An Application of Deception in Cyberspace: Operating System Obfuscation[1]

Sherry B. Murphy, J. Todd McDonald, and Robert F. Mills
Department of Electrical and Computer Engineering,
Air Force Institute of Technology, Wright Patterson, USA
sherry.murphy@peterson.af.mil
jeffrey.mcdonald@afit.edu
robert.mills@afit.edu

Abstract: The art of deception has played an integral role in military operations throughout history. In this research we investigate the efficacy of using operating system (OS) obfuscation as a form of deception in the cyber domain. Specifically, we study the effectiveness of host-based OS obfuscation as a way of understanding whether the technique warrants further research and application development before becoming an integral part of Air Force network defense. We accomplish this objective by examining the theoretical foundation of cyber deception and then evaluating a specific OS obfuscation tool against selected OS fingerprinting tools.

The results from our experiment show that current OS obfuscation tools are developed enough to consistently mask OS information on systems running a Windows OS.

Keywords: Operating System Masking, Polymorphic Host-based Defense, Digital Decoys, Deception, Network Reconnaissance, Network Scanning

1.0 Introduction

The number of reported cybersecurity attacks against federal agencies has more than tripled since 2006 (Hoover, 2009). In 2008, Federal agencies reported to the U.S. Computer Emergency Readiness Team that they had been victims of 18,050 cybersecurity attacks (Hoover, 2009). The opening phase of these attacks often involves attempts to gather data that will guide later stages of the attacks. One key piece of information that attackers need is the identity of the target system's operating system. Various OS fingerprinting techniques have been developed and bundled in OS fingerprinting tools. Denying the hacker access to accurate OS information by defeating these tools represents an important first step to defeating adversary scanning efforts. In this research, we posit that host-based OS obfuscation may be an effective way to mitigate such tools and methods.

For the purpose of this paper, deception refers to "those measures designed to mislead the enemy by manipulation, distortion, or falsification of evidence to induce the enemy to react in a manner prejudicial to the enemy's interests" (Chairman, Joint Chiefs of Staff, 2006) and OS obfuscation refers to "altering the signature of a computer so an adversary's tools identify the incorrect operating system, resulting in an ineffective attack" (Repik, 2008). Given these definitions, OS obfuscation is a form of deception.

2.0 Literature Review

In order to defeat an attacker, we must look for opportunities to defeat any and all phases of the attack process. Though many network professionals offer different versions of the attack process, the general anatomy of the attack process is made up of five steps: reconnaissance, scanning, gaining access, maintaining access, and covering tracks and hiding. In this paper we focused mainly on the scanning step, but for a more detailed look at the attack process we refer the reader to Repik's work on defeating adversarial intelligence gathering in the network (Repik, 2008) or Counter Hack Reloaded (Skoudis & Liston, 2006).

Operating systems have unique characteristics that can help an attacker to identify the operating system being used in the target system. Examples of these characteristics include the TCP/IP packet, response messages to queries, response messages to errors, predictability of sequence numbers, and banner information. Some specific attributes in a TCP/IP session are the values set for time-to-live (TTL), window size, Don't Fragment (DF) bit, and type of service (TOS) (Scambray, McClure, & Kurtz, 2001). By

[1] The views expressed in this article are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.

comparing the information gathered about the target OS to profiles established for various systems, a potential attacker can make educated guesses about the target OS (Skoudis & Liston, 2006).

OS information can be gathered through active and/or passive techniques. Tools that use passive fingerprinting techniques intercept and examine existing traffic to make guesses about to the OS. As passive tools are dependent on existing traffic, this method may take longer to get a more accurate answer, but they are typically not detectable (Zalewski, 2006). On the other hand, we refer to tools that interact with the target to glean information about the OS as active methods. These active measures have proven to be very effective at accurately identifying the target OS, but can be detected by an intrusion detection systems (IDS) (Scambray, McClure, & Kurtz, 2001).

One very popular tool for OS fingerprinting that uses active techniques, Network Mapper (Nmap), has timing options specifically designed to help avoid detection. By changing the timing, Nmap spreads out the appearance of log entries that result from the scan. Other than the "Normal" speed, Nmap has timing options ranging from a super-slow scan called "Paranoid" that sends one packet every 5 minutes to an accelerated mode called "Insane" for the attacker in a hurry (Skoudis & Liston, 2006).

Unlike the wide selection of OS fingerprinting tools returned from a brief Internet search, options for existing OS obfuscation tools are not as robust. One reason for this may be how difficult it is to make enough changes to deceive fingerprinting techniques, while at the same time leaving enough packet integrity so the system still works. As most AF systems run on a Windows operating system, we searched for an OS obfuscation tool that worked on Windows. We found only one OS obfuscation tool that runs on Windows systems: OSfuscate (Crenshaw, 2008). OSfuscate allows the user to select an OS to emulate and implements that selection by changing registry values. These changes are designed to stop fingerprinting tools from successfully matching the collected information to the OS profiles in the OS signature found in the fingerprinting tool databases.

2.1 Impacts of OS Obfuscation on Scanning Efforts

Denying accurate OS information about the target disrupts the attacker in several ways. On the most basic level, it prevents the attacker from exploiting default passwords established by the vendor in the event a system administrator failed to change it (Skoudis & Liston, 2006). Furthermore, if the target OS has been obfuscated, the target may be perceived as too hard of a target, especially if the hacker's goal is to use a specific exploit against a known vulnerability on a particular OS. An erroneous OS and patch level may cause the attacker to ignore the potential target. For instance, some organizations hack the registry of their own systems if they are unable to quickly patch the system. This action deceives focused network scans that look for systems to exploit based on a specific vulnerability (Birch, 2009).

