

Logical Weapons

INFORMATION IN THIS CHAPTER

- Reconnaissance Tools
- Scanning Tools
- Access and Escalation Tools
- Exfiltration Tools
- Sustainment Tools
- Assault Tools
- Obfuscation Tools

Logical tools are the weapons that we likely envision when discussing cyber warfare. These are the set of tools that is used to conduct reconnaissance, scout out the networks and systems of our opponents, and attack the various targets we might find. When we look at the use of such tools in a cyber warfare context, we might ask how they are different than the tools used in everyday penetration testing of applications, systems, and networks. The answer is that, in many cases, they are not conceptually different to any great degree, but the scope of their use is greatly increased in a cyber warfare scenario. In April of 2013, the U.S. Air Force officially designated six cyber capabilities as weapons systems: Air Force Cyberspace Defense, Cyberspace Defense Analysis, Cyberspace Vulnerability Assessment/Hunter, Cyber Command and Control Mission System, Air Force Intranet Control, and Cyber Security and Control System [1]. This reflects the changing nature of attitudes regarding cyber warfare as “real” war.

Where penetration testers may be bound, contractually in some cases, to shy away from the tools or settings in tools that are labeled “dangerous” due to their possible deleterious effects on the target at the other end, such effects may be acceptable, or even desirable in a cyber conflict. This may not always be the case, and we certainly may still want to be stealthy and cautious in some scenarios, but this opens up the use of the common tools in such a way that we do not normally see in penetration testing outside of a lab environment.

Another common question that arises in discussions of tools that might be used during cyber warfare is that of “secret” military or government tools. There are always rumors of gigantic military botnets that are a billion nodes strong, or tools that can cut through

encryption like butter. As we have yet to see a no-holds-barred cyber war publicly erupt, the good answer to this question is that although such tools may exist, we will not know the specifics regarding them until they are brought out of hiding and publicly deployed.

Given past examples of military weapons that were held in extreme secrecy, such as the Manhattan Project and the Stealth Fighter, we would certainly be rash to assume that similar projects do not exist for cyber weaponry, or that super-skilled hackers are not being trained in the bowels of the National Security Agency (NSA). However, from the examples that we can see publicly, government and military cyber warriors are going through the same training, in many areas, and using many of the same tools as their counterparts in the civilian world.

In the case of individuals, corporations, hacktivists, criminal organizations, and other nonstate actors, we are more likely to see the use of common rather than custom developed tools. Such groups are often presumed to be the origin of many of the more pervasive items of malware, attack websites, and the source of any number of small-scale cyber attacks. As such, they are on a good footing to participate in cyber warfare on a larger scale and certainly can be considered a serious threat.

When we discuss the broad categories of tools: reconnaissance, scanning, infrastructure, application, and operating system, we might also consider the sources of such tools. A very large portion of the tools in the arsenal used by cyber warriors, penetration testers, hacktivists, and terrorists are free and/or open source, and are regularly maintained and enhanced by their base of users. There are also quite a few commercial tools, or tools that are free with commercial components, some of which are very good indeed, but can be quite expensive.

NOTE

The selection of tools available for use in cyber warfare, penetration testing, and security in general is truly staggering. Although a complete discussion of the various popular security tools would have been great to be able to include, we would have had to devote an entire book to it to have been able to do so. In this chapter, we discuss a few of the highlights, but for those still wanting more, Insecure.org is a great resource. They maintain lists of password crackers, sniffers, vulnerability scanners, web scanners, wireless tools, and numerous other tools of the trade.

We may very well find commercial tools in the hands of cyber warfare forces that are backed by, or in the employ of, nation-states, but we are less likely to find them in the hands of individuals or small groups. Nonetheless, in skilled hands, the free tools can be highly effective, if less automated, and are used regularly by a variety of attackers.

RECONNAISSANCE TOOLS

Reconnaissance tools, as should be clear from the name, are those that we use to gather information, usually in a passive state, about the networks and systems that we might plan to take action against in a logical sense. Such efforts may include gathering information from public websites, looking up Domain Name System (DNS) records, collecting metadata from accessible documents, retrieving very specific information through the use of search engine,

or any of a number of other similar activities. Many such tools match closely with Open Source Intelligence (OSINT) techniques.

General Information Gathering

When looking for general information that can be used to provide intelligence on a target, there are a variety of sources that we can turn to. We can mine websites for data on companies and individuals, we can search job postings for a variety of information, we can look for personal and technical information in resumes, we can use search engines both in a general and very specific sense, and we can also use specialized searching tools such as Maltego. In all likelihood, we will utilize a combination of such techniques to assemble a more complete picture of our target.

Websites and Web Servers

All manner of interesting information can be found on the websites of individuals and organizations. Some such information may be intentionally displayed, such as corporate organizational information, and some of it may be shared in an unintentional or unauthorized manner.

In 2007, the U.S. Internal Revenue Service (IRS) conducted an internal audit to inventory servers with the aim of ensuring that security maintenance and patching activities were taking place properly. In the course of the audit, 1811 unauthorized web servers were discovered, constituting 87% of the total web servers discovered. Of these servers, 661 were being used for legitimate business purposes, but were still operating outside of the processes that would have ensured that they were operating in secure configurations [2]. When compliance failures are found in environments that are theoretically highly secure, the implications regarding the security of servers in less strictly regulated environments are frightening indeed.

Search Engines

Search engines, such as Google, can be of great use when conducting research for an attack. They can be used to collect information regarding a particular target, look up application or hardware details, or even collect very specific information to locate vulnerabilities in the target environment.

Even more specifically, search engines can be used to collect data that does not appear during casual searching. Such methods often involve very specifically targeted queries to the standard search engines, or the use of specially tuned search engines, such as Pipl.com, that return information within a particular area of focus.

Google Hacking

Google hacking is the use of advanced operators in search engine queries, in order to enable more directly targeted searches. Although the name would tend to indicate that such searching would be specific to the Google search engine, in actuality, similar search parameters can be used with almost any search engine. Lists of such operators can generally be

found on the page for the search engine in question. For Google, the advanced operators can be found at <https://sites.google.com/site/gwebsearcheducation/advanced-operators> and for Bing at <http://msdn.microsoft.com/en-us/library/ff795620.aspx>. For most search engines, we can find an advanced operator listing by searching for the engine name and “advanced operators.” Although we will likely find some variation in advanced query construction from one search engine to the next, the construction is often fairly similar. For example, if we wanted to search Google for pages on the [Syngress.com](http://syngress.com) website that contained the string “advanced operators” in the page text, we could put together a search string like:

```
http://www.google.com/search?hl=en&lr=&ie=UTF-8&oe=UTF-8&q=intext: "advanced operators" site:syngress.com
```

More specific to security-related issues, we can also construct searches like:

```
http://www.google.com/search?hl=en&lr=&ie=UTF-8&oe=UTF-8&q=intext: "enable secret 5$"
```

Such searches will often locate configuration files for Cisco devices that contain the encrypted (and easily decrypted) administrative passwords for the device in question [3]. Such configuration files can also contain IP addresses for the networks on which they rest, effectively giving the keys to the kingdom to the entirety of the Internet.

Quite a bit of very specific information pertaining to these types of searches is available for public consumption. The book *Google Hacking for Penetration Testers, Volume 2* (ISBN: 978-1-59749-176-1, Syngress) by Johnny Long is an entire volume dedicated to this specific subject. Also available from and maintained by the same author is the Google Hacking Database (GHDB) at <http://www.hackersforcharity.org/ghdb/>. The GHDB contains a wide variety of security specific searches and makes them available to the public through a few simple clicks.

The Deep Web

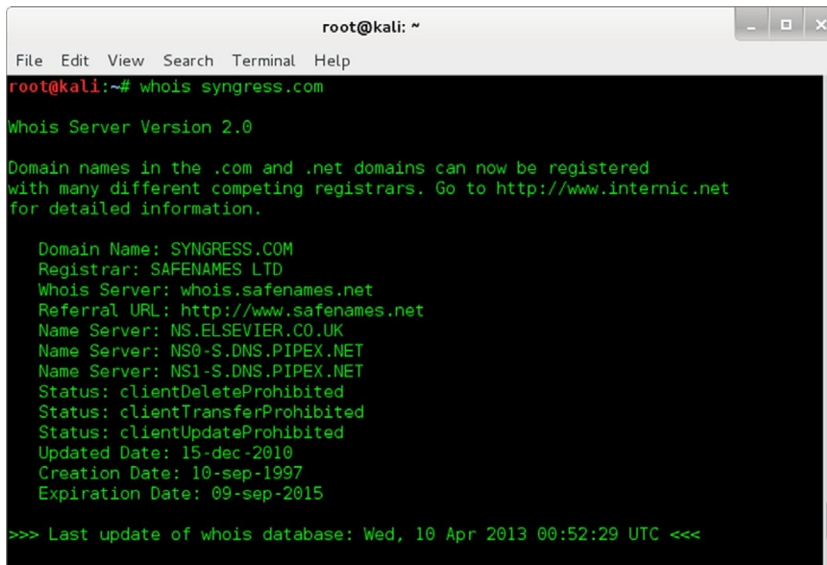
When search engines crawl the Internet to construct the indexes on which their search results are based, they touch only the very surface of the information that is available. Great unexplored depths of information exist unplumbed due to the nature of such indexing. When we are conducting research on a target, we may very well like to see some of this information.

In recent years, several specialized search engines, such as Shodan,^a have come into being to provide access to some portion of this hidden information. These search engines are generally rather specialized in the information that they provide.

Whois

Whois is a tool used to query the globally distributed set of databases that contain the information regarding domain names around the world. The databases contain information

^a<http://www.shodanhq.com>

A screenshot of a terminal window titled 'root@kali: ~'. The terminal shows a menu bar with 'File Edit View Search Terminal Help'. The command 'root@kali:~# whois syngress.com' has been entered. The output is as follows:

```
Whois Server Version 2.0

Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

Domain Name: SYNGRESS.COM
Registrar: SAFENAMES LTD
Whois Server: whois.safenames.net
Referral URL: http://www.safenames.net
Name Server: NS.ELSEVIER.CO.UK
Name Server: NS0-S.DNS.PIPEX.NET
Name Server: NS1-S.DNS.PIPEX.NET
Status: clientDeleteProhibited
Status: clientTransferProhibited
Status: clientUpdateProhibited
Updated Date: 15-dec-2010
Creation Date: 10-sep-1997
Expiration Date: 09-sep-2015

>>> Last update of whois database: Wed, 10 Apr 2013 00:52:29 UTC <<<
```

FIGURE 6.1 A Whois query from the command line.

regarding when the domain was registered or last updated, which registrar it was registered with, contact information for the owners of the domain, and the name servers that are used to resolve requests sent to the domain name. We can see part of the reply from a basic `whois` query in Figure 6.1. One of the more interesting items of information displayed here is the nameserver to which the domain name is directed, which will lead us to additional information in the next section.

