About Face: Defending Your Organization Against Penetration Testing Teams

Terrence OConnor

About Face: Defending Your Organization Against Penetration Testing Teams

GIAC (GSEC) Gold Certification

Author: TJ OConnor, terrence.oconnor@usma.edu
Advisor: David Shinberg

Accepted: December 1, 2010

Abstract

In this paper, we examine a series of techniques that a network administrator can use to frustrate external penetration testers while securing and obscuring the network under attack. We examine a deeper understanding of the penetration testing cycle performed by the team and the weaknesses that can be exposed in the cycle to survive a test. Throughout the paper, we document and enumerate several tools and scripts that can be of great value in surviving a penetration test.

1. Introduction

In the following paper, we outline several methods for obscuring your network from attack during an external penetration test. Understanding how a penetration testing team performs a test and the tools in their arsenal is essential to defense. The penetration testing cycle in the next section. Following that, we discuss defeating recon and enumeration efforts, how to exhaust the penetration testing team's time and effort, how to properly scrub outbound and inbound traffic, and finally, we present some obscure methods for preventing a successful penetration test. The techniques discussed in this are meant to keep a penetration testing team from successfully finding what might otherwise be insecure systems on our network.

2. Background

2.1. The Penetration Testing Cycle

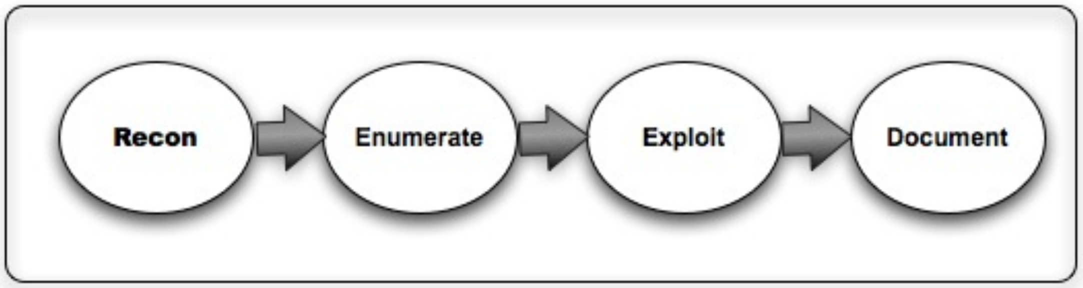


Figure 1: The Penetration Testing Cycle

In order to prevent penetration testers from succeeding in attacking our networks, it is important to understand the penetration testing cycle. Weissman (1993) and Wack (2003) outline different methodologies for a penetration testing cycle. While the terminology each use to describe the cycle differs, four unique phases exist in a penetration test cycle as shown in Figure 1.

During the first phase, reconnaissance, a penetration testing team attempts to get an overview of the target to develop more specific information. Penetration testers may perform reconnaissance with open-source information and tools to acquire a specific view of the target. After reconnaissance, penetration testers move into an enumeration phase. During this phase, the team identifies entry points into the network. In addition, the team may use automated toolkits to identify vulnerable services, servers, and hosts on the network. Once the team identifies a vulnerability, the team move into the next phase: exploitation. This is the pivotal phase in the cycle, by actively attacking the service or host the penetration tester proves the systems are vulnerable to an exploit. Finally, the team wraps up their work with successful documentation of their efforts in a penetration testing report that is provided to the client. Although we have described this order linearly, it is important to note that penetration testers move back and forth fluidly between several of the phases. For example, a penetration testing team will not necessarily document all of the vulnerabilities found before moving to exploitation. This ensures the team has results it can show a client, who usually has the team on a very tight time schedule.

In this paper about defeating penetration testing, our goal is not to make the network more secure in a strict sense. Rather, we examine ways to obscure our network from a successful penetration testing team. In order to achieve this goal, we discuss methods for keeping the penetration testing team stuck in the reconnaissance, enumeration, and documentation phases. The more of their time we can consume during these phases, the less likely it is that a team will be able to launch a successful exploit. To understand the different cycles and how we can break them, it is important to describe some of the tools a penetration tester might use in his or her attack.

2.2. A Pentester’s Arsenal

A penetration testing team may have thousands of different tools they rely on in order to recon, enumerate, and exploit during their penetration test. To present all of the tools used by a penetration testing team is beyond the scope of this paper. Rather, we describe some of the high-profile tools used by several different penetration testing teams. These tools may include automated, open-source information reconnaissance tools, port scanners, vulnerability scanners, brute force password guessing tools, automated web-application attack tools, or large all-encompassing frameworks that provide complete functionality.

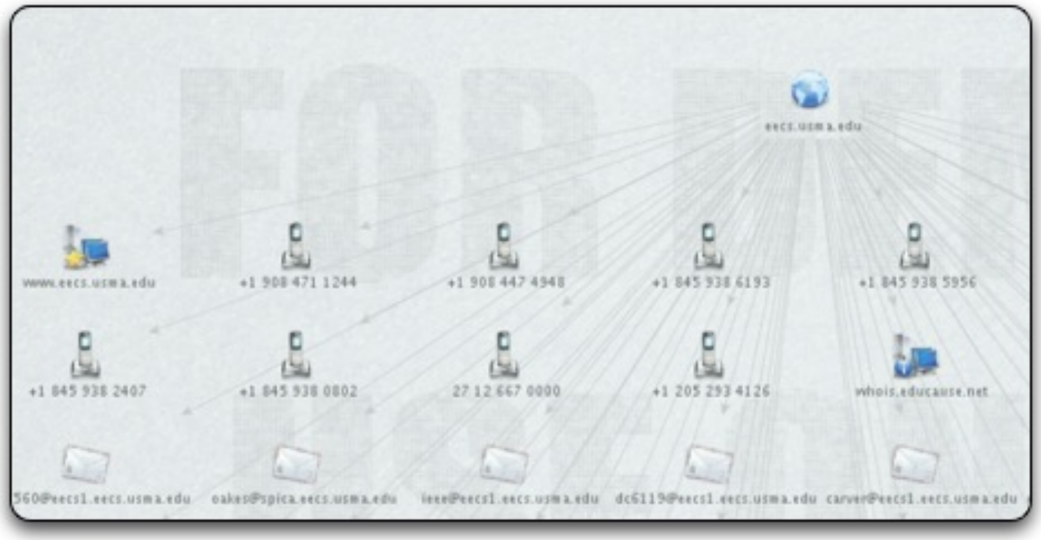
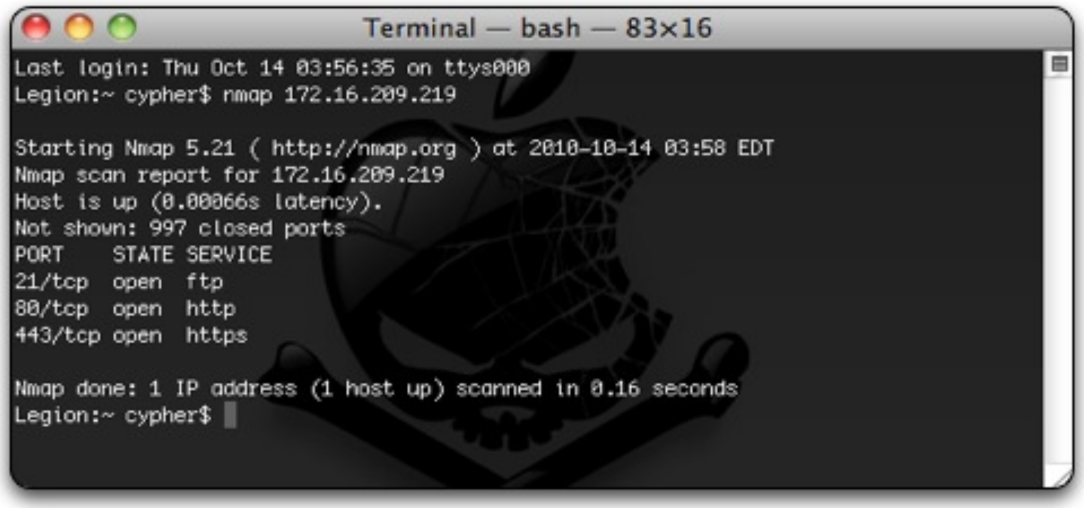