Another potential benefit of OS obfuscation is that if the attacker is unsure of the OS, the attacker may not use a known effective exploit for fear it will have unintended effects on the target. Instead the attacker invests resources into researching or applying alternate attack methods with lower probability of success, or may simply move to another target. If the attacker assumes the wrong OS or is deceived into accepting false OS indications and executes a plan against the wrong OS, the consequences can again put the attacker's success at risk. Applying exploits against the wrong OS can yield results from nothing happening all the way to a system crash (Lyon, 2009). For example, a buffer overflow exploit is OS independent, but the payload (written in machine language) must comply with the OS because it makes privileged calls to the OS (Skoudis & Liston, 2006). Linux interprets machine language differently than a Windows OS. Depending on the op codes, the first few bits of all computer commands that indicate what type of call is coming, the computer system could execute a benign command that does nothing, attempt to execute non-executable code causing the program to crash, or attempt to execute code the OS cannot handle causing a "blue screen of death." All of these unwanted results can increase chance of detection, heighten user and administrator alert, or deny access to the target: ultimately, each effect may disrupt an attack before it is ever launched.

2.2 Challenges to OS Obfuscation

When employing deception in traditional operations, we need to address a couple of issues: (1) how do we deceive the enemies but not ourselves, and (2) how much of our resources should we commit to the deception that would be used for other activities? These same issues apply when employing deception in cyber. For example, system administrators use scanning and management tools to monitor the status and health of the network, troubleshoot problems, and ensure systems have the necessary patches loaded. If the hosts are reporting false OS information as a result of obfuscation tools, how will administrators have visibility into their networks and use the tools designed to automate maintenance actions across the network? Additionally, many organizations develop and promulgate standard system configurations to reduce system operating costs and increase operating efficiencies. These standardization efforts could ultimately reduce the effectiveness of obfuscation if those standards become well known; in effect, can obfuscation fool anyone when the attacker already knows the published system standards? The potential benefits of obfuscation, thus, must be considered within the overall context of network operations.

Additionally, information about the OS may be leaked through other means such as in services banners (Lyons, 1998). For example, a telnet or ftp banner shows what OS is running unless the banner is removed or changed (Berrueta, 2003). Attackers can also gain OS information by sniffing DHCP queries from the target (Crenshaw, 2008). In such cases, no amount of OS obfuscation will help.

As a final challenge, OS obfuscation is hard! Making enough changes to normal OS functions that trick an attacker while maintaining a stable system that functions, is no small task (Scambray, McClure, & Kurtz, 2001). Like other obfuscation areas, obfuscation tools must keep up with the many current and emerging fingerprinting techniques.

3.0 Research Design

In order to address the feasibility of an OS obfuscation tool, we simulate a "standard" AF network environment to examine the effectiveness of a currently available OS obfuscation tool against chosen OS fingerprinting tools. The selected OS obfuscation tool is OSfuscation and the selected OS fingerprinting tools are: Nmap (Network Mapper, an active OS fingerprinting tool) & p0f2 (passive OS fingerprinting version 2, a passive OS fingerprinting tool). As a good defense is tailored to the attacker; we now use threat modeling to look at the attacker.

3.1 Threat Model

The actions an attacker takes depend largely on the attacker's goals, how important success of those goals is, and the perceived level of risk in mounting the attack. Looking at our systems from the perspective of the attacker helps us to anticipate and mitigate attack goals (Swiderski & Snyder, 2004). According to JTF-GNO, attackers want intelligence, counterintelligence, targeting information, operations information, technical information, financial and ID theft, intelligence preparation of the battlefield, and resources (bandwidth and processing power) (Joint Task Force Global Network Operations, 2009). Given this, the attacker profiled in this study desires to have the system remain active and presence undetected.

During the scanning phase of the attack, the attacker looks for entry points into the system and the trust level attained after gaining access through a particular entry point (Swiderski & Snyder, 2004). Knowing the OS helps the attacker identify entry points and is necessary to write a payload or use an existing vetted payload. It is our premise that OS obfuscation can reduce attackers to guessing which OS(s) are used in a target network. This in turn could force attackers to use broader attack profiles and thereby increase the likelihood of drawing attention to their attacks. As such, obfuscation may ultimately reduce or prevent the success of their attacks.

In summary, OS obfuscation is most effective against an attacker that 1) lacks physical access, 2) does not have knowledge of the network configuration, 3) specifically targets an OS, 4) wants presence and actions to remain undetected, and 5) wants the system to remain up in order to access or manipulate information. Though the attacker may start from outside the network, our tests simulated an attacker already successfully inside the network to evaluate the obfuscation tool in a worst case scenario.