The information displayed in Figure 6.1 is the result of a command line `whois` query, a tool often found in Linux and Unix operating systems, but not so common in others, such as those distributed by Microsoft. We can also run such queries through a variety of web pages dedicated to such purposes, one of the more common being whois.net.

In some cases, the contact information found in the data returned from `whois` queries will contain a great deal of useful information, such as a physical address, phone number, and contact name from someone directly associated with the domain. Such information can be used as the basis for conducting searches for additional information when researching a target. In recent years, however, it has become more common for domains to be registered through a service that acts as a proxy for domain contact information, thus hiding the actual contact information for those associated with the domain.

In addition to conducting `whois` queries on domain names, we can also run queries on IP addresses. The information from these queries is returned from the databases maintained by the Regional Internet Registries (RIR), who keep track of IP address assignments for their particular regions. The RIRs are distributed as follows:

- North America and some of the surrounding regions—American Registry for Internet Numbers (ARIN).

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# whois 66.33.206.206
#
# The following results may also be obtained via:
# http://whois.arin.net/rest/nets;q=66.33.206.206?showDetails=true&showARIN=false&ext=netref2
#
NetRange:        66.33.192.0 - 66.33.223.255
CIDR:            66.33.192.0/19
OriginAS:
NetName:         DREAMHOST-BLK1
NetHandle:       NET-66-33-192-0-1
Parent:          NET-66-0-0-0-0
NetType:         Direct Allocation
Comment:         ADDRESSES WITHIN THIS BLOCK ARE NON-PORTABLE
Comment:         ** For abuse issues, please contact abuse@dreamhost.com **
RegDate:         2002-04-26
Updated:         2012-03-02
Ref:             http://whois.arin.net/rest/net/NET-66-33-192-0-1

OrgName:         New Dream Network, LLC
OrgId:           NDN
Address:         417 Associated Rd.
Address:         PMB #257

```

FIGURE 6.2 A Whois query on an IP address.

- Europe, the Middle East, and some of Asia—Réseaux IP Européens Network Coordination Centre (RIPE NCC).
- Asia Pacific—Asia Pacific Network Information Centre (APNIC).
- Latin America and the Caribbean—Latin American and Caribbean Internet Address Registry (LACNIC).
- Africa—AfriNIC.

Shown in Figure 6.2 are some of the results of an IP address query against an IP controlled by ARIN.

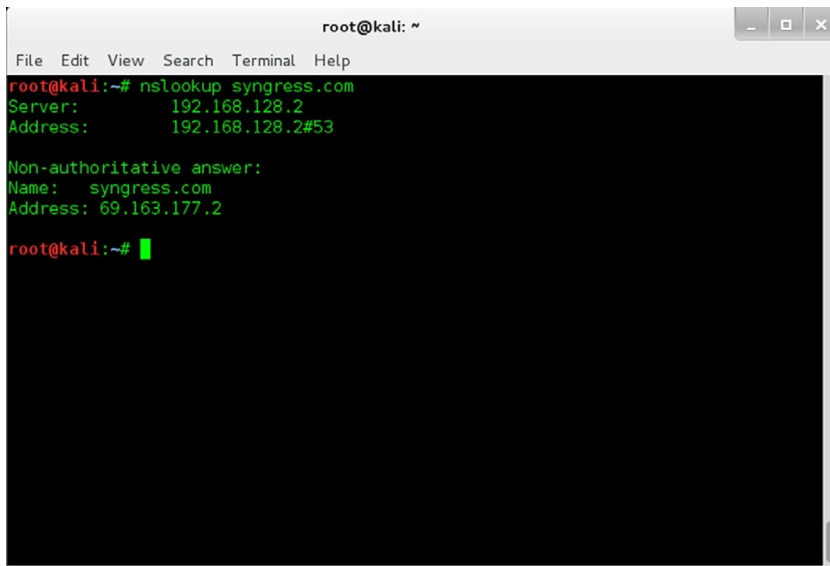
We can also conduct such queries based on information other than an IP address, such as a point of contact or an organization name.

DNS

Given the nameservers of our target from the `whois` queries that we have conducted, we can query them for still more information. The DNSs are responsible for fulfilling name resolution requests for clients that are attempting to resolve the domain name to an IP address.

Numerous tools are available to aid us in our quest for DNS information. We can conduct command line queries in most operating systems by using the `nslookup` command, shown in Figure 6.3. `Nslookup` will generally return the IP address of the DNS server, which we can then interrogate for additional information.

Ideally, we would like to conduct a zone transfer against the DNS server, causing it to send us a complete copy of the record that it has, and allowing us to get a fairly complete view of

A terminal window titled 'root@kali: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the command 'nslookup syngress.com' and its output: 'Server: 192.168.128.2', 'Address: 192.168.128.2#53', 'Non-authoritative answer:', 'Name: syngress.com', and 'Address: 69.163.177.2'. The prompt 'root@kali:~#' is visible at the bottom.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nslookup syngress.com
Server:      192.168.128.2
Address:     192.168.128.2#53

Non-authoritative answer:
Name:   syngress.com
Address: 69.163.177.2

root@kali:~#
```

FIGURE 6.3 Nslookup information for [Syngress.com](https://syngress.com).

the machines that it knows. In some cases, we can still use `nslookup` for this purpose. In Windows, we can do this with an interactive mode command like:

```
nslookup
> server [DNS server name or IP]
> set type=any
> ls -d [domain name]
```

In most cases, this will fail to return the information we are looking for, as DNS servers will not generally perform a zone transfer to an arbitrary requestor on the Internet. Additionally, `nslookup` may have the zone transfer functionality disabled in some operating systems. A similar tool to `nslookup`, called `dig`, can be found on most Linux systems and is capable of conducting zone transfers where the DNS server in question is willing to cooperate.

Even in the case of not being able to get a full zone transfer from a DNS server, we can still query other useful information. In [Figure 6.4](#), we can see the results of a `dig` query asking for the MX records for the domain syngress.com. In this case, we now have a starting point for further investigation on this domain, as we have located its mail server.

As with the `whois` records, DNS information can also be requested from a variety of public servers on the Internet. One of many sites that provide such functionality is dnsquery.org. Additionally, a variety of other tools exist to query DNS servers, even to the point of brute forcing through possible subdomain and hostnames, using tools such as `dnsenum`.

Metadata

Metadata is data about data. For instance, if we have a file containing the text of this chapter, and the file has a file size, last accessed timestamp, and bits set for file permissions, none of

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# dig @66.33.206.206 syngress.com -t MX

; <<>> DiG 9.8.4-rpz2+r1005.12-P1 <<>> @66.33.206.206 syngress.com -t MX
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 37114
;; flags: qr aa rd; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;syngress.com.                IN      MX

;; ANSWER SECTION:
syngress.com.                14400   IN      MX      0 elsevier.com.s200a1.psmtip.com.

;; Query time: 58 msec
;; SERVER: 66.33.206.206#53(66.33.206.206)
;; WHEN: Tue Apr 9 21:00:49 2013
;; MSG SIZE rcvd: 72

root@kali:~# █

```

FIGURE 6.4 Dig information for [Syngress.com](http://syngress.com).

this data has anything directly to do with the contents of the file itself, but is data about the file storing the text. Although such information may seem to be rather mundane outside of digital forensics circles, some of the information contained in document or image metadata may be very interesting indeed. We may find items such as the usernames that have edited the file, paths where the file has been stored, previous revisions of the text, coordinates that indicate where a picture was taken, image thumbnails, or any of hundreds of other items of information, all stored in the file with the actual intended content.

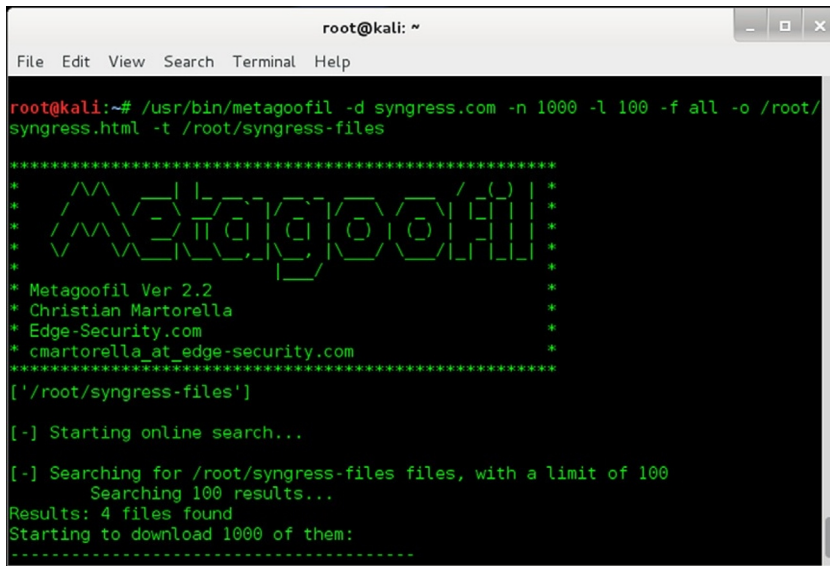
Metagoofil

Metagoofil is an excellent tool for hunting down metadata. Metagoofil is a script that conducts very directed Google searches, using some of the advance operators that we discussed earlier in the Google hacking section of this chapter, to locate documents that are stored on the web servers of a given domain name. In [Figure 6.5](#), we can see the launch of a Metagoofil search on the domain syngress.com.

Once these documents have been located, they are downloaded and parsed for interesting information in the document metadata, which is then displayed in html format for easy perusal of the user.

Exiftool

Exiftool is another wonderful tool for extracting metadata from documents and images. Exiftool is named for a type of metadata, called EXIF data that is normally attached to image files. This data can include information regarding the equipment that the image was created on, including serial numbers, thumbnails of the original image, coordinates where the image was created, for GPS-enabled devices, and a host of other information.