Figure 2: The Maltego Information Reconnaissance Toolkit

During the reconnaissance phase, a team may use a toolkit such as Maltego, as shown in Figure 2. The Maltego toolkit provides an open source intelligence-gathering engine that mines the web for data. By performing transforms against input such as a domain name or URL, Maltego searches the web for useful targeting information such as email addresses that can be used as a vector to deliver a social engineering or client side attack later in the penetration test.

Another tool used for reconnaissance and enumeration of targets is nmap (network mapper). It is an open source utility for network exploration and security auditing. The author, Feydor, provides the tool for download at http://nmap.org/. Nmap contains a series of rules and fingerprints to identify the services and operating systems of different machines on a network. It compares the responses from a target against its massive database to provide the penetration testing team with more specific data about a target. Figure 3 shows results generated by the nmap tool.



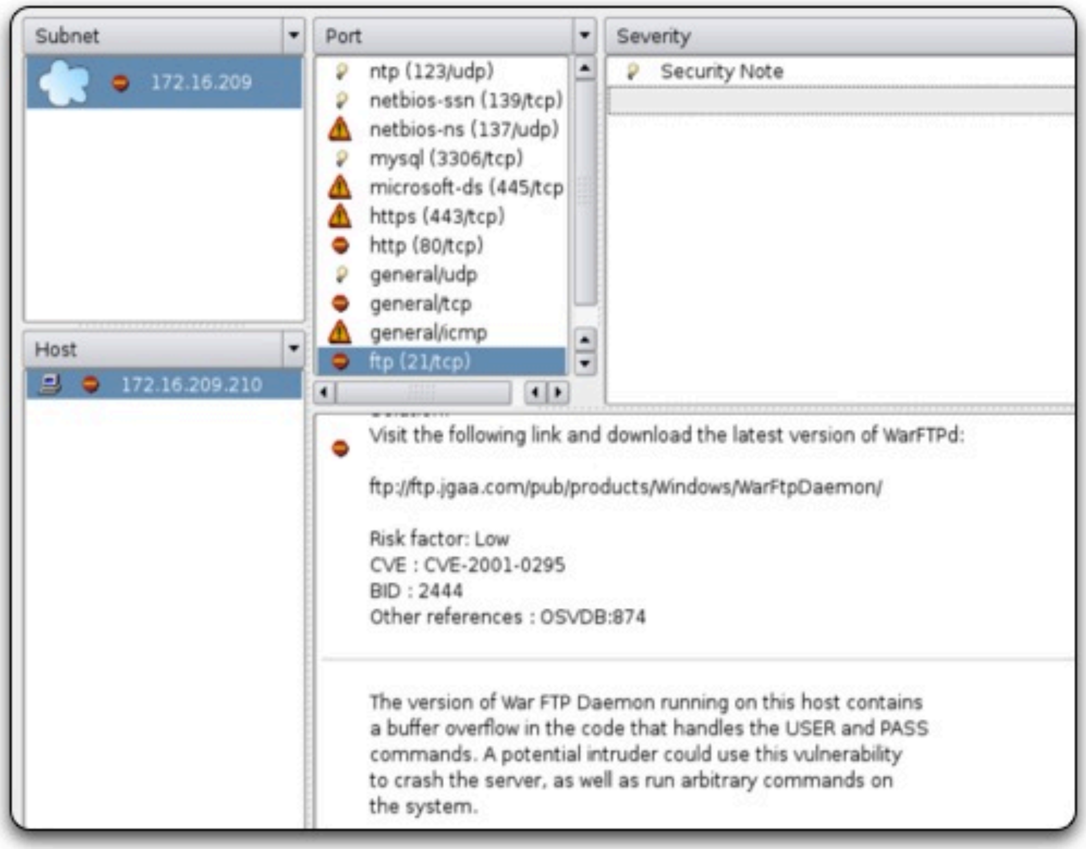
```
Last login: Thu Oct 14 03:56:35 on ttys000
Legion:~ cypher$ nmap 172.16.209.219

Starting Nmap 5.21 ( http://nmap.org ) at 2010-10-14 03:58 EDT
Nmap scan report for 172.16.209.219
Host is up (0.00066s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE
21/tcp  open  ftp
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.16 seconds
Legion:~ cypher$
```

Figure 3: NMAP (Network Mapper) – Host/Service Enumeration Toolkit

Penetration testers may take the results from an nmap automated scan against the network and import it into commercial tools like the Nessus Vulnerability Scanner available at http://www.nessus.org/nessus/. The Nessus Vulnerability Scanner maintains a large database of vulnerable software and services and connects to different hosts and servers to determine whether they are vulnerable to any of these exploits. The Nessus Vulnerability Scanner, depicted in Figure 4, provides the penetration testing team with an overview of vulnerable services by host and port.



Visit the following link and download the latest version of WarFTPd:

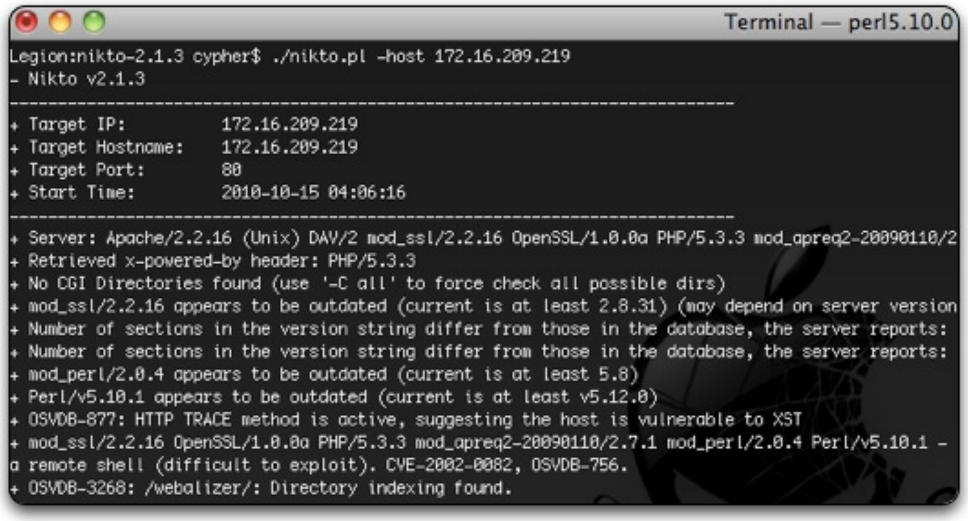
ftp://ftp.jgaa.com/pub/products/Windows/WarFtpDaemon/

Risk factor: Low
CVE : CVE-2001-0295
BID : 2444
Other references : OSVDB:874

The version of War FTP Daemon running on this host contains
a buffer overflow in the code that handles the USER and PASS
commands. A potential intruder could use this vulnerability
to crash the server, as well as run arbitrary commands on
the system.