3.2 Defense Model

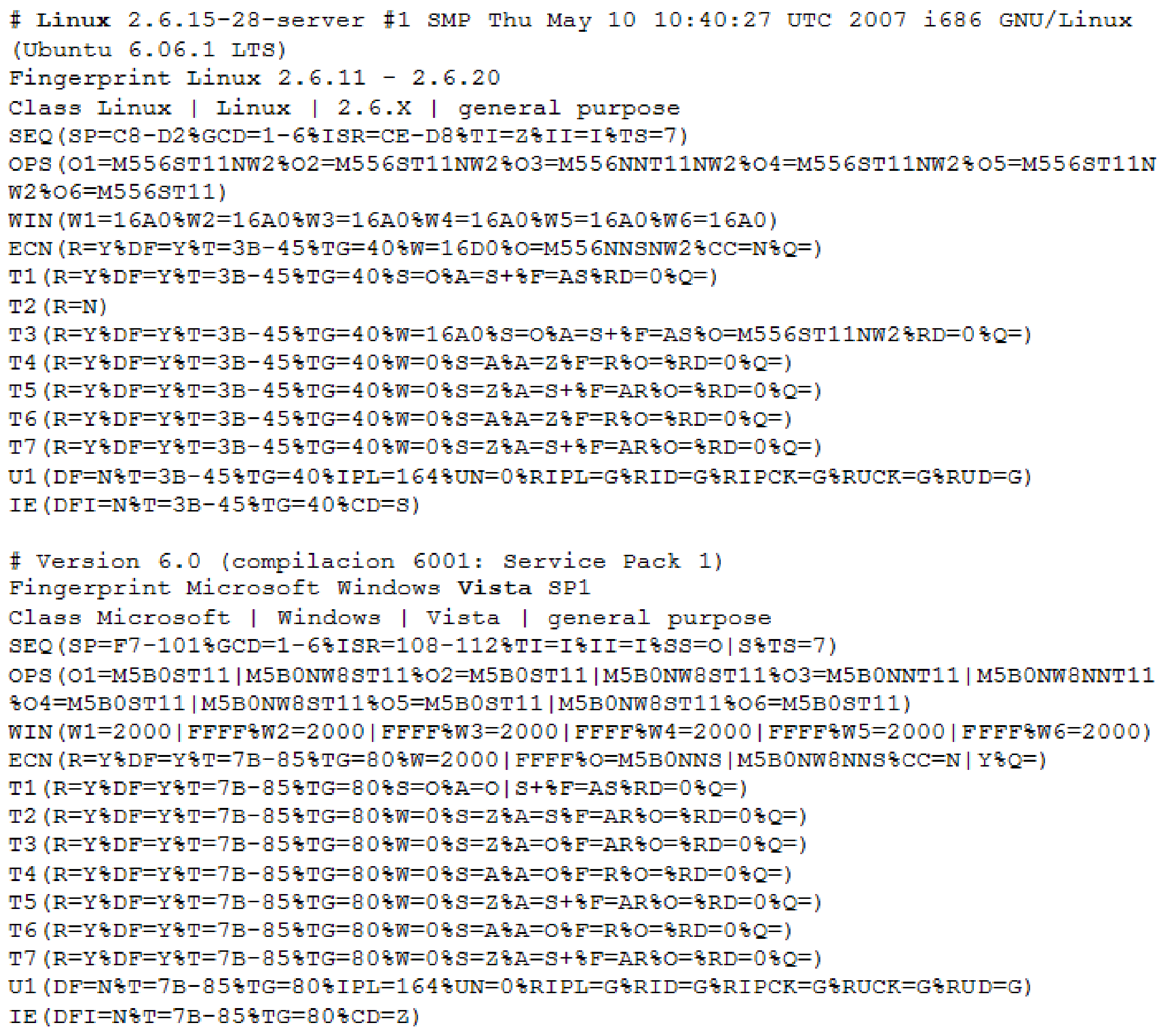
To defend a network against the described attacker, we configure the established test environment to mask the OS of a server providing mail and domain controller services and two clients. Other defensive measures were to patch the systems and install the latest service packs on the clients and servers.

We chose Nmap and p0f2 as the active and passive fingerprinting tools for this test. Nmap is easy to use and updated frequently (Smart, Malan, & Jahanian, 2000). It's also well documented, the winner of the 2003 LinuxQuestions.org Members Choice Awards, and mentioned in several resources as a useful tool for hackers and administrators alike. p0f2 was voted #1 OS detection tool in 2006 (Nmap not part of the survey since it was an Nmap mailing list) (Lyon, 2006). Both tools were readily available at no cost.

Nmap uses raw IP packets to determine what hosts are available on the network, what services (application name and version) those hosts are running, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. Nmap runs on all major OSs (Lyon, 2009). Nmap sends out a series of packets to different ports on the target (Skoudis & Liston, 2006). Specifically it sends up to 16 TCP, UDP, and ICMP probes to known open and closed ports of the target (Lyons). Nmap analyzes the response attributes to generate a fingerprint.

Nmap also measures the predictability of the initial sequence number by sending several SYN-ACK packets to open ports and analyzing how the sequence number changes (Skoudis & Liston, 2006). The predictability of sequence numbers is another differing characteristic of OSs. Other packet probes sent out by Nmap are: SYN packet to open port, NULL packet to open port, SYN|FIN|URG|PSH packet to open port, ACK packet to open port, SYN packet to closed port, ACK packet to closed port, FIN|PSH|URG packet to closed port, and UDP packet to closed port (Skoudis & Liston, 2006).

Nmap has over 1,000 OS fingerprints in its database (Skoudis & Liston, 2006). Figure 1 provides a representative sample fingerprint for Linux and Windows Vista (Lyons, 2009).

