gd: **Installed**


reCAPTCHA key: **Array**

Writable folder `/app/hackable/uploads/`: **Yes**
Writable file `/app/external/phpids/0.6/lib/IDS/tmp/phpids_log.txt`: **Yes**

Status in red, indicate there will be an issue when trying to complete some modules.

Create / Reset Database

DVWA(Damn Vulnerable Web App)



The logo for DVWA (Damn Vulnerable Web App) features the letters "DVWA" in a bold, dark blue font. To the right of the text is a stylized graphic consisting of two curved, overlapping lines, one in a light green color and the other in a dark blue color, forming a partial circle or swoosh around the text.

Username

Password

Login

DVWA(Damn Vulnerable Web App)



[Home](#)
[Instructions](#)
[Setup / Reset DB](#)

[Brute Force](#)
[Command Injection](#)
[CSRF](#)
[File Inclusion](#)
[File Upload](#)
[Insecure CAPTCHA](#)
[SQL Injection](#)
[SQL Injection \(Blind\)](#)
[Weak Session IDs](#)
[XSS \(DOM\)](#)
[XSS \(Reflected\)](#)
[XSS \(Stored\)](#)
[CSP Bypass](#)
[JavaScript](#)

Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with various levels of difficulty, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

DVWA(Damn Vulnerable Web App)

Windows

- If you don't have a ready web server, the easiest steps are the following:
 - Download DVWA
 - Install XAMPP
 - Unzip dvwa.zip and place files into public html folder
 - Browse to <http://127.0.0.1/dvwa/setup.php>

DVWA(Damn Vulnerable Web App)

- Owasp Top 10 web application security
- Application Inspection tools – Zed Attack Proxy
- Sqlmap
- DVWA

Thank you!