```

root@kali: ~
File Edit View Search Terminal Help

root@kali:~# /usr/bin/metagoofil -d syngress.com -n 1000 -l 100 -f all -o /root/
syngress.html -t /root/syngress-files

*****
*                               *
*  A E D G O O F I L          *
*  \ / \ / \ / \ / \ / \ / \ *
*  \ / \ / \ / \ / \ / \ / \ *
*                               *
* Metagoofil Ver 2.2           *
* Christian Martorella         *
* Edge-Security.com           *
* cmartorella_at_edge-security *
*                               *
*****
['/root/syngress-files']

[-] Starting online search...

[-] Searching for /root/syngress-files files, with a limit of 100
    Searching 100 results...
Results: 4 files found
Starting to download 1000 of them:
-----

```

FIGURE 6.5 Metagoofil information for Syngress.com.

WARNING

When taking pictures on most GPS-enabled devices, including almost all modern cell phones, and quite a few cameras, the location information is often embedded in the EXIF data of the image file. On some devices, this functionality cannot be disabled without disabling the GPS entirely. When posting images for public consumption, it is always a good idea to review the EXIF data beforehand to see what exactly we will be sharing.

Strings

Strings is a utility that will parse a given file for strings of text, generally consisting of several printable characters in a row. Strings can be very helpful in finding data hidden in files, even data, such as deleted content, which may not be accessible through normal applications that are used to access and manipulate the file. Strings is a common tool on Linux and Unix distributions and is available as a download for Microsoft operating systems.

We can use strings to locate metadata not only in documents and images, but also in a variety of other files as well. Although some of the other metadata-centric tools may be more efficient at finding known metadata, strings will find all of the strings in a given file. We may get back quite a bit of irrelevant or useless data, but we will likely get back all of the data that is in the file in plaintext.

Maltego

Throughout this section, we have discussed a number of types of data that can be helpful when conducting reconnaissance on a target. We have also talked about a number of tools that

For information that might be gained from DNS servers, we are somewhat limited in the steps that we can take to not over share our data. We can deny zone transfers to unknown machines, so as to not give our information away wholesale. We can also use domain registration and hosting services that are willing to proxy our actual information so that we do not share network or company data that might be of use to an attacker. Such actions might not always be appropriate for a business environment, as we may not wish to hide this information from our customers, depending on our line of business.

Data leakage via metadata is one area in which, at least from a technical perspective, we can easily limit what we are sharing to the outside world. Recent versions of many tools that are used to produce documents these days, such as Microsoft Word or Adobe Acrobat, have functionality built into the application to scrub the metadata from them before they are released externally. These tools often do a very good job, but it often pays to check with a secondary source, such as the strings utility, just in case something was missed. As images often do not undergo the same processing as a document before they are used, we will also want to use something along the lines of Exiftool to ensure that we have not inadvertently included any information that we did not intend to.

SCANNING TOOLS

Scanning tools are the category of tools that we use to find more information about our target environment, the systems within it, and the details of those systems. With such tools, we can be very general, in the case of running ping sweeps; somewhat more specific, in the case of running port scans; or very specific, in the case of grabbing banners or enumerating users on particular systems.

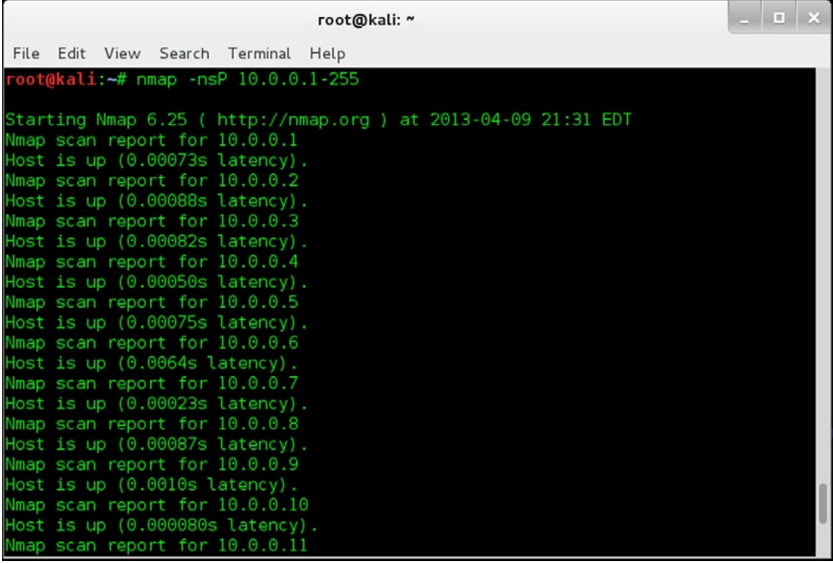
Given the limits of our discussion on tools in this chapter, we have grouped network mapping, port scanning, and enumeration tools together in one section. Each of these areas could deservedly be the focus of its own chapter, but we will go over some of the highlights here.

Nmap

Nmap is a wonderful tool. It is principally a port scanner, but can do quite a bit more as well. It can be used to ping IPs, detect vulnerabilities, fingerprint operating systems, run traceroutes, and much more. Almost all of the uses to which nmap can be put can also be tweaked in various ways to avoid detection, alter the speed at which it carries out its processes, change methods of communication, and more. Nmap is truly a versatile tool. Additionally, nmap is a free tool and ships with many Linux and Unix operating systems. Nmap is also available for Windows. In addition to the command line version that we will be looking at in this section, there are also a variety of GUIs that can be used as a frontend to nmap, including Zenmap which was created by the author of nmap.

Depending on what our actual goal is when running nmap, we might construct a command in a variety of ways. To do a basic ping sweep of a subnet, we might do something like this:

```
nmap -nsP -n 10.0.0.1-254
```

A screenshot of a terminal window titled 'root@kali: ~'. The terminal shows the command 'nmap -nsP 10.0.0.1-255' being executed. The output displays the Nmap version (6.25) and the start time (2013-04-09 21:31 EDT). It then lists scan reports for each IP from 10.0.0.1 to 10.0.0.11, all indicating the host is up with various latency times.

```
root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -nsP 10.0.0.1-255

Starting Nmap 6.25 ( http://nmap.org ) at 2013-04-09 21:31 EDT
Nmap scan report for 10.0.0.1
Host is up (0.00073s latency).
Nmap scan report for 10.0.0.2
Host is up (0.00088s latency).
Nmap scan report for 10.0.0.3
Host is up (0.00082s latency).
Nmap scan report for 10.0.0.4
Host is up (0.00050s latency).
Nmap scan report for 10.0.0.5
Host is up (0.00075s latency).
Nmap scan report for 10.0.0.6
Host is up (0.0064s latency).
Nmap scan report for 10.0.0.7
Host is up (0.00023s latency).
Nmap scan report for 10.0.0.8
Host is up (0.00087s latency).
Nmap scan report for 10.0.0.9
Host is up (0.0010s latency).
Nmap scan report for 10.0.0.10
Host is up (0.00080s latency).
Nmap scan report for 10.0.0.11
```

FIGURE 6.7 Nmap scan results.

This example performs a basic probe of each IP in the specified range to see if anything responds (`-sP`), and does not attempt to resolve names (`-n`), which will speed us up a bit. We can see the results in [Figure 6.7](#). If we wanted to get a little more information back, we could alter our command to conduct a ping sweep, like so:

```
nmap -nsT 10.0.0.247
```

This will both probe each IP to see if anything responds and conduct a port scan using the default settings when a device is found (`-sT`). By default, nmap will scan the 1000 most commonly used ports. As we can see from [Figure 6.8](#), this returns us quite a bit of useful information.

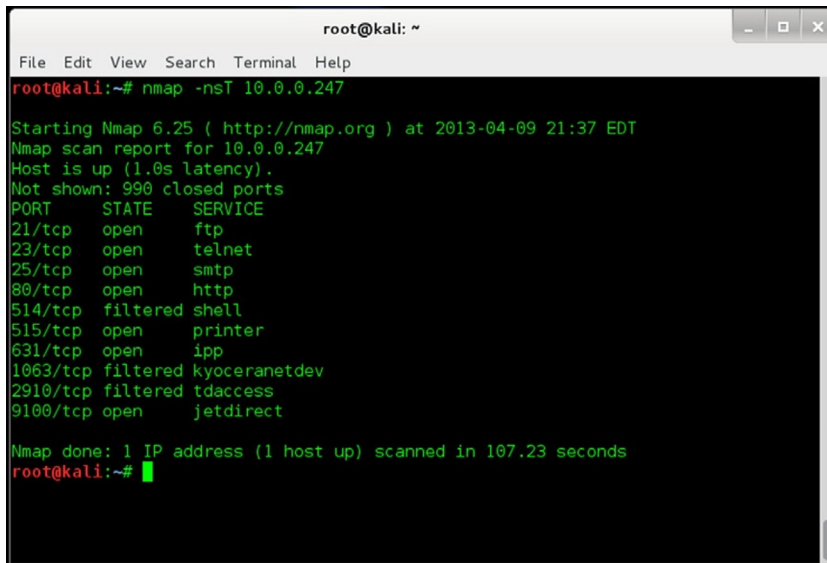
When running both the ping sweep and the port scan, we can see quite a bit of difference in the amount of time that each takes. Given the small range of IPs that we are scanning, the ping sweep will likely return in a minute or so, whereas the port scan could take hours.

TIP

Although an nmap scan is running, we can press enter (or one of several other keys) in the terminal window to get an estimated time of completion. We can watch the flow of packets that nmap is sending by pressing `p` in the terminal window to enable packet tracing. To switch back to the normal nmap mode, press `shift+p`.

We can continue to add complexity to our nmap searches by adding additional features and can indeed spend quite a bit of time constructing complex nmap commands. One compound switch that incorporates several of the others is the `-A` switch:

```
nmap -A 10.0.0.1-254
```

A screenshot of a terminal window titled 'root@kali: ~'. The terminal shows the command 'nmap -nS 10.0.0.247' being executed. The output includes the Nmap version (6.25), the target IP (10.0.0.247), and a list of open and filtered ports with their corresponding services. The scan took 107.23 seconds to complete.