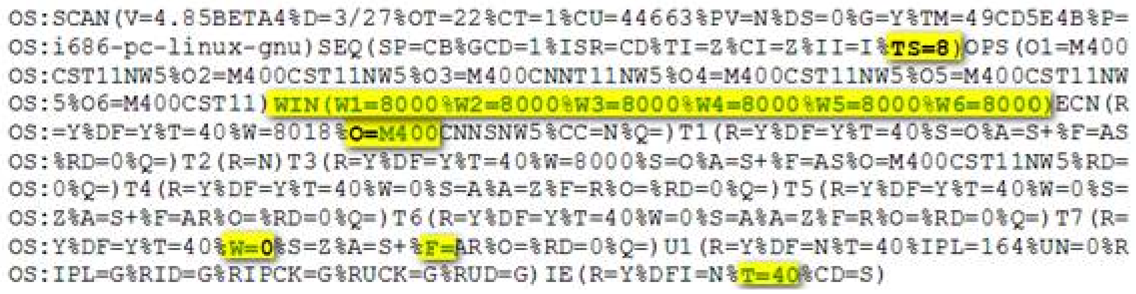


```
# Linux 2.6.15-28-server #1 SMP Thu May 10 10:40:27 UTC 2007 i686 GNU/Linux
(Ubuntu 6.06.1 LTS)
Fingerprint Linux 2.6.11 - 2.6.20
Class Linux | Linux | 2.6.X | general purpose
SEQ(SP=C8-D2%GCD=1-6%ISR=CE-D8%TI=Z%II=I%TS=7)
OPS(O1=M556ST11NW2%O2=M556ST11NW2%O3=M556NNT11NW2%O4=M556ST11NW2%O5=M556ST11N
W2%O6=M556ST11)
WIN(W1=16A0%W2=16A0%W3=16A0%W4=16A0%W5=16A0%W6=16A0)
ECN(R=Y%DF=Y%T=3B-45%TG=40%W=16D0%O=M556NNSNW2%CC=N%Q=)
T1(R=Y%DF=Y%T=3B-45%TG=40%S=O%A=S+%F=AS%RD=0%Q=)
T2(R=N)
T3(R=Y%DF=Y%T=3B-45%TG=40%W=16A0%S=O%A=S+%F=AS%O=M556ST11NW2%RD=0%Q=)
T4(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=3B-45%TG=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=3B-45%TG=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=3B-45%TG=40%CD=S)

# Version 6.0 (compilacion 6001: Service Pack 1)
Fingerprint Microsoft Windows Vista SP1
Class Microsoft | Windows | Vista | general purpose
SEQ(SP=F7-101%GCD=1-6%ISR=108-112%TI=I%II=I%SS=O|S%TS=7)
OPS(O1=M5B0ST11|M5B0NW8ST11%O2=M5B0ST11|M5B0NW8ST11%O3=M5B0NNT11|M5B0NW8NNT11
%O4=M5B0ST11|M5B0NW8ST11%O5=M5B0ST11|M5B0NW8ST11%O6=M5B0ST11)
WIN(W1=2000|FFFF%W2=2000|FFFF%W3=2000|FFFF%W4=2000|FFFF%W5=2000|FFFF%W6=2000)
ECN(R=Y%DF=Y%T=7B-85%TG=80%W=2000|FFFF%O=M5B0NNS|M5B0NW8NNS%CC=N|Y%Q=)
T1(R=Y%DF=Y%T=7B-85%TG=80%S=O%A=O|S+%F=AS%RD=0%Q=)
T2(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)
T3(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=O%F=AR%O=%RD=0%Q=)
T4(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T5(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
T6(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=A%A=O%F=R%O=%RD=0%Q=)
T7(R=Y%DF=Y%T=7B-85%TG=80%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)
U1(DF=N%T=7B-85%TG=80%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)
IE(DFI=N%T=7B-85%TG=80%CD=Z)
```

Figure 1: Sample Fingerprint

After Nmap completes the scan, it matches the collected attributes against the database. Figure 2 provides a representative sample scan return on a subject that is used to match against the database of profiles. Of interest to this study are the attributes changed by the OS obfuscation tool called, OSfuscate. Values present in the example are bolded and highlighted in yellow above and listed in Table 1 (below Figure 2).



```
OS:SCAN(V=4.85BETA4%D=3/27%OT=22%CT=1%CU=44663%PV=N%DS=0%G=Y%TM=49CD5E4B%P=
OS:i686-pc-linux-gnu)SEQ(SP=CB%GCD=1%ISR=CD%TI=Z%CI=Z%II=I%TS=8)OPS(O1=M400
OS:CST11NW5%O2=M400CST11NW5%O3=M400CNNT11NW5%O4=M400CST11NW5%O5=M400CST11NW
OS:5%O6=M400CST11)WIN(W1=8000%W2=8000%W3=8000%W4=8000%W5=8000%W6=8000)ECN(R
OS:=Y%DF=Y%T=40%W=8018%O=M400CNNSNW5%CC=N%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS
OS:%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M400CST11NW5%RD=
OS:0%Q=)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=
OS:Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=
OS:Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%R
OS:IPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)
```

Figure 2: Sample Scan Return

p0f2 works similarly to Nmap in that it analyzes packets from the target and matches the attribute values to a database of OS fingerprints. p0f2 fingerprints the OS on machines that connect to the machine it is running on--incoming connection (SYN mode-default), machines that its host machine connects to--outgoing connection (SYN+ACK mode), machines the host machine can't connect to--outgoing connection refused (RST+ mode), and machines whose communications p0f2 is set to observe--established connection (stray ACK mode) (Zalewski, the new p0f: 2.0.8, 2006). p0f2 fingerprinting accuracy gets better with time as more packets are available for collection and analysis.