Figure 4: The Nessus Vulnerability Scanner

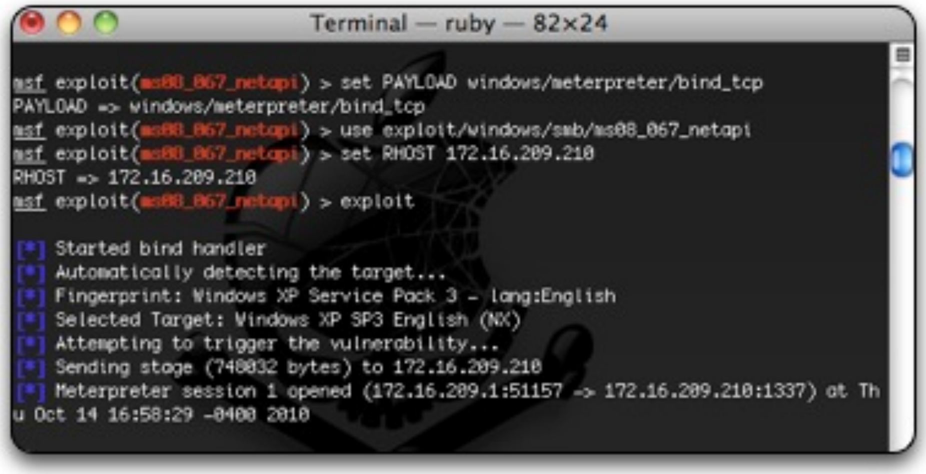
For attacking web applications, a team may use an automated scanner such as Nikto (http://cirt.net/nikto2) to determine vulnerabilities. Nikto connects to webservers, looking for over 6,400 potentially dangerous files/scripts, and checks for outdated version of over 1,000 servers. Using Nikto, a penetration testing team can identify a vector such as cross-site-scripting, file-upload, or remote-file-inclusion to attack a server. Additionally, as depicted in Figure 5, the results of Nikto may steer the penetration testing team toward exploits that would succeed against the target.



```
Legion:nikto-2.1.3 cypher$ ./nikto.pl -host 172.16.209.219
- Nikto v2.1.3
---------------------------------------------------------------------------
+ Target IP:          172.16.209.219
+ Target Hostname:    172.16.209.219
+ Target Port:        80
+ Start Time:         2010-10-15 04:06:16
---------------------------------------------------------------------------
+ Server: Apache/2.2.16 (Unix) DAV/2 mod_ssl/2.2.16 OpenSSL/1.0.0a PHP/5.3.3 mod_apreq2-20090110/2
+ Retrieved x-powered-by header: PHP/5.3.3
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ mod_ssl/2.2.16 appears to be outdated (current is at least 2.8.31) (may depend on server version
+ Number of sections in the version string differ from those in the database, the server reports:
+ Number of sections in the version string differ from those in the database, the server reports:
+ mod_perl/2.0.4 appears to be outdated (current is at least 5.8)
+ Perl/v5.10.1 appears to be outdated (current is at least v5.12.0)
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
+ mod_ssl/2.2.16 OpenSSL/1.0.0a PHP/5.3.3 mod_apreq2-20090110/2.7.1 mod_perl/2.0.4 Perl/v5.10.1 -
a remote shell (difficult to exploit). CVE-2002-0082, OSVDB-756.
+ OSVDB-3268: /webalizer/: Directory indexing found.
```

Figure 5: The Nikto Web Vulnerability Assessment Toolkit

When preparing to exploit a target, the penetration testing team may choose to use an all-encompassing framework such as MetaSploit to attack the attack. The MetaSploit development team makes the project freely available under the BSD license and allowable for download at http://www.metasploit.com. Figure 6 depicts a screenshot of MetaSploit being used to attack a Windows XP SP2 machine. In the figure, the user selects an exploit, MS08-067 (Conficker), and a payload, the meterpreter shell. As a result of the exploit, the attacker can remotely command the target via the shell.



```
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/bind_tcp
PAYLOAD => windows/meterpreter/bind_tcp
msf exploit(ms08_067_netapi) > use exploit/windows/smb/ms08_067_netapi
msf exploit(ms08_067_netapi) > set RHOST 172.16.209.210
RHOST => 172.16.209.210
msf exploit(ms08_067_netapi) > exploit

[*] Started bind handler
[*] Automatically detecting the target...
[*] Fingerprint: Windows XP Service Pack 3 - lang:English
[*] Selected Target: Windows XP SP3 English (NX)
[*] Attempting to trigger the vulnerability...
[*] Sending stage (748032 bytes) to 172.16.209.210
[*] Meterpreter session 1 opened (172.16.209.1:51157 -> 172.16.209.210:1337) at Th
u Oct 14 16:58:29 -0400 2010
```

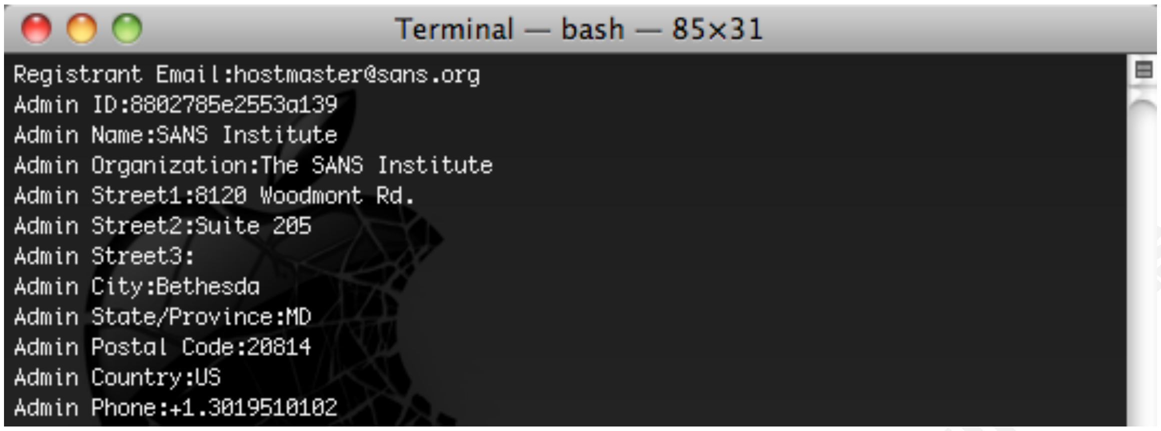
Figure 6: The MetaSploit Exploitation Framework

3. Mess with the Early Information Gathering

Understanding where a penetration testing team may begin their reconnaissance efforts is critical to starting an early defense. In black box testing, the team assumes the role of an outside hacker with no prior knowledge of the infrastructure. Finding internal targets and contacts inside the organization will most likely be a logical first step. To do this, they will use open-source intelligence gathering tools like Maltego, Whois, and to some degree, Google. They will crawl webpages, looking to enumerate targets inside the organization, comb through job add requirements and find publically posted personal information about the target.