```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nmap -nS 10.0.0.247  
  
Starting Nmap 6.25 ( http://nmap.org ) at 2013-04-09 21:37 EDT  
Nmap scan report for 10.0.0.247  
Host is up (1.0s latency).  
Not shown: 990 closed ports  
PORT      STATE SERVICE  
21/tcp    open  ftp  
23/tcp    open  telnet  
25/tcp    open  smtp  
80/tcp    open  http  
514/tcp   filtered shell  
515/tcp   open  printer  
631/tcp   open  ipp  
1063/tcp  filtered kyoceranetdev  
2910/tcp  filtered tdaccess  
9100/tcp  open  jetdirect  
  
Nmap done: 1 IP address (1 host up) scanned in 107.23 seconds  
root@kali:~#
```

FIGURE 6.8 Nmap scan results.

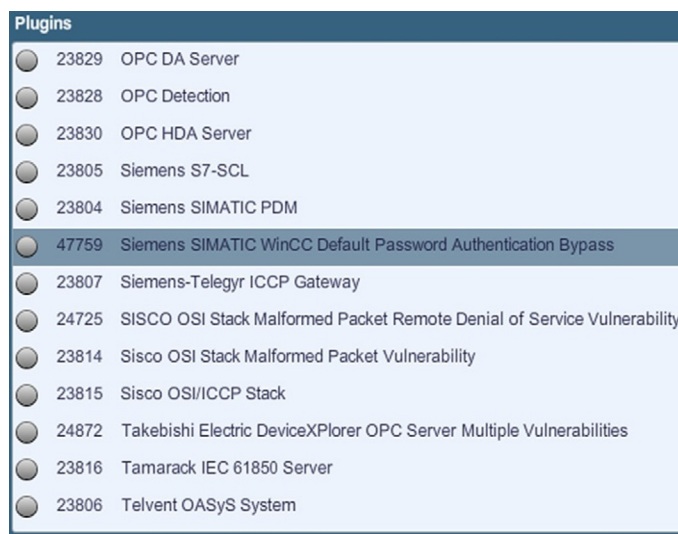
This will execute a scan of our IP range while conducting OS fingerprinting and version detection against the 1000 most common ports.

The examples above only just scratch the surface of nmap's capabilities. There are many more switches that enable various features and functionality, without even getting beyond the standard portions of the tool. In addition to this, we can use the Nmap Scripting Engine (NSE) to extend the functionality of nmap to do other interesting things. The author of nmap, Fyodor, has written an excellent book on the wide variety of things that we can make this tool do called *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*—ISBN-13: 978-0979958717. This is a highly recommended reference for those that use nmap frequently.

Nessus

Nessus is primarily a vulnerability scanning tool, but, as we discussed with nmap, a variety of other features have crept in over the years in order to add to its utility. Nessus was, once upon a time, an entirely free and open source tool. In 2005, Nessus was changed to a closed source license, and certain features were restricted to the commercial version. A free version is still available, but is limited in the circumstances under which it may be used and the vulnerability listing that it is allowed to access. An alternative open source solution has been created, which we will discuss later in this section.

Nessus classifies vulnerabilities into sets of plugins, with each family of plugins focusing on a particular type of vulnerability. These families include a variety of different operating systems, databases, protocols, and services. The professional plugin feed includes swift



Plugins		
23829	OPC DA Server	
23828	OPC Detection	
23830	OPC HDA Server	
23805	Siemens S7-SCL	
23804	Siemens SIMATIC PDM	
47759	Siemens SIMATIC WinCC Default Password Authentication Bypass	
23807	Siemens-Telegyr ICCP Gateway	
24725	SISCO OSI Stack Malformed Packet Remote Denial of Service Vulnerability	
23814	Sisco OSI Stack Malformed Packet Vulnerability	
23815	Sisco OSI/ICCP Stack	
24872	Takebishi Electric DeviceXPlorer OPC Server Multiple Vulnerabilities	
23816	Tamarack IEC 61850 Server	
23806	Telvent OASyS System	

FIGURE 6.9 Nessus SCADA plugins.

access to the newest plugins, and some reserved categories of plugins as well, such as those for detecting vulnerabilities in Supervisory Control and Data Acquisition (SCADA) systems, as shown in [Figure 6.9](#).

NOTE

As of the time of writing, the full impact and activities of the Stuxnet worm are still being discovered. For the latest information on Stuxnet, check into the most recent documentation from any of the major antivirus vendors.

The easiest way to use Nessus, due to the complexity of the product, is through the GUI. Although earlier versions of the Nessus client featured a self-contained client, the current version, as of this writing, is accessible through a web browser. In [Figure 6.10](#), we can see a partially completed Nessus scan, showing the machines located; the current state of completion for the scan of each device; the number of vulnerabilities in the high, medium, and low categories; and the number of open ports on each device.

Drilling down into a specific device, as shown in [Figure 6.11](#), we can then see the information for the specific ports found, the services in use on these ports, the count of vulnerabilities related to the particular service, again segmented into high, medium, and low categories of risk. At this level, we can start to get a better idea of how a given machine might be vulnerable to attack, and start to formulate a more specific strategy for attacking it.

From here we can step down to the listing of all of the vulnerabilities on a given service for the device in question. This will give us yet another level of specificity for where the gaps in security might be, but the truly interesting bits are one level further down still, in the specific vulnerability detail, as shown in [Figure 6.12](#). This detailed listing will give us a specific

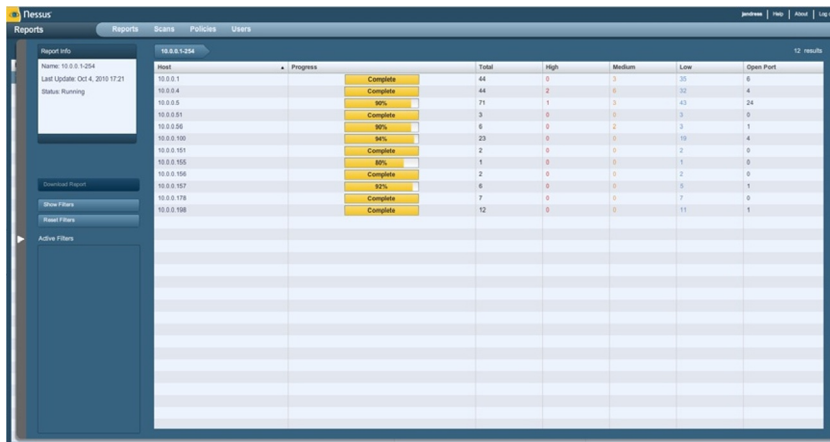


FIGURE 6.10 Nessus scan in progress.

Port	Protocol	SVC Name	Total	High
0	udp	general	1	0
0	tcp	general	7	0
0	icmp	general	1	0
137	udp	netbios-ns	1	0
139	tcp	smb	2	0
445	tcp	cifs	14	2
1900	udp	upnp-client	1	0
3689	tcp	www	3	0
5353	udp	mdns	1	0

FIGURE 6.11 Nessus port scan results.

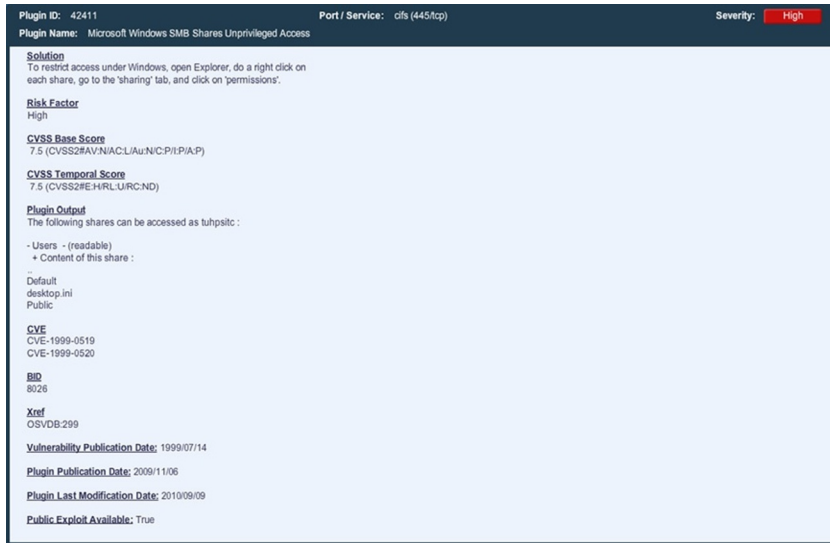


FIGURE 6.12 Nessus vulnerability listing.

description of how the vulnerability might be used to attack the device, as well as references to other possible sources of information.

If we wish to extend or automate the functionality of Nessus, the Nessus Attack Scripting Language (NASL) enables us to do so. This also allows us good access to run Nessus through the command line. For those interested in NASL and more in-depth coverage on Nessus in general, an excellent book on the topic is *Nessus Network Auditing, Second Edition* (ISBN: 978-1-59749-208-9, Syngress) by Russ Rogers.