OSfuscate prompts the user to select an OS to emulate. Figure 3 is a screen capture of OSfuscate showing the available OS to emulate. OSfuscate makes changes to the following registry settings to match the selected OS. At this time, these are the only modifications OSfuscate makes in order to obfuscate the OS. We list and describe the registry settings in Table 1. In addition, the third column of Table 1 shows our correlation of the registry changes made by OSfuscate to values found in Nmap OS fingerprints.

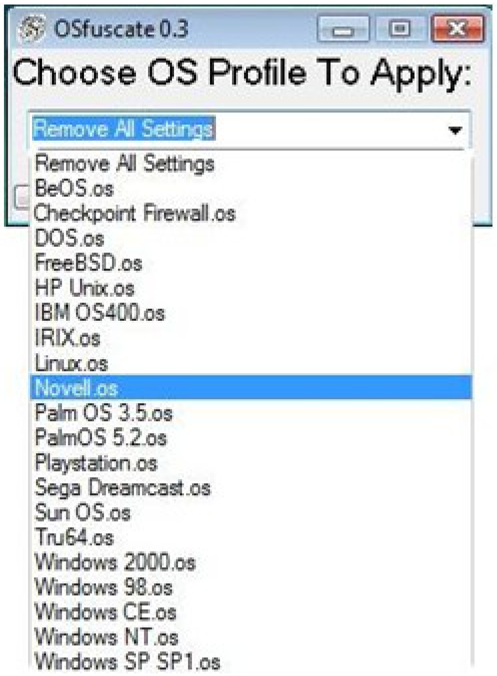


Figure 3: OSfuscate

Table 1: OSfuscate Registry Changes

In the registry under:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\`

Registry Entry	Purpose	Nmap Abbrev
DefaultTTL	Specifies the default Time to Live (TTL) value in the header of outgoing IP packets. The TTL determines how long an IP packet that has not reached its destination can remain on the network before it is discarded.	T
Tcp1323Opts	Determines whether TCP uses the timestamping and window scaling features described in RFC 1323, TCP Extensions for High Performance	TS
EnablePMTUDiscovery	Determines whether TCP uses a fixed, default maximum transmission unit (MTU) or attempts to detect the actual MTU	No match found
TcpUseRFC1122UrgentPointer	Specifies which mode TCP uses for urgent data. The two modes interpret the urgent pointer in the TCP header and the length of the urgent data differently. BSD or RFC 1122	F = U
TcpWindowSize	Determines the largest TCP receive window that the system offers. The receive window is the number of bytes a sender can transmit without receiving an acknowledgment.	W, W1-W6
SackOpts	Enables and disables the Selective Acknowledgment (SACK) feature. SACK is an optimizing feature that lets you acknowledge receipt of individual blocks of data in a continuous sequence, rather than just the last sequence number.	o = S
Interfaces*\MTU	Sets MTU is the size of the largest packet that can be transmitted over the underlying network, including the size of the transport header	o = M

(Purpose taken from Microsoft TechNet (Microsoft, 2009)
*Interface name

Test Environment
The basic structure of the test network followed the idea of defense in depth strategy in an attempt to model a notional AF network. Though virtual machines (i.e. VMware) could have been used, we chose not to do so to eliminate another variable that may have affected test results. Routers and firewalls provide layered protection (demilitarized zone-DMZ). Inside the network we configure two clients with OS obfuscation tool installed on them: Windows XP Professional SP 3 and Windows Vista SP1. In order to generate SMTP, UDP, and HTTP traffic, we configured a second network consisting of a domain controller, mail server, web server, and client.

Passive OS fingerprinting tools must be in a position to receive the packets. To accomplish this, the attacker must either entice the user to come to the attack machine or gain access to a router, switch, or firewall and work to get closer and closer to the final target. In our experiment we simulate the hacker's successful exploitation through the inside firewall and into the switch by dumping the spanned port traffic from the switch to the machines loaded with packet capture software and OS fingerprinting tools. Exploiting the routers and firewalls to gain access to the inside switch is no small feat but as stated earlier this is an ideal scenario for passive OS fingerprinting.

Assumptions and limitations
The placement of the packet capture tool (WireShark) assumes insider threat access or that an attacker has already gained access to a router, firewall, or internal system. The study did not incorporate wireless connections, IPv6 traffic, or IPSec. Anti-virus software was not installed on any of the test machines. Figure 4 illustrates the configuration of the Windows clients, servers, and router. As the attack was simulated for inside access, the firewall and IDS are not depicted in the figure.