3.1. Defensive Social Engineering

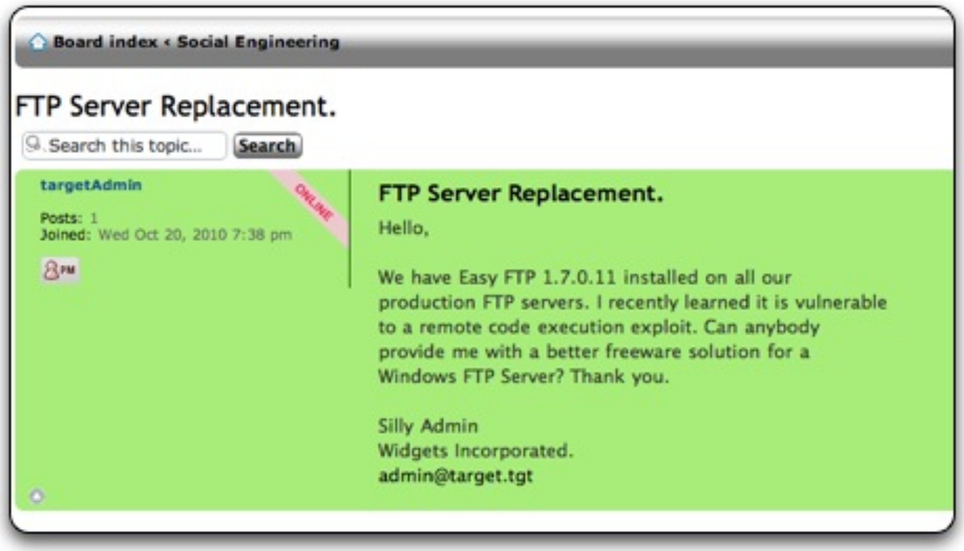
The earlier that you, the potential victim, can affect the recon and enumeration cycle, the better. Consider, for example, the results returned by a Whois lookup, shown in Figure 7. The information stored in a Whois registration typically contains contact information for the Internet target. This could be a phone number, physical address, or email address. All provide a perfect opportunity to provide false information. It might be a good idea to register a specific email address and your organization's phone number, both of which are unique to the domain registration. When you receive correspondence at these addresses and numbers, you know it originated from a Whois lookup, which helps to safely avoid a social engineering, client-side, or phishing attack. Further inspection of this correspondence can help identify the types of phishing and social engineering attacks that the penetration testing team uses and allows you to write signatures for defense as well as inform your organization of mitigation factors.



```
Registrant Email:hostmaster@sans.org
Admin ID:8802785e2553a139
Admin Name:SANS Institute
Admin Organization:The SANS Institute
Admin Street1:8120 Woodmont Rd.
Admin Street2:Suite 205
Admin Street3:
Admin City:Bethesda
Admin State/Province:MD
Admin Postal Code:20814
Admin Country:US
Admin Phone:+1.3019510102
```

Figure 7: Whois Registrant Information for sans,org

Another opportunity to frustrate a penetration testing team is to dangle tidbits of information directly in front of them. Consider the posting depicted in Figure 8. In this example, an administrator explains that his entire organization is vulnerable to a particular FTP exploit. How much time will a penetration testing team then spend to identify FTP Servers in the organization and attempt to weaponize existing exploits before eventually quitting because the exploit fails? A subsequent post to the message board might read something like this: *"We also need a 64-bit OS version since we run mostly on 64-bit architecture."* The penetration testing team will have to go back, refine their exploits to conform to the memory-addressing scheme in a 64-bit OS, and try again. Subtly placed traps such as this can be used to keep the penetration testing team from finding and exploiting actual vulnerabilities that exist in your organization.



FTP Server Replacement.

targetAdmin
Posts: 1
Joined: Wed Oct 20, 2010 7:38 pm

FTP Server Replacement.

Hello,

We have Easy FTP 1.7.0.11 installed on all our production FTP servers. I recently learned it is vulnerable to a remote code execution exploit. Can anybody provide me with a better freeware solution for a Windows FTP Server? Thank you.

Silly Admin
Widgets Incorporated.
admin@target.tgt

Figure 8: Posting of Bogus Information to Public Forums

3.2. Tarpit Their Scanning and Enumeration

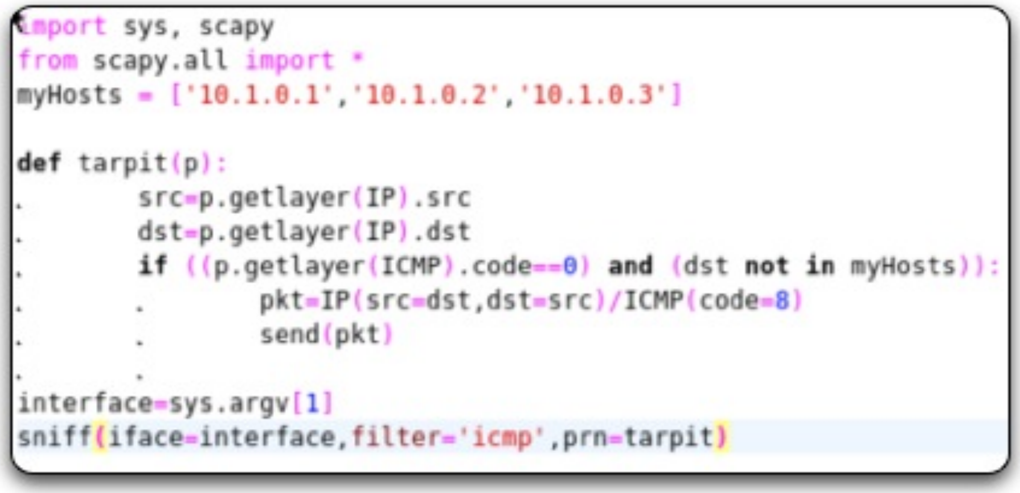
If a penetration testing team is able to safely negotiate a series of subtly placed social engineering traps, they will begin enumerating your network and services through tools such as Nmap and identify vulnerabilities with tools such as Nessus. One method to slow this process is to use a tool such as the LaBrea Tarpit. Written by Tom Liston, LaBrea is available for download at http://labrea.sourceforge.net/. Essentially, the tool responds to ICMP requests and TCP SYNs for unused IP addresses in your network. Look at Figure 9 to see how LaBrea handles sending back ARP responses with a bogus physical address of DE:AD:BE:EF:01 for the IP address 192.168.1.22 (an address which doesn't exist on our network). LaBrea sends these back after a timeout of 10 seconds and then will continue to watch for traffic at the physical address DE:AD:BE:EF:01, essentially capturing all traffic for 192.168.1.22.

LaBrea can create virtual servers that are attractive to the penetration testing team. By gradually increasing the TCP RTT threshold, LaBrea "tarpits" penetration testing teams as they try to break into the virtual servers. LaBrea can be used effectively to invalidate the results found by Nessus or Nmap when a penetration testing team is trying to enumerate hosts, services, and vulnerabilities on the target network. Combined with another means of slowing the penetration testing team's ability to attack, it can be a very effective tool for frustrating a penetration testing team.

```
09:41:45.08809 ARP who-has 192.168.1.22 tell 192.168.1.1
09:41:46.619628 ARP who-has 192.168.1.23 tell 192.168.1.1
09:42:46.631154 ARP who-has 192.168.1.24 tell 192.168.1.1
09:42:56.398505 ARP reply 192.168.1.22 is-at DE:AD:BE:EF:01
```

Figure 9: Network Reconnaissance Under LaBrea

Another very simple approach to solving this problem is to write a small Python Script that responds to ICMP echo requests or TCP SYNs for the unused IP space in your network using the Python library, Scapy (http://www.secdev.org/projects/scapy/) . A powerful interactive packet manipulation library, Scapy, can allow a user to write scripts to listen to and respond to traffic. This type of script should run on a machine that can monitor all traffic entering your corporate network and respond on behalf of the unused IP space. This can be done in minimal lines of code as depicted in Figure 10.



```
import sys, scapy
from scapy.all import *
myHosts = ['10.1.0.1','10.1.0.2','10.1.0.3']

def tarpit(p):
        src=p.getlayer(IP).src
        dst=p.getlayer(IP).dst
        if ((p.getlayer(ICMP).code==0) and (dst not in myHosts)):
                pkt=IP(src=dst,dst=src)/ICMP(code=8)
                send(pkt)

interface=sys.argv[1]
sniff(iface=interface,filter='icmp',prn=tarpit)
```

Figure 10: Python Script to Respond to all ICMP Echo Requests.