For those looking for an open source alternative to Nessus, there is a Nessus variant called the Open Vulnerability Assessment System (OpenVAS). OpenVAS is a fork of Nessus from when the product was open source, thus sharing many of its characteristics. OpenVAS is largely compatible with the standard Nessus plugins, as well as being able to use custom plugins written in NASL. OpenVAS also offers its own plugin feed to the public, containing many of the same or similar plugins that are available from the Nessus plugin feed. Comparison tests have been done between Nessus and OpenVAS which, although declaring Nessus to be the superior product, noted that OpenVAS still performed well and was a reasonable alternative [4].

Defense

Protecting information from scanners can be a difficult prospect. If a scanner is positioned in such a way as to have network access, or be able to eavesdrop on network traffic, particularly if the target is exposed to the Internet, then we are likely vulnerable to scanning attacks. A common maxim in martial arts is that “the best defense is to not be there” [5]. This concept directly applies to preventing information leakage to scanners. In our case, not being there means not sending traffic out in ways that it is easily visible to unauthorized listeners, not running services on standard ports, not sending unencrypted traffic, and any of a number of similar hardening measures.

Many scanning tools depend on services existing on common ports and open access to information to generate their reports. In many cases, until a version scan has been attempted, scanning tools will report a service to be running based on the associated port being open. For example, if the scanner finds a port open on 21, it will generally assume that the service behind it is FTP. Changing these basic parameters in an environment can very quickly invalidate the information being returned by a scanning tool and can force the attacker to put quite a bit more time and effort into discovering what exactly is running on a given device.

ACCESS AND ESCALATION TOOLS

A great number of the hacking and penetration testing tools available, both open source and commercial, are focused on gaining access to systems and escalating our level of privilege once we are able to access the system. There are far too many tools for us to discuss any number of them individually, so we will cover some of the more common and more popular tools in this section.

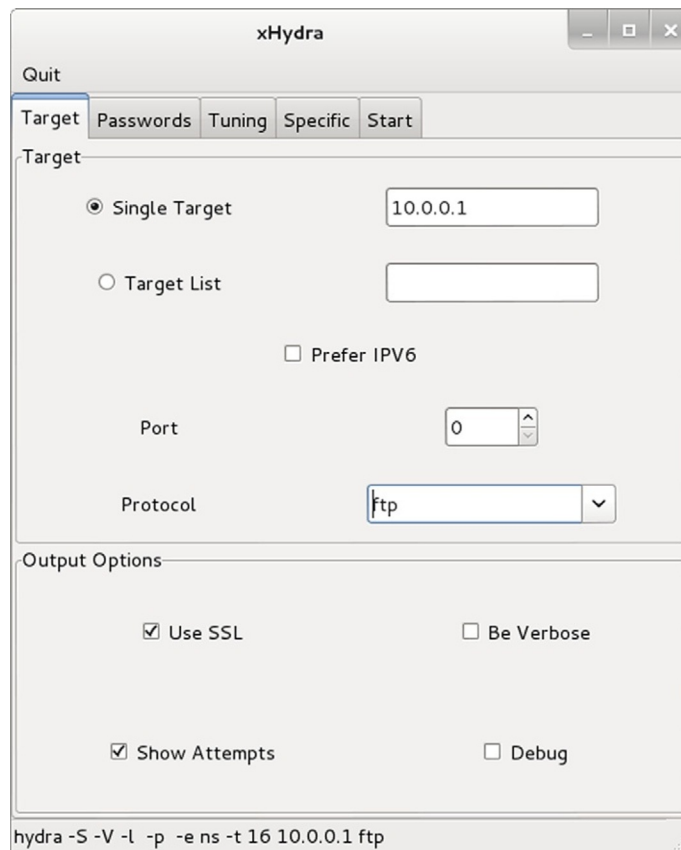
Password Tools

As poorly constructed passwords are all too common, password attack tools are a good place to start when attempting to access a machine or get into an account with a higher level of privilege. Two of the most common tools used when conducting such attacks are Hydra and John the Ripper.

Hydra^b is a tool for conducting password guessing over a variety of services and protocols. Hydra can run on a variety of operating systems and from the command line or GUI, as shown in Figure 6.13.

Hydra can be used with single usernames and passwords, or can work from lists of either or both. Given a weak password policy on the target system, and a reasonable password list to work from, we stand a reasonable chance of guessing the password for an account, given enough time to do so. With a bit of searching, we can find password lists containing default

FIGURE 6.13 Hydra GUI.



^b<http://freeworld.thc.org/>

passwords for a variety of hardware devices,^c or common passwords in a number of languages.^d

John the Ripper^e takes a slightly different approach to attacking passwords. Instead of guessing passwords from a list, as Hydra does, it takes the encrypted form of the password, commonly referred to as a password hash, and attempts to recover the password from this. Password hashes are mathematical functions that are, when properly implemented, generally considered impossible to reverse. We can work around this, when we know what hashing algorithm has been used, by using the known algorithm to hash a variety of guesses as to what the password might be, until we find a matching hash. At this point we now know what the password represented by the hash is. John the Ripper, commonly known as John, can perform this exercise with password hashes from many operating systems and can run on a variety of operating systems as well.

The Metasploit Project

The Metasploit Project^f is a well-known collection of open source security tools, launched by HD Moore in 2003. In 2009, Metasploit was acquired by Rapid7, and now enjoys greatly increased funding for development. This has led to Metasploit branching out into more fully featured commercial versions, in addition to continued development on the original free tools. Metasploit is, at the time of this writing, available in three main versions: the free Metasploit Framework and the commercial Metasploit versions, Express and Pro.

The Metasploit Framework

The Metasploit Framework is the free offering of Metasploit and, prior to the Rapid7 acquisition, was the only version available. Framework is primarily intended to be used as a command line tool. Although there is a rudimentary GUI available, it does not offer access to the full might of the toolset that is available from the command line, as shown in Figure 6.14.

The process of using Framework to attack a system is, in broad strokes:

- Collect information about the target system using scanning and vulnerability assessment tools, such as nmap and Nessus
- Select an exploit that matches the system based on the collected information
- Select a payload to accompany the exploit, often a remote shell
- Execute the exploit and payload

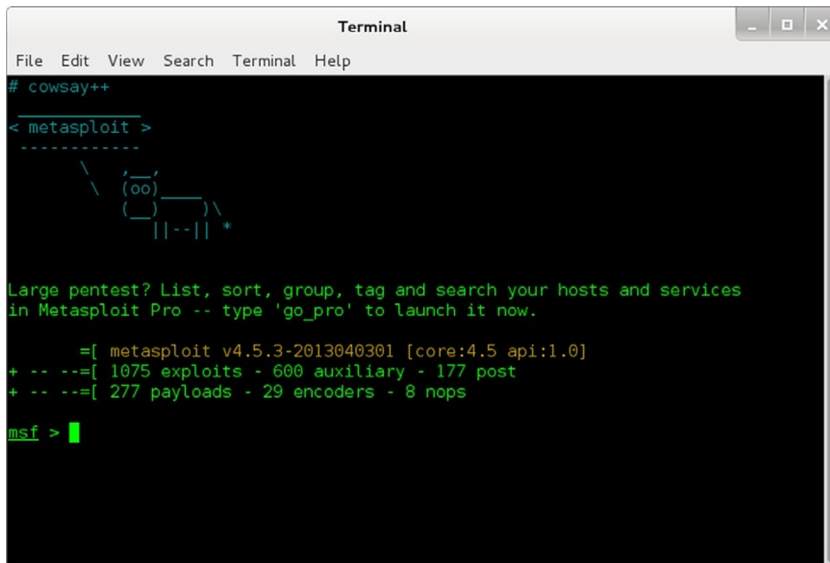
Framework offers, at the time of this writing over 600 exploits with over 200 payloads that can be used in conjunction with them. Framework also supports more advance attacks, such as proxy pivoting, communication with other tools, such as Nessus, via Extensible Markup

^c<http://www.phenoelit-us.org/dpl/dpl.html>

^d<http://www.cyberwarzone.com/cyberwarfare/password-cracking-mega-collection-password-cracking-word-lists>

^e<http://www.openwall.com/john/>

^f<http://www.metasploit.com/>