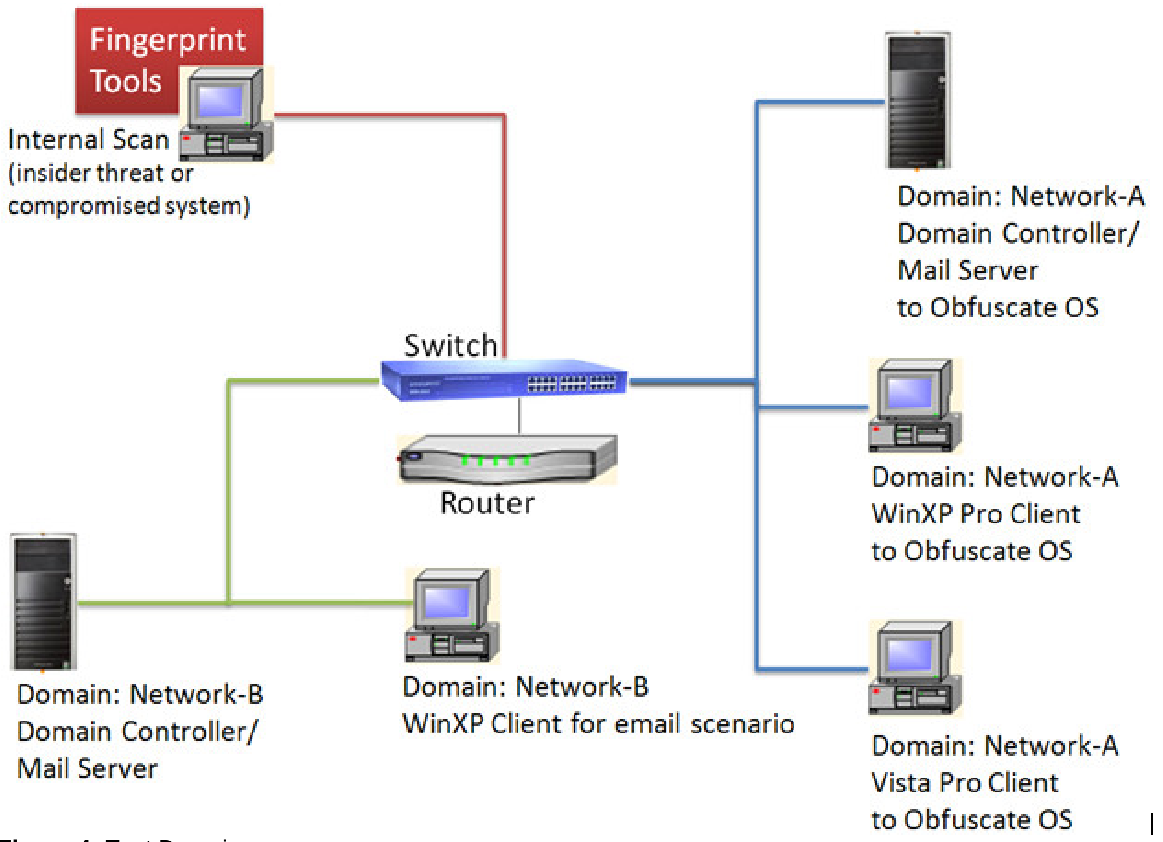

Fingerprint Tools
Internal Scan (insider threat or compromised system)
Switch
Router
Domain: Network-A Domain Controller/ Mail Server to Obfuscate OS
Domain: Network-A WinXP Pro Client to Obfuscate OS
Domain: Network-A Vista Pro Client to Obfuscate OS
Domain: Network-B Domain Controller/ Mail Server
Domain: Network-B WinXP Client for email scenario

Figure 4: Test Domains

Success vs. Failure
We define a test instance as one scan against a target. In the case of our passive fingerprinting tool, one scan is the completion of the test scenario. A successful test instance means the fingerprinting tool could not correctly profile the target OS (to include inconclusive results), and a failure constitutes accurate OS identification by the fingerprinting tool. The reason an inconclusive result is considered a success is that it does not give any OS information away to the attacker. In fact, an inconclusive result is better than an incorrectly guessed OS because the attacker may attempt to run an exploit designed for the incorrect OS which could cause a system crash.

Scenario
The overall flow of the test was to first observe and collect OS fingerprinting scan results without OS obfuscation installed on any of the test machines. The next step was to observe and collect OS fingerprinting scan results with OS obfuscation running on the test machines. Since the passive and active tools use different methods to deduce the target OS, we use two different test steps to match the method. Complete the test steps first with no obfuscation on the target machines and then again after the OS on the target clients has been obfuscated. To generate traffic and create as consistent a scenario as possible, we repeated the actions from Table 2 and captured results after 5, 10, 15, 20, 25, and 30 minutes. We repeated Table 3 actions in order using an aggressive and polite scan.

Table 2: Test Scenario for Passive Fingerprinting

Location	Action
On the Switch	Plug Attack Client into the span
From attack client on Network-B	Logon on then start WireShark capture and p0f2: p0f –i 2 –o out.txt –V –N -l
From control client on Network-B	Logon on as user1
Perform remaining steps 6 times at approximately 5 minutes separation between each start:	
From control client on Network-B	Annotate the time
	Send an email to user1@network-a.local and user2@network-a.local on target network
From XP client on domain Network-A	Log on as user1 and open Outlook
	Send an email to user2@network-a.local on target network not using encryption or digital signing
	Send an email to user1@network-b.local on control network not using encryption or digital signing
	Open browser (IE7) and stop attempts to go to runonce Microsoft
	Navigate to www.network-b.local , click on test link, click back button, and close browser
	Open shared folder on A-DCX.Network-A.local called TestSharedFolder
	View test.rtf file
	Close all programs and log off the system
From Vista client on domain Network-A	Log on as user2 and open Outlook
	Send an email to user1@network-a.local on target network not using encryption or digital signing
	Send an email to user1@network-b.local on control network not using encryption or digital signing
	Open the browser (IE7) and stop attempts to go to runonce Microsoft
	Navigate to www.network-b.local , click the test link, click back button, and close browser
	Open the shared folder on A-DCX.Network-A.local called TestSharedFolder
	View test.rtf file
	Close all programs and log off the system
From attack client on Network-B	Copy p0f output file and add a line at the end of the output annotating completion of test scenario number (1-6)