Another final option on the *BSD Operating System is to employ a blackhole. On the *BSD kernel, *blackhole* is a special option to control system behavior when someone connects to a closed TCP or UDP port. Changing the *BSD kernel to blackhole TCP or UDP traffic is fairly easy, as seen below in Figure 11.

```
change (sysctl -w net.inet.tcp.blackhole=[0 | 1 | 2]sysctl
—w net.inet.udp.blackhole=[0 | 1]
```

Figure 11: *BSD Blackhole Functionality

3.3. Setting the Bait for Honeypots

A penetration testing team may navigate through the series of tarpits to find the organization's Domain Name Server (DNS). After discovering the DNS Server of an organization, a penetration team may attempt to enumerate targets by brute forcing or zone transferring information out of the DNS server. Typically, DNS does not require any authentication and receives messages via UDP port 53. The team may attempt to query for services such at the target organization by submitting requests to resolve IP addresses for hosts named www, smtp, or filesvr. RSnake wrote the Fierce.pl tool available at http://ha.ckers.org/fierce/ to enumerate through a series of common hostnames and attempt to resolve them. The list of hosts he uses to brute force through is available at http://ha.ckers.org/fierce/hosts.txt.

Provided the hostnames aren't already in your DNS records, it would be a good idea to add them all to your DNS Server with bogus records pointing to either unused IP space or honeypots. Additionally, you could take advantage of the Bind 9 features that let you return different results based on the origin of the query. Allowing zone transfers on a bogus DNS server not in production use by your organization also provides false information. Providing DNS entries to honeypots or tarpits will surely frustrate the penetration testing team as they attempt to exploit the target. One honeypot is HoneyD, available at http://www.HoneyD.org/. HoneyD can mimic different operating system personalities and can offer bogus services through a series of different scripts. Several of these scripts and configurations are available at http://www.HoneyD.org/configuration.php. Further, it can instantiate several different virtual machines with unique IP addresses. Imagine the excitement level when the penetration team discovers three vulnerable Microsoft IIS servers in the target space as demonstrated in Figure 12.

```
create default
set default personality "Microsoft Windows XP Professional
SP2"
add default tcp port 80
"/usr/share/HoneyD/scripts/win32/web.sh"
set default default tcp action reset set
default default udp action reset
bind 192.168.1.99 default
bind 192.168.1.98 default
bind 192.168.1.97 default
```

Figure 12: HoneyD Sample Configuration File

3.4. Give Them a *Vulnerable* Web App to Attack

When the penetration team attempts to attack your webservers, it is important to understand their methodology. If they run automated tools such as Nikto, it is a good idea to respond to those tools with bogus responses. Nikto begins its efforts by examining the robots.txt file that lists directories that web-spiders should not traverse. Typically, these are excellent vantage points to attack a target. Imagine the penetration testing team's excitement when they see a directory called *router-firmware-upload* or *admin-config-file.* Try considering using a server script such as the one depicted in Figure 13. This PHP script renders a web page that pretends to accept and publish any files. Penetration testing teams may attack webservers by uploading PHP toolkits such as Commander99, PHPSHell or AJAXShell that provide a command and control channel once uploaded to the server. This script will ideally give the penetration testing team a sense of false success as they attempt to find where their PHP toolkits landed on the server.

```
<?php
$target_path = "uploads/" . basename(
$_FILES['uploadedfile']['name']);
# removed to remove file upload capability
echo "The file ".  basename(
$_FILES['uploadedfile']['name']).  " has been uploaded to
".$target_path;
?>
```

Figure 13: Bogus PHP File Upload Trap

Further, setting up bogus and unusable web applications is a proven concept. Consider the recent news about the ZeuS botnet. The authors of the ZeuS botnet setup a phony administrative panel that provided law enforcement and researchers fake statistics. The login system for the admin panel accepted default passwords and accepted SQL injection strings. (Leyden, 2010)

3.5. Hide Your Identity with Banner Masking and OS Personality

Another common obfuscation technique is to change the header/banner hostname on a server. Servers present the hostname when you connect to the service to tell what version of software is running. It might tell the user that the server is running an Apache Webserver (version 2.0.41 development) on a UNIX platform. For example:

Server: Apache/2.0.41-dev (UNIX)

On the Apache Server, this can be modified by using the *mod_headers* module, available at http://httpd.apache.org/docs/current/mod/mod_headers.html. Using mod_headers, an admin could present the server as a Microsoft-IIS/5.0 server instead sending a new banner. For example:
Server: Microsoft-IIS/5.0

Conversely, if an admin being penetration testing wanted to present an IIS Server as an Apache Server, he could use similar, Windows-based tools such as IISLockDown and use the configuration option in URLScan's INI file to modify and replace the header that IIS presents to connected clients.

Changing the header banner may trick some penetration testing teams but a further inspection or scan will use TCP Stack Fingerprinting to determine the Operating System and server software of a target. Penetration tools such as p0f, available at http://lcamtuf.coredump.cx/p0f.shtml, identify the operating system by looking at a variety of TCP options during communications. By identifying the fingerprint, p0f can tell the penetration testing team the Operating System, release version and server that would employ those options.

Negating the ability for tools like p0f to succeed means deviating from the standard fingerprint. This is fairly easy to do by changing some options. But this simply removes the fingerprint. To replace the fingerprint means that the TCP stack on the target must employ the same options as the TCP stack of the machine you would like to imitate. On the Linux >2.4 Kernel, IPPersonality (http://ippersonality.sourceforge.net/) can modify the TCP Stack to take the personality of other operating systems. Similar software on Windows offers the same functionality. Servermask, available at http://www.seoconsultants.com/windows/servers/servermask, modifies both the hostname banners and changes the IIS Web Server fingerprint by removing unnecessary HTTP header data and adjusting other response information.

4. Consume Their Time

Tsun Tzu warned "*To secure ourselves against defeat lies in our own hands, but the opportunity of defeating the enemy is provided by the enemy himself.*" (Sawyer, 1994) Let us explain what that means in terms of a good penetration test. At the end of the day, a good penetration testing team should succeed. The enormous amount of resources most targets have to protect with a limited force and training favors the adversarial penetration testing team. Further, if the penetration testing team is really good, they may even have 0-day exploits developed. This makes it very difficult for a target to protect their resources. The sole advantage the target has is time. The penetration testing team provides that weakness. Time is usually fixed on a penetration test and written into the contract by the team. This opens a window of success for the target. In the following section, we examine some ways to take time away from the penetration testing team.