```
Terminal
File Edit View Search Terminal Help
# cowsay++
< metasploit >
-----
      \      (oo)_____)
       \      (__)_____)
        \      ||--|| *

Large pentest? List, sort, group, tag and search your hosts and services
in Metasploit Pro -- type 'go_pro' to launch it now.

      =[ metasploit v4.5.3-2013040301 [core:4.5 api:1.0]
+ -- --=[ 1075 exploits - 600 auxiliary - 177 post
+ -- --=[ 277 payloads - 29 encoders - 8 nops

msf > █
```

FIGURE 6.14 Metasploit framework.

Language Remote Procedure Call (XML-RPC), and extensibility through the Ruby language, which the current version of Metasploit is developed in. Framework has a truly massive set of functionality, much of it contributed by the security community, and is an extremely versatile tool.

Metasploit Express and Metasploit Pro

In 2010, we saw the arrival of commercial Metasploit offerings, which are, of course, not free. Metasploit Express, the first released commercial Metasploit version, contains all of the functionality of Framework, but adds a number of new features. One of the most immediately apparent features in Metasploit Express and Pro is the implementation of a fully featured GUI, as shown in [Figure 6.15](#).

In addition to the functionality that is provided by Framework, Express adds a number of features designed to automate and ease the use of Metasploit in larger attack or penetration testing environments. Express includes automation for network discovery, attacks against accounts, and the use of exploits. Additionally, for use in team environments, Express adds workflow features, evidence collection and audit, and improved reporting tools. Being a commercial tool, support arrangements are also available for users of the tool.

Metasploit Pro is the latest addition, to the Metasploit family. Pro has all of the features and functionality of Framework and Express, with the addition of the ability to do Virtual Private Network (VPN) pivoting and web server scanning and exploitation, as well as additional features that allow better team collaboration and reporting.

The leap in functionality between Express and Pro is really oriented at those that would use the tool in larger environment where multiple people or teams of people are attacking closely

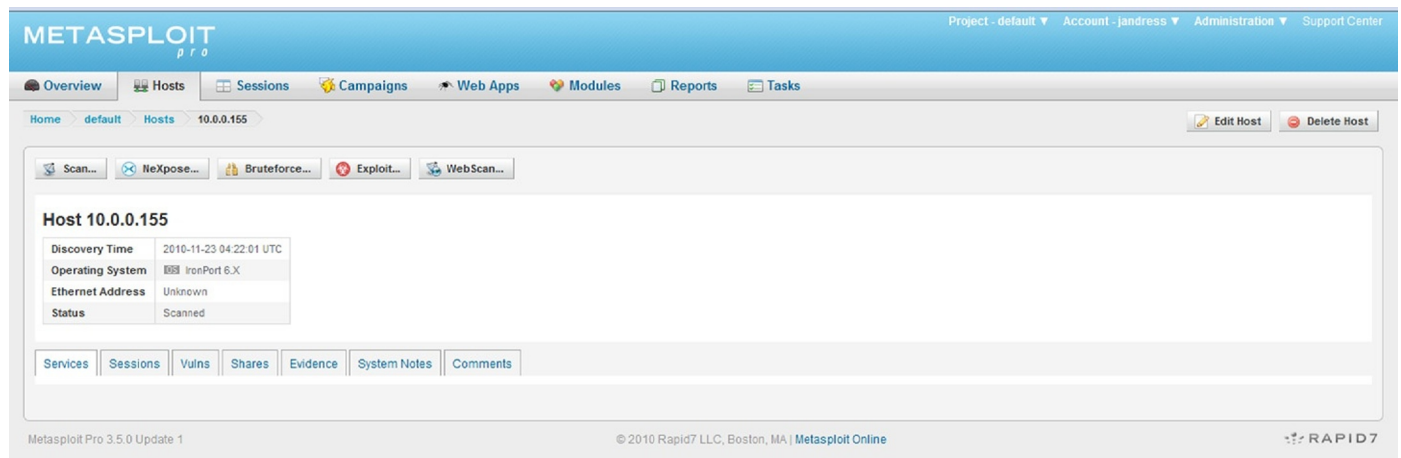


FIGURE 6.15 Metasploit Pro.

related targets. One of the gaps in the Metasploit tools has always been what to do with the information and access once it is gained, and Pro addresses this issue directly.

Immunity CANVAS

CANVAS Professional⁸ from Immunity is a tool that enables access and exploitation of systems in a semi-automated or automated fashion. CANVAS contains a good selection of exploits and payloads (around 300 exploits at the time of this writing), as well as a number of exploit packages from third parties that allow access to truly bleeding edge exploits. Although some might point out that CANVAS does not have as large of a library of exploits as some of the other tools, it does tend to be updated very quickly to include some of the newer and more interesting exploits soon after, or in some cases before, they are publicly released.

CANVAS is developed in Python, and includes both GUI, as shown in Figure 6.16, and command line interfaces. In addition to some of the other functionality that we expect from this class of tools, such as pivoting (Immunity calls it bouncing), network scanning, client side attack tools, and other functionality, CANVAS also includes some more unique features.

One such feature that some might find convenient is the geolocation and mapping feature, allowing target systems to be displayed on a world map within the interface. Additionally, CANVAS has several areas in which the GUI can be used to display graphical

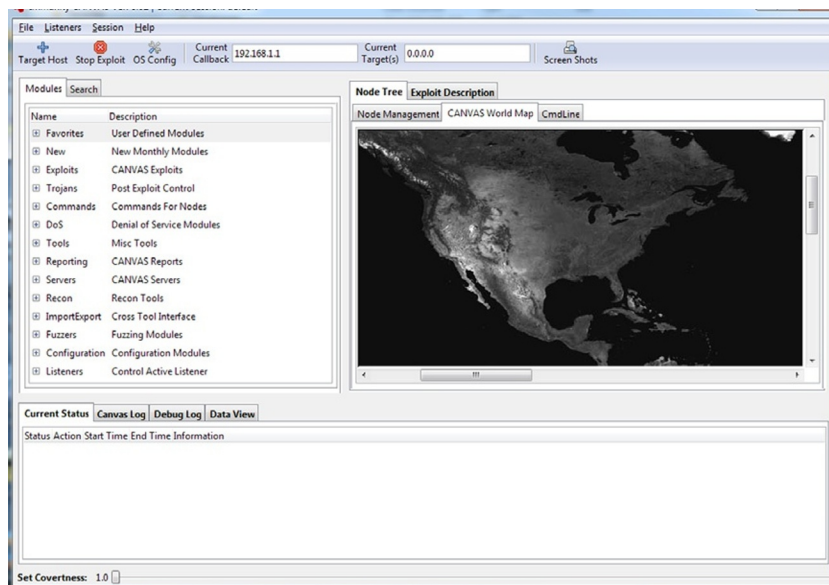


FIGURE 6.16 Immunity CANVAS.

⁸<http://www.immunitysec.com/products-canvas.shtml>

tools, such as VisualSploit for graphically building exploits, or GUI tools that can be used to explore remote file systems. Being written in Python, CANVAS is also a highly configurable tool and can be tweaked by the savvy user. Immunity prides themselves on producing a tool that is intended for use by experienced security professionals. It is extremely versatile, but like any good tool, has some sharp edges that the inexperienced wielder should be wary of.

Defense

Defenses against access and escalation tools largely revolve around well-written and implemented password policy, patching, and system hardening. All are common and well-known security techniques, and are, in theory, some of the most basic security measures that we can put in place when securing our environment, but they are not as ubiquitously implemented as we might think.

Protection against password guessing and cracking tools largely revolves around ensuring that we have strong passwords in place. The common standard for strong passwords is: minimum length of eight characters, at least one uppercase character, at least one lowercase character, at least one number, and at least one symbol. Although this may seem excessive to some, we can see the difference in using such a password versus a more simple password quite easily.

An eight-character password using only lowercase and uppercase characters has 200 billion possible combinations. Given a reasonably powerful workstation (100,000,000 guesses per second), we could brute force our way through all of the possible combinations in around 30 min. Using the stronger password scheme that we specified above (uppercase, lowercase, numbers, symbols), our eight-character password has 7.2 quadrillion combinations and would take a little over 2 years to brute force [6]. Increasing the password length and adding additional character sets continues this trend, and can quickly make password guessing or cracking infeasible entirely, even for very powerful or distributed cracking tools.

Another key step to take, particularly in the case of defending against tools such as Metasploit or CANVAS, is to ensure that our systems are quickly patched. Many such tools can penetrate a system in a few seconds given unpatched vulnerabilities with which to work, and this is an easily avoidable situation. We can argue that installing application and operating system patches immediately after they are released is foolhardy and that we may cause more problems than we will fix, and this is likely true. We should absolutely take the time to test patches before we apply them, with exceptions to this being very few and far between. It is likely true that the exploits with which attackers gain entry to our systems will be older and more common, rather than cutting edge, but we should be patching for everything that we reasonably can, as soon as we can.

Lastly, we should harden our systems as much as we reasonably can and still allow them to execute their functions. The more ports, services, accounts, and so on that we leave enabled on a system, the larger attack surface that we present to those that would seek to compromise it. In many cases individual systems have very few tasks that require leaving outside access open, either incoming or outgoing, and closing down such potential methods of access greatly limits the set of tools that we leave for an attacker to utilize.

EXFILTRATION TOOLS

Exfiltrating data from an environment can be an interesting and challenging problem, particularly if the environment in question is secured against exactly the activities that we are attempting to carry out. In broad strokes, some of the main methods that we can use to exfiltrate data are to physically carry it out, to use steganography or encryption to disguise the data, to make use of common protocols that are normally allowed to leave the environment, or to use out of band methods.

Physical Exfiltration

Physical removal of data is one of the methods most proof against detection, even in the most carefully guarded environments. The shrinking size of storage media makes such methods even more easily hidden on a person or in equipment, with the latest, at the time of this writing, standard for microSDXC memory cards topping out at a theoretical limit of 2 TB of storage and with dimensions of 11×15 mm at 1 mm thick, roughly the size of a fingernail [7]. Given the ability to store such amounts of data in a package so small, items to be exfiltrated could be secreted nearly anywhere and are beyond the reasonable realm of detection, even in the case of an extensive physical search.

Encryption and Steganography

Various tools exist for hiding data in formats that are not immediately visible to casual search, or in some cases, even to exhaustive search. Encryption tools in general can be useful for hiding data in such a fashion, rendering the data with which we are concerned potentially invisible, or at least unreadable. Certain encryption tools, TrueCrypt^h for instance, can create encrypted volumes on storage media which appear to be random noise on the media and are neither detectable nor recoverable without the proper keys or passwords. Such a volume could easily be created on portable storage media, such as a flash drive or MP3 player and would appear to be empty space.

Steganography is the science of creating messages that are hidden from those that do not already know that they exist. Such methods have existed from time immemorial, using special inks, works of art, and numerous other methods, but the age of computers has provided us with a far more suitable and information dense media in the mass of information that flow around the globe on a daily basis. Files which contain a certain amount of noise, such as graphic, video, or audio files can be used to encode information within them without altering the presentation of the file contents to the point of being detectable to the naked eye.

Steganography tools such as OpenPuffⁱ or OutGuess^j can make secreting such data in digital files a relatively simple task. Once hidden in such a file, our data can be exfiltrated by placing an image on an externally facing web server, in a background graphic or logo attached to an email, or even in an audio message transmitted over a Voice over IP (VoIP) connection.

^h<http://www.truecrypt.org/>

ⁱhttp://embeddedsw.net/OpenPuff_Steganography_Home.html

^j<http://www.outguess.org/info.php>

Using Common Protocols

Even in highly secure environments, there are likely to be a few protocols, perhaps closely monitored, that are allowed to leave the environment. We can generally find Hypertext Transfer Protocol (HTTP) and the various email protocols to be allowed some degree of freedom, as well as protocols that are more infrastructure related such as Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP). We can simply, in the case of mail protocols, ship our information out in encoded and in small pieces, or if need be, we can tunnel over various protocols, using tools such as OzymanDNS.^k

In some cases where the use of our favored protocols is prohibited, we can even create a tunnel to move Secure Shell (SSH) over HTTP using utilities such as Corkscrew,^l and utilize the provided proxy server to exfiltrate our data. Given an unmolested SSH connection to the outside world, we can accomplish a great number of tasks, including exfiltrating our data.

Out of Band Methods

We can use a number of methods that step outside of the purview of the security and detection mechanisms that are put in place in order to prevent the leakage or deliberate exfiltration of data. In the case of application or host level security that would use any number of technical controls in order to prevent data from leaving the system in an unauthorized fashion, we need merely to move to methods that such systems are not capable of detecting or controlling. For such systems, we can hand copy data onto paper, take pictures of the information on the display, memorize the data for later retrieval, or any number of similar methods. Such methods can be very simple or very complex, depending on the density of the information that needs to be communicated, and could be as simple as leaving a light turned on during the day or closing the blinds in a window. Such methods are highly effective and can be extremely difficult to detect when properly executed.

Defense

Defending against exfiltration of information can be a very difficult task, depending largely on the inherent security posture of the environment. Preventing data exfiltration other than in the most egregious cases can be all but impossible, especially in a standard corporate setting, where personnel are able to move freely in and out of the environment, in both a physical and a logical sense, at will and are not prohibited from bringing personal electronics devices into the environment and not searched when entering or leaving the premises. In such environments, there are so many avenues, both physical and logical, that could be used to move data out that we will never be able to protect them all without making major changes.

The answer to this issue is to move to a more secure footing, such as what we would find in the environments used by many militaries and governments. In such environments, the

^k<http://dankaminsky.com/2004/07/29/51/>

^l<http://www.agroman.net/corkscrew/>

activities of personnel, both physical and logical, are very restricted and closely monitored. Additionally, personnel are often much more tightly screened before being allowed access to the environment at all, often in the form of extensive background check and security clearances. In any case, if a determined attacker is able to penetrate the environment in a physical or logical fashion and is sufficiently patient and persistent, they will likely find a way eventually to exfiltrate the data that they are interested in removing.

SUSTAINMENT TOOLS

Once we have gained access to a system and reached the desired level of access, we will likely want to ensure that we can continue to access the system in the future. Although we may have been able to successfully use a particular vulnerability or similar means to access the system in the first place, we cannot necessarily depend on the same hole to still exist in the future.

Adding “Authorized” Access

One of the simplest and, at times, most effective means of securing our access to a system is to add ourselves to the list of users that is legitimately allowed access. This is typically accomplished with built-in operating system commands such as `useradd` on Unix-like systems and the `netuser` command on Windows systems. In addition to adding simple users, we can also create additional access to applications, networks, and any number of other systems in the environment. Although such access may eventually be audited and removed in many environments that do not operate on an enhanced security posture this may not happen for several years, if ever.

We can see an example of such a tactic in the TJX breach that occurred in 2006. Once the TJX systems were penetrated, the attackers were able to install accounts on Internet accessible applications in order to access the information that they wished to obtain [8]. At this point, the vulnerabilities that originally allowed the attack to be successful were no longer a weak point in maintaining access to the environment, as they were then able to enter through the virtual front door of the system.

Backdoors

Adding backdoors to a system or application is another method that we can use in order to sustain our access. A great variety of such backdoors exist for any number of applications, and an attacker with a good knowledge of programming can easily create custom varieties. One useful set of web-based backdoors can be found in the Web Malware Compilation,^m which is also included in recent versions of the Backtrack/Kali Linux distributions^{no}.

^m<https://code.google.com/p/web-malware-collection/>

ⁿ<http://www.backtrack-linux.org/downloads/>

^o<http://www.kali.org/downloads/>

There are many subtle ways that we can use to create backdoors on systems, but the old standby tool netcat can perform this task for us very nicely. Versions of netcat can be found for many operating systems, and it can often be found to already exist on many Unix-like operating systems. Creating a listening port that will allow us access to a shell on the system with netcat is very simple and can be accomplished with a command on Linux such as:

```
nc -l -p 1234 -e /bin/bash
```

And we can accomplish the same on Windows with a slight tweak, like so:

```
Nc -l -p 1234 -e cmd.exe
```

In each case, we are telling netcat to listen for connections on port 1234 and to execute a program that will give the connecting client a shell. Although the listening process will be obvious to any administrator who takes the time to look for odd processes or ports being listened on, clever naming of the tool and selection of the port number can help to minimize this. Additionally, the command can be run as a scheduled job, set to run when the system boots, or a variety of other methods to ensure that the backdoor stays in place. More on backdoors using netcat can be found in *Netcat Power Tools* (ISBN: 978-1-59749-257-7, Syngress).

Defense

Defending against backdoors being inserted requires a twofold approach. We first want to make sure that successfully inserting such backdoors is difficult to begin with. We can help to mitigate such attacks by ensuring that our systems and applications are as hardened as we can reasonably make them, and that both our outgoing and incoming traffic is as restricted as we can make it and still function properly. We can also lock down administrative access to our systems through the use of utilities such as powerbroker and Cisco Security Agent (CSA). These will help to prevent the insertion of backdoors and make a considerably more difficult task for those that are attempting to attach to them.

The second portion of defending against backdoor attacks is auditing. If we carefully audit accounts, system access, open ports, and other items that could be used to create a backdoor, we at least stand a chance of quickly catching anything that has been put in place. Unfortunately, this type of auditing is a time-consuming and thankless task, and so is not commonly implemented. In many environments, a subtly implemented backdoor many never be found, largely due to lack of anyone looking.

ASSAULT TOOLS

The tools that can be used to assault a compromised machine are many and varied. They can take the form of simple changes to configurations or environment variables on a system, to purpose-built botnets that can conduct a concentrated Denial of Service (DoS) attack on a given system or environment. Such tools of destruction can generally be categorized into those related to software or oriented on hardware.

Meddling with Software

Most software is not built to withstand deliberate tampering by authorized users, as such users are generally more interested in it functioning properly than in causing it to fail. Additionally, by the point that we have decided to use such tools, we have likely compromised the target machine already and have administrative rights, which allow such tampering to take place regardless of the software vendors wishes. Even in the cases where we might not have such rights, there are often still steps that we can take.

System Resources

System resources can often be affected, even by unprivileged users. Although such measures will be immediately obvious to anyone investigating the subsequent issues, such users can start a sufficient number of long-running processes as to use large amounts of system resources such as memory, CPU, and hard disk, thus preventing other processes from being able to access them. Any number of simple commands can be used to create such resource drains. In order to quickly fill a file system on a Unix-like operating system to the point of nonfunctionality we can use a command such as:

```
cat /dev/zero>file
```

This command will attempt to place the contents of `/dev/zero`, a never ending stream of zeros, into the file called `file`. Based on experimentation by the authors on an average system, this command will produce 4 GB of zeros in a little over a minute. Depending on where it is run in the file system and how much free space exists, this can bring a system to its virtual knees in a few minutes. Similar commands can be used to highly utilize the CPU and memory, and such tactics can also be used on Microsoft operating systems.

System Environment

Altering the system environment can also be used to throw a wrench in the works of many environments. Many applications, particularly in more complex cases, depend on a delicate balance of environment and operating system settings. Interfering with these settings can have a variety of deleterious effects on said software.

One such setting that can wreak havoc with systems in a variety of ways is to alter the way in which the system calculates time. Various tools depend on the system time being both correct and consistent over a period of time. When the system time is altered in either direction, sped up or slowed down, the system date is changed, or the time zone is incorrect for the location of the system, a multitude of effects that are generally undesirable to the system owners can ensue. For example, we could use the following:

```
tzutil/s "Ulaanbaatar Standard Time"
```

This command will change the time zone on a Windows system to that of the capital of Mongolia. This, for many countries, would change the system time considerably, and perhaps even the date. Changing such settings repeatedly would skew timestamps in logs, send times on email, entries in calendars, and any number of other places in which timestamps are utilized. This is a small change, but can have far-reaching effects.

There are a multitude of similar small changes that we can make. Another example is to change the `umask` setting on a database server. This command can change the permissions on newly created files. If the permissions are not exactly right on files that the database creates and uses, it will fail. It can be a rather difficult proposition to figure out what exactly has happened.

Across many operating systems, environment variables are used to hold a variety of information critical to keep the systems in working order. Environment variables hold information pointing to locations in the file system where various utilities are stored, where library files can be found, aliases to commands, and a multitude of other bits and pieces. Altering environment variables, depending on the variable in question, can very specifically effect an individual application, or can bring the entire operating system to a screeching halt. In addition, we have many other similar settings, such as those that alter the functionality of the operating system kernel, which behave similarly. If, on a Linux operating system we were to run:

```
# echo "fs.file-max=1" >> /etc/sysctl.conf
```

We would change the kernel parameter that specifies number of files that can be opened at once, across the entire operating system, to one file. Most busy systems will need to open something on the order of thousands of files at a time, so, of course, this will very quickly bring the system down once it takes effect.

Attacking Hardware

There are a variety of ways that we can attack computing and related hardware with the intent of disabling it in some fashion or other. In many items of hardware, we can find Read Only Memory (ROM) modules, consisting of electronically reprogrammable memory. Such ROMs often contain firmware that regulates how the specific piece of hardware functions or communicates with other hardware. Using somewhat universal ROM flashing tools, such as `flashrom`,^P we can rewrite the contents of such modules in order to reprogram them to alter the functionality of the hardware, or, easier yet, to disable the hardware entirely. Using `flashrom` in particular, we can flash ROMs from remote on a variety of operating systems, presuming that we have administrative access.

Another easy way to disrupt hardware, although generally on a temporary basis, is to alter the software that controls communications with it. In the sense of drivers, we can fairly easily disrupt the files of which they are composed. Usually, this will be sufficient to prevent the device from being used until the driver is reinstalled, potentially requiring physical access to the machine to do so. On a somewhat more simple level, we can alter the way that the software talks to the hardware, often through the use of configuration files. We can change the settings of videos cards in order to temporarily render displays nonfunctional, disrupt a hard disk array by changing its composition, or any number of other small changes. As we said, such changes are unlikely to have any long-term effects on the hardware itself, but may have profound effects on the systems that depend on the availability of that hardware.

^P<http://www.flashrom.org/Flashrom>

One of the most impactful ways that we can presently hold up as an example for potential outright hardware damage is in interfering with SCADA systems. Much supposition has been done about the potential for damage to such systems, and there are a few examples. An excellent example, discovered in July of 2010, occurrence was shown to exist in the Stuxnet worm. In addition to the other effects of Stuxnet, which we will discuss at greater length in [Chapter 8](#), Stuxnet appears to interfere with the frequency of the motors that are used in the gas centrifuges used to enrich uranium. Not only can this impede the uranium enrichment process, but it can also potentially cause the centrifuge to catastrophically fly apart [9]. Clearly, this would be less than optimal, considering the environment in question.

Defense

Defending against software and hardware manipulation is a difficult prospect. Once an attacker has administrative rights on a machine, there is little that we can do to prevent them from taking such steps. Conversely, if an attacker does not have administrative rights on a machine, they are generally prevented from taking such measures. In short, the defense against such actions largely revolves around preventing attackers from gaining administrative rights on the system, a task often involving system hardening and including many of the methods that we have covered in the various defense sections in this chapter.

OBFUSCATION TOOLS

To obfuscate means to “confuse, bewilder, or stupefy”; “to make obscure or unclear”; or “to darken” [10]. This definition perfectly suits the set of tools that we might use to cover our tracks when operating on a system or in an environment. In general, there are three main types of tasks that we are concerned with in such cases: obscuring our location, manipulating logs, and manipulating files.

Location Obscuration

One of the chief concerns when conducting Computer Network Operations (CNO), which we will discuss at length in [Chapters 10–13](#), is hiding or obscuring the location, in either a logical or a physical sense, from which we are operating. Generally, this is accomplished through the use of some sort of proxy, whether this is a purposely built network specifically for doing so, or merely a compromised system through which we are operating.

The Onion Router (Tor) is a system, developed with the support of the U.S. Naval Research Laboratory [11], with the specific purpose of insuring the anonymity of communications over the Internet. Tor is used by the Navy for open source intelligence gathering, by law enforcement agencies for surveillance and intelligence gathering [12], and by a large number of organizations and individuals for various purposes where privacy and secrecy of communications are desired. Tor provides this anonymity by routing communications through several intermediary proxies, other nodes operating in the network, before the traffic reaches an endpoint and is delivered to its final destination. In practice, this makes the traffic very difficult to trace back to its origin, but, depending on the configuration of the client and the type of traffic, not impossible.

WARNING

Tor and similar proxy networks, sometimes referred to as mixed networks, are great tools for obscuring the origination of traffic and adding a layer of security and/or privacy to our activities, but they are not a magic bullet. Depending on the exact configuration of the systems involved, the traffic being sent, the source and ultimate destination, and a number of other factors, it may be possible to trace the origin of the traffic. For those interested in reading further on such issues, see the paper Low-Resource Routing Attacks against Anonymous Systems.^q

Similar proxy networks to Tor, such as Bitblinder, Perfect Dark, and I2P exist as well, and all have similar issues to one degree or another. Other measures can be taken to ensure some measure of anonymity, such as the use of VPNs, or even using one or more compromised machines as a sort of manual proxy. Such simpler measures obviously do not provide the same level of anonymization of communications, but they also do not have some of the same issues. Measures of such a simple nature can also be useful in the case where we do not necessarily desire to entirely hide the end point of an attack, but instead wish to implicate another party. This is commonly considered one of the reasons for the large number of attacks that appear to originate from China.

Log Manipulation

With nearly any activity that we might care to conduct on or against a system, we are sure to generate some sort of an entry in logs of systems and network devices. Depending on what exactly our purpose is when conducting such operations, we may wish to remove such traces in order to hide our presence from future investigators, system administrators, and the like.

On Unix-like systems, presuming that we have the proper permissions to do so, logs can often be altered through the use of a text editor. In some cases, we may find that a particularly savvy system administrator has set attributes for the logs that we are interested in in order to make them append only. Additionally, it is possible to disable the capability that would allow us, even as root, to remove such a flag from a log file. In such cases, by the time that we have discovered that these measures are in place, we have likely left a great deal of evidence behind in various logs, and may need to adopt somewhat of a scorched earth approach, as we discussed in the Assault Tools section of this chapter.

On Windows systems, the logs are stored in a somewhat more protected format, and are difficult to manipulate directly, by design. Not only is the file format resistant to tampering, but the logs are generally held in a constantly locked state by the logging processes. Any account with administrator access can clear the event logs entirely, but this is a heavy-handed tactic, and generally very obvious. Fortunately, some tools do exist that will allow us to selectively manipulate these logs, usually the Security log being our specific concern. One such tool is WinZapper,^r which enables us to easily, given administrator access, remove specific

^q<http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.4562>

^r<http://www.ntsecurity.nu/toolbox/winzapper/>

events from the log on Microsoft operating systems from Windows NT 4.0 to Windows Server 2003. For more recent operating systems, such as Windows 7/8 and Server 2008/2012, the holes that allow these types of tools to function have been patched; fortunately, other vulnerabilities, such as MS10-041, do exist and can be used to carry out log manipulation in a similar fashion on unpatched systems.

In many operating systems, writing to logs files is not restricted to the same extent as is manipulating them, and in some cases, is not restricted at all. In such cases, when we have failed to remove data from the log files in a cleaner fashion, we can simply fill the logs with our own events in order to push the log past its retention period in order to obscure our own activities. Depending on how the logging mechanisms in question are configured, we may simply find that the log entries are overwritten past a certain time or size specified, as is common in many Windows implementations, or we may find that the logger rolls to a new file and that the older log is renamed and stored. In such cases, writing authentic looking events to the log, such as the repeated events that we might find from a hardware or software failure of some sort, or a replay of older events, may more easily escape notice than just writing garbage to the logs.

File Manipulation

When attempting to hide files on systems in which we are operating, there are a variety of approaches that we might take. We can simply use the built in commands of the file systems in order to hide files, which may work to a certain extent, for casual users. We can also rename our files to something obscure which matches the system files of the operating system on which we are operating and hide them in the midst of similar files, which will likely enjoy some measure of success. On Microsoft OSs using the NTFS file system, we can place data in Alternate Data Streams (ADS). ADS are storage areas in a file that are typically intended to store metadata, such as thumbnails for image files. Using tools such as streams,^s we can easily access ADS and insert information that will be invisible to those that are not specifically looking for it.

WARNING

Using rootkits, and malware in general, is a form of attack that should be used very carefully. Even in the case of custom malware, such tools may behave in unexpected ways outside of our testing environment.

As somewhat of a final measure, we can also use rootkits to hide our files, which will be a nearly impenetrable method, as long as we still have control of the operating system itself. Given a kernel mode rootkit, we can prevent nearly any tool or utility from finding our files through the simple expedient of telling such tools that our files do not exist.

^s<http://technet.microsoft.com/en-us/sysinternals/bb897440.aspx>

In the course of manipulating various files on the systems in which we are operating, we will have likely modified the timestamps of said files in the process of doing so. Particularly with files on which the timestamps are set to a commonly known value and not frequently changed, such as the files that comprise portions of an operating system, our efforts may cause these altered timestamps to clearly stand out to someone looking for the signs of compromise. Fortunately, such timestamps, again given appropriate administrative permissions, are relatively easily reset. On Unix-like systems, timestamps can be reset with the `touch` command. On Windows systems, timestamps can be reset with utilities such as `Timestomp`^t or through the use of PowerShell commands like the following [13]:

```
$(Get-Item).creationtime=$(Get-Date "mm/dd/yyyy hh:mmam/pm")
$(Get-Item).lastaccesstime=$(Get-Date "mm/dd/yyyy hh:mmam/pm")
$(Get-Item).lastwritetime=$(Get-Date "mm/dd/yyyy hh:mmam/pm")
```

Given sufficient attention by a skilled forensic investigator, it may be possible to eventually discover some traces of such file manipulation; however, our primary concern here is to keep from attracting such attention in the first place. If we are sufficiently diligent in our obfuscatory efforts, traces of our activities should be very difficult to find and should be all but invisible to casual users of the environments concerned and to the administrators as well.

Defense

Defending against obfuscation measures can be very difficult or very easy, depending on where exactly the manipulation is taking place. When dealing with the tactics that an attacker can use to obscure their location, the countermeasures that can be taken to reverse such efforts can fall solidly into the very difficult category. Although we can take steps to attempt to work our way backwards through the proxies and perhaps other steps that an attacker has taken, this is very much a manual process and would require the cooperation of the owners of each intervening layer, and would be rather unlikely to bear fruit. This being said, with the resources of a nation-state to back up such an investigation, and placing sufficient importance on recovering the information, it is not impossible that we could do so successfully. In any case, such tactics are almost entirely reactive in nature and would be carried out in the aftermath of the incident that prompted them.

In the case of file and log manipulation, there are many defensive measure that can be put in place to ensure that the efforts of our attacker are unsuccessful. We can use tools such as `Tripwire`^u to monitor for file manipulation in real time and issue alerts when something untoward takes place. We can also send copies of our log entries to remote servers that are hardened against attack. These two measures will go a great deal of the way to ensuring that if such attacks do occur, they will not go unnoticed.

^t<http://www.offensive-security.com/metasploit-unleashed/Timestomp>

^u<http://www.tripwire.com>

SUMMARY

In this chapter we discussed the various tools that we might use in conducting cyber warfare, and the methods that we might use to defend against an attacker using them.

We discussed the tools that we might use for reconnaissance, for activities including: general information gathering, searching whois and DNS records, and metadata from media and documents. We covered scanning tools, such as Nmap and Nessus, that we might use to find systems and detect potential areas where vulnerabilities might exist. We went over access and privilege escalation tools, such as Metasploit and CANVAS, that we might use to gain entry to a system and work our way into accounts with greater levels of access to the system. We talked about exfiltration methods, using tools to encrypt, hide, or smuggle data over common protocols in order to remove it from a compromised system. We looked at means that we might use to sustain our connection to a compromised system, such as adding backdoors or additional access, so that we can still operate on the system if our original method of access is removed. We went over assault tools, which we might use to damage or disrupt systems that we have compromised, often using common operating system utilities. Lastly, we discussed obfuscation tools that we might use to hide our location, in both the logical and physical sense, while we are attacking.

The majority of the tools that we discussed in this chapter are free, or have free versions, and are available to the general public. It is important to realize that the tools that are required to conduct pure cyber warfare are freely available, unlike many of the tools that are required to conduct conventional warfare on any large scale. This easy access to such tools means that nation-states may find themselves facing enemies that are fully capable of causing severe damage to computers and the systems to which they are attached.

References

- [1] Shalal-Esa A. Six U.S. Air Force cyber capabilities designated “weapons”. Reuters, <http://www.reuters.com/article/2013/04/09/net-us-cyber-airforce-weapons-idUSBRE93801B20130409>; 2012 [accessed 05.22.13].
- [2] Aitoro J. IRS finds unauthorized web servers connected to its networks. Nextgov.com, http://www.nextgov.com/nextgov/ng_20080904_3324.php; 2008 [accessed 04.09.13].
- [3] Offensive Security. Google Hacking Database. Exploit Database, <http://www.exploit-db.com/google-dorks/?function=detail&id=795>; 2013 [accessed 04.09.13].
- [4] Laboratory for Systems and Signals. Nessus/OpenVAS comparison test. s.l.: faculty of electrical engineering and computing, University of Zagreb, http://security.lss.hr/images/stories/documents/Nessus_vs_OpenVAS_en.pdf; 2009 [accessed 03.24.13].
- [5] Morris D. Ninja Billy Fun Page. Ninjabilly.com, <http://www.ninjabilly.com/fun.html>; 2009 [accessed 03.24.13].
- [6] Lucas I. Password recovery speeds. Lockdown.co.uk, <http://www.lockdown.co.uk/?pg=combi>; 2009 [accessed 03.24.13].
- [7] SD Association. SDXC. SD Association, <https://www.sdcard.org/developers/overview/capacity/>; 2010 [accessed 03.24.13].
- [8] Vijayan J. TJX violated nine of 12 PCI controls at time of breach, court filings say. Computerworld, http://www.computerworld.com/s/article/9044321/TJX_violated_nine_of_12_PCI_controls_at_time_of_breach_court_filings_say; 2007 [accessed 03.24.13].
- [9] Keizer G. New Stuxnet clues suggest sabotage of Iran’s uranium enrichment program, http://www.computerworld.com/s/article/9196458/New_Stuxnet_clues_suggest_sabotage_of_Iran_s_uranium_enrichment_program?taxonomyId=17&pageNumber=1; 2010 [accessed 03.24.13].

- [10] Dictionary.com. Obfuscate. Dictionary.com, <http://dictionary.reference.com/browse/obfuscate>; 2013 [accessed 03.24.13].
- [11] U.S. Naval Research lab. onion routing, <http://www.onion-router.net/>; 2010 [accessed 03.24.13].
- [12] The Tor Project, inc. Tor: overview. Tor project, <http://www.torproject.org/about/overview.html.en>; 2010 [accessed 03.24.13].
- [13] Hull D. Touch on Windows via PowerShell. trustedsignal. [blogspot.com, http://trustedsignal.blogspot.com/2008/08/touch-on-windows-via-powershell.html](http://trustedsignal.blogspot.com/2008/08/touch-on-windows-via-powershell.html); 2008 [accessed 03.24.13].
- [14] Mills E. Stuxnet: fact vs. theory. CNET news, http://news.cnet.com/8301-27080_3-20018530-245.html; 2010 [accessed 03.24.13].