Table 3: Test Scenario Steps for Active Fingerprinting

Location	Action
Switch	Plug attack client into one of the VLANs
From attack client on Network-B	Logon on and start WireShark capture
	Start an "aggressive" Nmap scan nmap -T4 -A -v -PE -PA21,23,80,3389 192.168.30.11, 192.168.30.2, 192.168.30.15
	Upon completion save scan results
	Start a "polite" Nmap scan nmap -T2 -A -v -PE -PA21,23,80,3389 192.168.30.11, 192.168.30.2, 192.168.30.15
	Upon completion save scan results

4.0 Test Results

OSfuscate successfully obfuscated the target OS against p0f2 and Nmap's aggressive scan, but was only partially successful against Nmap's polite mode. This finding is consistent with Nmap documentation that states though it takes less time to run an aggressive scan, some accuracy is sacrificed (Lyon, 2009). We observed that p0f2 did not fingerprint the Vista client with or without obfuscation enabled. As such, we don't consider the test instances using p0f2 against the Vista client as successful. We used the Windows version of p0f2 which is a version behind the UNIX version. p0f2 may have performed better if we had run the most current version. The creator of OSfuscation discloses on his web site that some of the fingerprints are better than others (Crenshaw, 2008). We speculate that OSfuscation may perform better against Nmap if we chose one of the better fingerprints; however, a ranking of the fingerprints is not readily available.

5.0 Discussion

This experiment showed that current OS obfuscation tools are developed enough to consistently mask OS information on systems running a Windows OS. In our opinion, OS obfuscation implementation could be used in critical systems or on outbound gateway traffic masking packets after they leave the internal network. This placement may deconflict OS obfuscation from blue force system management tools used to maintain the network, and could also confuse passive OS fingerprinting tools monitoring traffic on web servers. However, prior to implementing OS obfuscation on AF networks, system administrator tools and processes must be considered. If OS obfuscation deceives inventory, patch monitoring and installation, and configuration verification tools, it should not be implemented until those conflicts are mitigated. Additionally, if the AF moves to a standards-based configuration instead of the current standard desktop environment, OS obfuscation would provide even more defense.

6.0 Conclusion

Attackers scan target networks to gather system information, find vulnerabilities and fine tune attacks. The OS of a target is intelligence that supports all three goals. Denying the attacker accurate OS information can stop or impede an attacker's mission success. Current OS obfuscation tools are capable of providing some OS obfuscation for AF networks and can add a layer of security in a defense-in-depth strategy. Prior to implementation into AF network defense, current OS obfuscation tools need to be improved and evaluated for impacts on network maintenance tools and processes, to include future initiatives like IPv6.

6.1 Improvements

Several improvements can be made to OS obfuscating tools. Instead of making registry edits, the tool could intercept all packets leaving the system, make the necessary changes to the packet, and send the packets on. The necessary changes would be defined by the user through an interface that allows the user to choose a particular OS to emulate; to choose to constantly change what OS to emulate; to choose not to emulate an OS but to force a fingerprinting tool to return inconclusive results; or to turn off obfuscation. Upon completion of or in conjunction with the development of the improved OS obfuscation tool, we recommend integrating the use of 'chaff', (introducing generated spoofed packets onto the network) with existing obfuscation techniques to further confuse fingerprinting tools.

6.1 Future Research

Several future research topics naturally evolve from this study. We recommend a study that tests OS obfuscation effectiveness under configurations that result from possible AF initiatives: IPv6, IPSec, virtual machines (i.e. VMware), and Common Access Card (CAC) authentication. The testers would evaluate OS obfuscation under each configuration separately then in combination. We also recommend using a passive fingerprinting tool that is more effective under non-obfuscated conditions. After conducting a survey of OSs used in the AF, a good study would be to research how to do OS obfuscation for other OSs starting with an evaluation of a Linux based OS obfuscation tool called Morph (Wang, 2003).

Researchers should also use Attack Trees and/or Value Focused Thinking to determine the optimum way to implement OS obfuscation: inconclusive OS fingerprinting results vs. a consistent false OS presentation vs. a changing false OS presentation.

Another recommended study is to analyze how OS obfuscation affects system administrator network tools. A possible solution and additional study may be to implement a key that triggers the unmasking of an obfuscated packet (Birch, 2009).