4.1. Give Them Password Hashes to Crack

A penetration testing team may attempt to perform a directory traversal attack by navigating outside of the webserver directory into the base operating system. On a Linux webserver, it is common for a penetration testing team to try navigating to pages such as **http://target.tgt/../../etc/passwd** or **http://target.tgt/../../etc/shadown** an effort pull the files containing user accounts and hashed passwords. Next, the team will import those hashes into a program such as John The Ripper (http://www.openwall.com/john/) to crack the passwords by either dictionary-based or brute force-based attacks.

During a brute force attack, the penetration testing team's software attempts to enumerate all possible passwords and then performs the appropriate hashing function against them, comparing the results with the hashed credentials stolen off the target. Typically this takes several CPU cycles and is occasionally run distributed by the team with tools like djohn (http://www.bindshell.net/tools/dnetj). Consider the attacking resources you could take away from a team that consumes their CPU cycles cracking bogus passwords found by a bogus directory traversal.

4.2. Slow Down Their Attacks with Sticky HoneyPots

In addition to consuming the team's efforts by having them crack bogus passwords on legitimate webservers, another option to frustrate them is to steer the team entirely away from legitimate targets to bogus targets. There are several different methods for doing this. We have already mentioned the use of false DNS records and creating honeypots using HoneyD. HoneyD provides the option to create *sticky* targets.

These *sticky* targets consume the time of the attacker by gradually increasing the RTT time used in the TCP connection. The penetration testing team's level of frustration will grow as they connect to a bogus mailserver and attempt to issue SMTP commands painstakingly waiting on the console to echo their keystrokes. Look at the example configuration in Figure 14 for how to create a sticky mailserver.

```
Create default
set default personality "Apple Mac OS X 10.1 - 10.1.4"
set default default tcp action tarpit open
```

Figure 14: Configuration of a *Sticky Honeypot* using HoneyD

In addition to HoneyD, another excellent honeypot is tiny honeypot available for download at http://freshmeat.net/projects/thp/. Tinyhoneypot can also be used to an excellent intrusion detection method (Hammer, 2006). Setting up Tinyhoneypot is rather easy, as it is available in most package repositories such as apt. Once installed, the easiest way to get Tinyhoneypot running is to create an inetd.conf configuration file that starts the Tinyhoneypot services at bootime. Figure 15 has a sample configuration that starts ftp, smtp, and http services.

```
ftp  stream tcp   nowait  thpot  /usr/sbin/thpot thpot ftp
smtp stream tcp   nowait  thpot  /usr/sbin/thpot thpot smtp
http stream tcp   nowait  thpot  /usr/sbin/thpot thpot http
```

Figure 15: Tinyhoneypot Inetd Configuration File

Notice that Tinyhoneypot already has built-in scripts that know how to start these services, as shown in Figure 16. An administrator can manipulate the Tinyhoneypot configuration file (/etc/thpot/thpot.conf) in order to specify version of the service Tinyhoneypot will attempt to emulate. Any subsequent connections to that service are logged in /var/log/thpot.

```
  my @fver;
  $fver[1] = "FTP server (Version wu-2.6.0(1))";
  $fver[2] = "FTP server (BSDI Version 7.00LS)";
  $fver[3] = "FTP server (PFTP 0.13)";
  $fver[4] = "NcFTPd Server";
  $fver[5] = "Microsoft FTP Service (Version 5.0)";
```

Figure 16: Tinyhoneypot Application Configuration File

While directing users to honeypots and bogus targets, it is important to realize that the penetration testing team will eventually stumble across your actual targets. One option to delay this is to place services on nonstandard ports. Placing SSH on port 65,022 instead of port 22, or a webserver on 65,080 instead of 80, may help in delaying tools from finding your actual services ,and have minimal impact on your customers. Anecdotally, Daniel Miesser set up a server with SSH on port 22 and observed 7,025 connection attempts from malicious tools. He then moved the SSH server to port 24 and counted only 4 connection attempts during an equivalent period of time (Miesser, 2010).

5. Whitewash Traffic

By shoveling inbound and outbound traffic through a proxy, you can significantly limit the amount of client-side attack applications that will succeed in entering your organization. Squid is an open-source proxy available for download at http://www.squid-cache.org that functions as a HTTP, HTTPS and FTP proxy. Setting up and configuring a Squid Proxy for simple operation is relatively easy (Bauer, 2009).

5. 1. Strangle Their Exploits with a Squid and a Python

However, the beauty of Squid is the amount of highly configurable access controls and external scripting that it can perform. To start, let's use it to protect our browsers from leaking user-agent strings with information about vulnerable clients. As client-side exploits continue to grow, penetration testing teams will target browsers and other desktop applications. The browser sends a user-agent string with the browser name, operating system, and version information. A database of user-agents can be found at http://www.user-agents.org/. The user-agent can be used by the penetration testing team to refine an exploit for delivery to the correct application and operating system. Using a squid Proxy, an admin could easily replace the user-agent string with the most up-to-date browser with different operating system as depicted in Figure 17. Then, seeing that the majority of web clients are using Mozilla on FreeBSD, a penetration team may stop trying to use a Windows-specific exploit like Aurora to target the browser.

```
header_replace User-Agent Mozilla/5.0 (X11; U; FreeBSD
i386; en-US; rv:1.9.0.10) Gecko/2009060215 Firefox/3.0.11
```

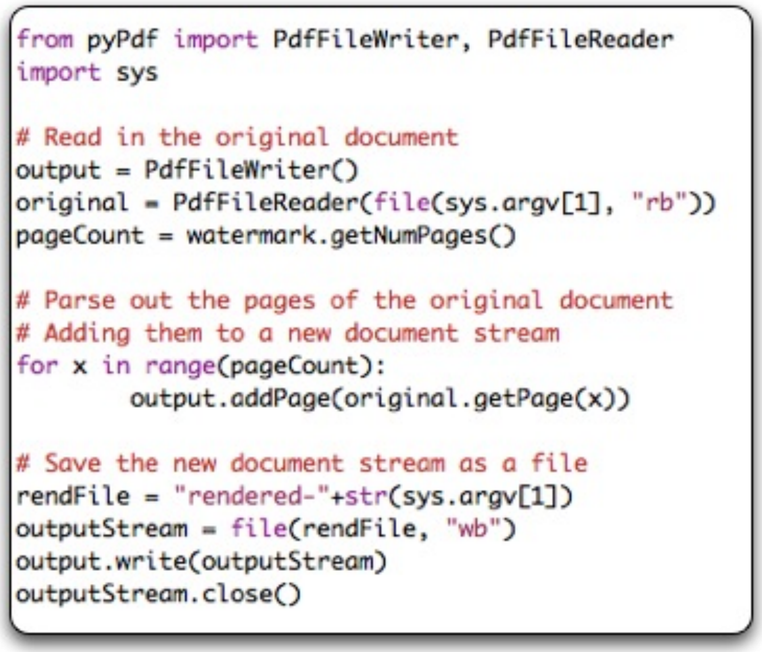
Figure 17: Replacing User Agent Strings in Squid

To strengthen the defense even more, Squid can be used to render content as it enters the network perimeter. A network administrator can make an external script call such as the one in Figure 18 below:

```
url_rewrite_program /usr/share/scripts/scrubMyTraffic.py
```