Bibliography

Berrueta, D. B. (2003, March 11). *A practical approach for defeating Nmap OS-Fingerprinting.* Retrieved March 12, 2009, from Help Net Security: http://www.net-security.org/article.php?id=406
Birch, S. (2009, February 24). Basic OS Obfuscation. (S. Murphy, Interviewer)
Chairman, Joint Chiefs of Staff. (2006, July 13). *Joint Publication 3-13.4, Military Deception.* Retrieved 11 25, 2009, from Defense Technical Information Center: http://www.dtic.mil/doctrine/jel/new_pubs/jp3_13_4.pdf

Crenshaw, A. (2008). *OSfuscate: Change your Windows OS TCP/IP Fingerprint to confuse P0f, NetworkMiner, Ettercap, Nmap and other OS detection tools*. Retrieved Mar 12, 2009, from Irongeek.com: http://www.irongeek.com/i.php?page=security/osfuscate-change-your-windows-os-tcp-ip-fingerprint-to-confuse-p0f-networkminer-ettercap-nmap-and-other-os-detection-tools

Hernan, S., Lambert, S., Ostwald, T., & Shostack, A. (2006, November). *Uncover Security Design Flaws Using The STRIDE Approach*. Retrieved May 09, 2009, from MSDN Magazine: http://msdn.microsoft.com/en-us/magazine/cc163519.aspx

Hoover, J. N. (2009, April 22). *Pentagon Creating Cyber Warfare Command.* Retrieved May 02, 2009, from InformationWeek: http://www.informationweek.com/news/government/technology/showArticle.jhtml?articleID=217000202&pgno=1&queryText=&isPrev=

Joint Task Force Global Network Operations. (2009, February 17). *JTF-GNO Unclass Threat Brief.* Retrieved April 23, 2009, from Joint Task Force Global Network Operations: https://www.jtfgno.mil/JTF-GNO_Unclass_Threat_Brief.ppt

Lacey, T. (2009, April 04). (S. Murphy, Interviewer)

Levine, A. (2009, April 7). *Official: Millions spent defending Pentagon computers from attack*. Retrieved April 25, 2009, from CNN Politics.com: http://www.cnn.com/2009/POLITICS/04/07/military.computers/

Lyon, G. (2009, January 1). *Chapter 1. Getting Started with Nmap*. Retrieved March 15, 2009, from Insecure.org: http://nmap.org/book/intro.html#id497837

Lyon, G. (2009, January 1). *Chapter 8: Remote OS Detection.* Retrieved April 3, 2009, from Insecure.org: http://nmap.org/book/osdetect.html#id389729

Lyon, G. (2006). *Top 2 OS Detection Tools*. Retrieved Jun 19, 2009, from Insecure.Org: http://sectools.org/os-detectors.html

Lyons, G. (2009, January 1). *Chapter 14. Understanding and Customizing Nmap Data Files.* Retrieved May 08, 2009, from Insecure.org: http://nmap.org/book/nmap-os-db.html

Lyons, G. (1998, October 18). *Remote OS detection via TCP/IP Stack FingerPrinting.* Retrieved May 10, 2009, from Insecure.org: http://nmap.org/nmap-fingerprinting-article.txt

Lyons, G. (n.d.). *TCP/IP Fingerprinting Methods Supported by Nmap.* Retrieved March 19, 2009, from INSECURE.ORG: http://nmap.org/book/osdetect-methods.html

Microsoft. (2009). *Registry Entry Name.* Retrieved May 17, 2009, from Microsoft TechNet: http://technet.microsoft.com/en-us/library/

Miles, D. (2009, April 22). *Secretary Gates presses to boost network security*. Retrieved April 23, 2009, from Air Force Link: http://www.af.mil/news/story.asp?storyID=123145616

Repik, K. M. (2008). Defeating Adversary Network Intelligence Efforts with Active Cyber Defense Techniques. Air Force Institute of Technology.

Scambray, J., McClure, S., & Kurtz, G. (2001). *Hacking Exposed: Network Security Secrets and Solutions.* Berkeley: McGraw-Hill.

Skoudis, E., & Liston, T. (2006). *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses.* Upper Saddle River: Prentice Hall.

Smart, M., Malan, G. R., & Jahanian, F. (2000, Aug 14). *Defeating TCP/IP Stack Fingerprinting.* Retrieved April 23, 2009, from USENIX: http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/smart/smart_html/

Swiderski, F., & Snyder, W. (2004). *Threat Modeling.* Redmond: Microsoft Press.

Wang, K. (2003, July 30). *Frustrating OS Fingerprinting with Morph.* Retrieved February 26, 2009, from Synacklabs: http://www.synacklabs.net/projects/morph/Wang-Morph-DEFCON12.pdf

Washington Post. (2009, Feb 10). *Obama Asks for Review of Online Security*. Retrieved Feb 10, 2009, from Early Bird: http://ebird.osd.mil/ebfiles/e20090210656066.html

Wolfram Mathematica. (2009). Retrieved May 20, 2009, from Wolfram Alpha Computational Knowledge Engine: http://www61.wolframalpha.com/input/?i=obfuscation

Yuill, J., Denning, D., & Feer, F. (2006). Using Deception to Hide Things from Hackers: Processes, Principles, and Techniques. *Journal of Information Warfare* , 26-40.

Zalewski, M. (2006). *p0f 2: Dr. Jekyll had something to Hyde.* Retrieved May 04, 2009, from The "SfR Fresh" Software Archive: http://www.sfr-fresh.com/unix/privat/p0f-2.0.8.tgz:a/p0f/doc/README

Zalewski, M. (2006, September 6). *the new p0f: 2.0.8*. Retrieved April 10, 2009, from p0f2: http://lcamtuf.coredump.cx/p0f.shtml