Figure 18: Rewriting Traffic in Squid using an external Python Script

The options for the functionality of this external script are endless. Didier Stevens wrote several tools to detect malicious PDFs that deliver client-side exploits to Adobe. These scripts are available for download at http://blog.didierstevens.com/programs/pdf-tools/. Another option is just to read in the PDF and destroy the JavaScript functionality completely by removing any JavaScript objects in the document. This could get rather complex, through making sure the rendering doesn't destroy any of the indirect indexing in the document. After all, we still have to provide functionality on our network while defeating penetration testers. So, a very simple option is to regenerate the PDF using Python. When the PDF is generated using a script, the JavaScript options are removed since they do not get read in by the PDFFileReader. Thus, a flat version of the document is reprinted – causing exploits to fail. An example of such a script is show in Figure 19, and was written and used by Anthony Rodriguez to prevent the NSA's Red Team from landing any client-side attacks against Adobe during the 2010 Cyber Defense Exercise.



```
from pyPdf import PdfFileWriter, PdfFileReader
import sys

# Read in the original document
output = PdfFileWriter()
original = PdfFileReader(file(sys.argv[1], "rb"))
pageCount = watermark.getNumPages()

# Parse out the pages of the original document
# Adding them to a new document stream
for x in range(pageCount):
        output.addPage(original.getPage(x))

# Save the new document stream as a file
rendFile = "rendered-"+str(sys.argv[1])
outputStream = file(rendFile, "wb")
output.write(outputStream)
outputStream.close()
```

Figure 19: Reprinting PDF Files in Python to remove malicious JavaScript

Along with rendering exploits null, a good idea is to keep track of their callback location. This may involve a comprehensive inspection of the PDF and a fairly intensive malware analysis. However, one simple trick is to take a look at the /URI object inside the PDF. The /URI object as show in Figure 20 can force a PDF reader to start the default browser and navigate to a site each time the document is opened. An attacker could use this to recon a platform to develop a successful client-side exploit against the default browser. Using a Python script, you can redirect all /URI to some IP address where you have a listening post. This would let you get a good idea of who in the company is under a social engineering/client-side attack.

```
1337 0 obj
<<
/Type /Action
/S /URI
/URI (http://penetrationtestingteam.com)
>>
end obj
```

Figure 20: Replacing PDF /URI Fields to remove callbacks.

One other option is to use the browser to filter content that contains shellcode. Using a tool such as libemu (http://libemu.carnivore.it), an option is to write a Python script that inspects streams for shellcode and prevents communication with the offending host. This could be combined with a tool such as TCPKill or TCPNice (available from http://monkey.org/~dugsong/dsniff/) as a very primitive intrusion prevention system (IPS).

6. Hide and Protect the Castle

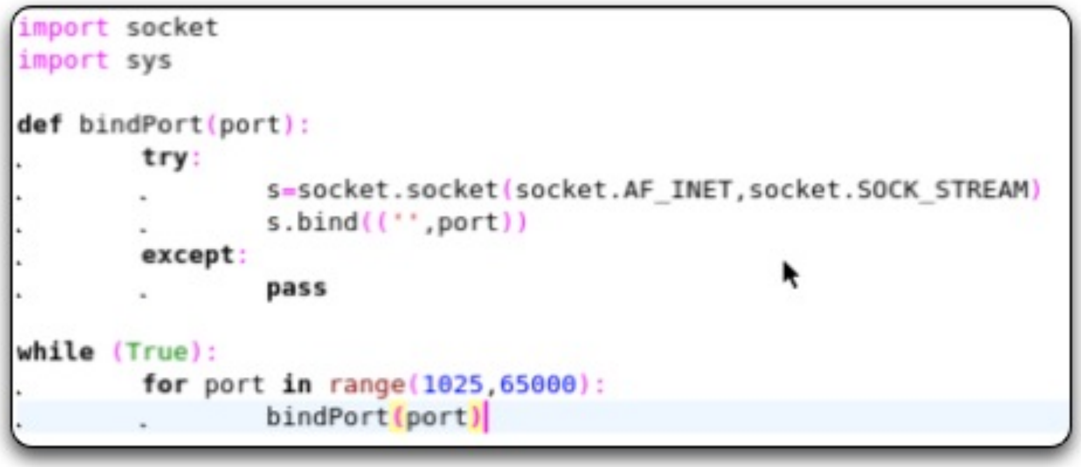
Greene insisted a good leader should "remove any targets you have for your opponents…. No targets will frustrate your opponents increasing the chance they will make a mistake." (Greene, 2006) Although Greene described this using an anecdote from Napoleon's 1812 invasion of Russian, the strategy still holds true against computer penetration testing teams of today. A penetration testing team that cannot route, scan, or even determine if targets are alive stands a tough chance of actually landing an exploit against them. In the following section we describe some methods for hiding the castle from the onslaught of attack by the penetration testing team.

6.1. Access Control List (ACL) The World

Properly configured Access Control Lists (ACLs) can prevent a penetration testing team from successfully landing exploits and triggering shellcode that calls back to the team. Consider the example where your office employees only require access to the Internet via a browser. ACLs can prevent these workstations from egressing the corporate network completely and instead force them to use a local proxy like Squid. Depending on your company's requirements, the approach here is to allow high ethereal ports to connect directly to the Squid proxy for outbound connections. Everything else should be blocked unless you have requirements such as VOIP that requires the use of externally open UDP ports. That being said, you have 65,536 ports on your machine, and you only use a couple of them for connections. To prevent an external attacker from binding shellcode on your machine, we present another option.

6.2. Bind Your Ports So Nobody Else Can

To prevent inbound exploits from succeeding, one option is to limit the target space of available ports on your servers. If ACLs cannot be used because your server is beyond the DMZ on your organization, one option is to prevent shellcode binding. Also, it's important to note that sometimes ACLs will drop if the router becomes overwhelmed, as the routing process takes a higher precedent than the process that checks the ACLs. To bind a good majority of your ports, consider writing a script similar to the one in Figure 21. Here, the ports 1025 through 65,000 are bound and will prevent a penetration testing team from binding their shellcode for a callback. This still leaves 535 ethereal ports for the operating system, which is more than the client needs for communication in most scenarios. With a little more creativity, you could randomize the ports that are bound.



```
import socket
import sys

def bindPort(port):
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.bind(('',port))
        except:
                pass

while (True):
        for port in range(1025,65000):
                bindPort(port)
```

Figure 21: Python Script to Bind Ephemeral Ports on a Workstation

6.3. Use NAT, IPv6, and Random Sprays of Traffic to Hide

In addition to hiding the target by binding ports, a variety of other techniques can be employed. One method is to use a double network address translation (NAT) implementation. As depicted in Figure 22, double NAT hides the target space and makes it hidden from source routing since a penetration testing team would have to jump through two separate NAT translation tables in order to communicate with the target space. Another option to consider is an interesting use of IPv6 and IPv4.

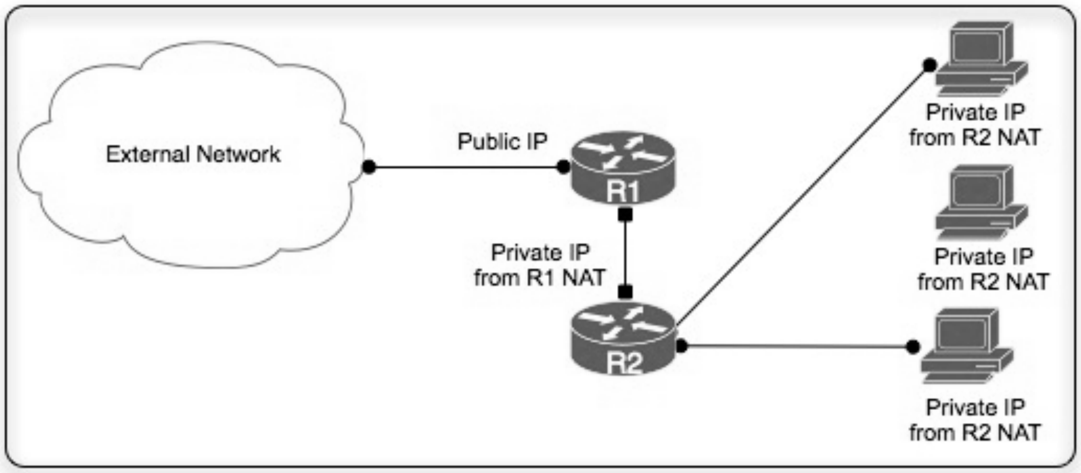


Figure 22: Security Provided by Double-NAT

The United States Naval Academy used an interesting strategy to win the 2010 Cyber Defense Exercise. Upon initial forensic investigation of compromised machines, they discovered all malware was set to callbacks to IPv4 addresses. They found infected malware throughout the network they were responsible for defending. Unable to clean the malware prior to the exercise, they converted the infected internal network infrastructure to an IPv4-only VLAN and then talked externally only via IPv6. For more information on the use of IPv4 and IPv6 VLANs, refer to RFC4554: Use of VLANs for IPv4-IPv6 Coexistence in Enterprise.

Finally, an excellent idea is to spray random ICMP traffic outside your network. If a penetration testing team can physically tap the line between our external gateway and the Internet, the penetration testing team will see ICMP traffic such as Echo-Replies without initiating Echo-Requests. Tools like BlackICE PC Protection will recognize this as internal NAT leakage and begin mapping out the internal network. However, all the penetration testing team will have are addresses to targets that do not exist on the internal IP space. While writing your own tools for this is preferred, hping3 (http://www.hping.org) is a great packet crafting toolkit that can be used to accomplish this effort.

7. Obscure and Secure

Anecdotally, we once heard of a team competing in a network warfare competition who were banned for cheating. With the understanding that one person's cheating is another's creativity, we investigated the scenario further. The team was responsible for hosting web services and preventing those services from being attacked and modified. The exercise was trivial since the machines hosting the content were already infected and contained vulnerable software. The banned team quickly copied the contents of their webserver to a cd-rom and then linked their /var/www directory to point at /mnt/cdrom/. Once the machine was compromised, attackers were infuriated to find out that their changes to files wouldn't commit even as a root or superuser. In this section, we discuss a few more examples, like running your static webserver off a non-writeable medium.

7.1. Take a Lesson from the FBI with DECAF

In early 2009, the FBI and Microsoft began distributing a toolkit known as COFFEE. Law enforcement built the COFFEE toolkit for basic digital forensics. The tool ran a variety of commands, dumped physical memory, and checked the status of processes and network connections. The tool provided a useful opportunity for untrained first responders to get access at data before it was destroyed, contaminated or compromised.

In response, a group of hackers built a toolkit known as DECAF. The DECAF toolkit deletes temporary files or processes associated with COFEE, erases all COFEE logs, disables USB drives, and contaminates or spoofs a variety of MAC addresses to muddy forensic tracks (Zetter, 2009). This leaves the machine essentially in a state that cannot be interrogated by COFFEE.

When defending against a penetration testing team, it's important to figure out the actions they take to achieve the objective. They will most likely try to get a listing of processes on the machine, a screenshot, or grab the log information. Any of these three can prove to the CEO that they have been able to compromise a machine in the organization. Taking away a lot of those capabilities can make it difficult for a penetration testing team to ever prove they compromised the machine.

7.2. Be Unique, Use Your Shell In Unexpected Ways

The location of key log files, the windows registry, the startup folder, and even the system directory should all be relocated to frustrate a team attacking a Windows machine. On a *Nix machine, a good idea is to make a copy of bash for use by your clients and name it something unique such as xbash. After providing your users functionality with xbash, an admin could go in and Trojan /usr/bash to defend against interrogation of a machine or callbacks by a script that makes a direct reference to #!/bin/bash.

```
Doskey ipconfig=echo silly rabbits trix are for kids
doskey /listsize=10
```

Figure 23: Modifying the Default Shell (cmd.exe) Behavior on Windows

Moving the default cmd.exe shell on Windows to bash.exe is not only creative but also entertaining. Aliasing commands in the Windows shell can be performed by using the *doskey* command, as shown in Figure 23. Another good idea is to limit the command history using *doskey.* On *Nix, this can be done for the bash shell in the .bashrc resource configuration file. Moreover, all of this can be done with relative ease and causes great frustration to a penetration testing team who successfully compromises one of your machines and attempts to execute commands from the console. This is really the heart of our argument and approach; to frustrate the penetration testing team long enough to survive a pentest.

8. Conclusions

In summary, we have outlined several methods for obscuring your network from attack during an external penetration test. Using defensive social engineering, tarpits, and OS personality tools, we demonstrated how to hide our targets while simultaneously offering traps. Next, we discussed methods for slowing the attack by consuming cycles of the team with fake traps. We then discussed methods for removing the network visibility and exposure with a good network design and access-control lists. Finally, we introduced some very obscure methods to prevent a box once it is under attack. All of our methods are to keep a penetration testing team from succeeding in finding what might otherwise be insecure systems on our network.

Greene argued that to succeed in warfare, you must "*Know your opponent's moves and do not let your motives be known. Understand their way of thinking*." (Greene, 2006) In this paper, we have provided a deeper understanding of how penetration testing teams recon, enumerate, exploit and document their attacks. To succeed against a penetration testing team requires this level of understanding and forming a defense that takes advantage of the penetration testing team's inherent weaknesses. We have introduced a strategy that will at the very least frustrate a penetration testing team but hopeful provide a mitigation strategy to the onslaught on an attack from a formidable penetration testing team.

9. References

Bauer, M. (2009). Paranoid penguin: building a secure Squid web proxy, part I. *Linux Journal*, *180*,10.

Greene, R. (2006). *The 33 strategies of war.* London, England: Penguin Group.

Hammer, R. (2006). *Enhancing IDS using tiny honeypot.* SANS Institute InfoSec Reading Room. Retrieved from http://www.sans.org/reading_room/whitepapers/detection/enhancing-ids-using-tiny-honeypot_1665.

Leyden, J. (2010). *ZeuS miscreants offer up honeypot: CyberCrooks turn the table on researchers with fake interfaces.*The Register. Retrieved from http://www.theregister.co.uk/2010/11/05/zeus_fake_interface_ruse/.

Miessler, D. (2010). *Security and obscurity: Does changing your SSH port lower your risk.* Retrieved from http://danielmiessler.com/blog/security-and-obscurity-does-changing-your-ssh-port-lower-your-risk

Sawyer, Ralph D. (1994), The Art of War, Westview Press.

Wack, J., Tracy, M., & Souppaya, M. (2003). *Guideline on network security testing.* (NIST Special Publication 800- 42).

Weissman, C. (1993). Security penetration testing guideline. In *Handbook for the Computer Security Certification of Trusted Systems,* Center for Secure Information Technology, Naval Research Laboratory (NRL), US, 1-66.

Zetter, K. (December, 2009). Hackers brew self-destruct code to counter police forensics. *Wired Magazine*, Retrieved from http://www.wired.com/threatlevel/2009/12/decaf-cofee/.