Toward Undetected Operating System Fingerprinting

Lloyd G. Greenwald and Tavaris J. Thomas
LGS Bell Labs Innovations
{lgreenwald, tjthomas}@lgsinnovations.com

Abstract

Tools for active remote operating system fingerprinting generate many packets and are easily detected by host and network defensive devices such as IDS/NIDS. Since each additional packet increases the probability of detection, it is advantageous to minimize the number of probe packets. We make use of an information-theoretic measure of test quality to evaluate fingerprinting probes and use this evaluation to derive effective probe combinations that minimize probe packets. While the default configuration of Nmap's second generation operating system detection transmits 16 different probe packets, we demonstrate successful fingerprinting with one to three packets. Furthermore, these packets are valid TCP SYN packets to open ports, which are less likely to be detected as fingerprinting probes than malformed packets or packets that are not part of a valid TCP three-way handshake.

1. Introduction

An attacker can use operating system fingerprinting to discover possible security vulnerabilities and evaluate the attack potential of a target machine. Open source tools are publicly available that permit an attacker to gain this intelligence remotely. However, the use of these tools may be easily detected because the default configurations generate too many probe packets or generate packets that are unusual, malformed, or otherwise easily identified as probe packets.

To understand how to build operating system fingerprinting tools that are more difficult to detect we make use of a measure to evaluate fingerprinting tests based on information gain developed in [11]. Fingerprinting tests with high information gain eliminate a lot of uncertainty about the target system while fingerprinting tests with low information gain leave a lot of uncertainty about the target system and are only worthwhile if higher quality tests are too costly. Test cost may be expressed in terms of the number of probes needed for the test and the likelihood that a probe will be detected by IDS/NIDS. Once we understand the quality of individual fingerprinting tests we can evaluate the quality of a probe that enables multiple fingerprinting tests. We can then select a minimum set of probes to perform operating system fingerprinting with low probability of detection.

We provide both analytical and empirical support for building operating system fingerprinting tools that use very few probes yet provide effective operating system classifications. The main contribution of this paper is to demonstrate the use of the theoretical results in [11] to evaluate fingerprinting probe packets. We additionally provide empirical results to substantiate these analytical insights. We demonstrate several sets of probes that provide highly accurate operating system fingerprinting with very few probes. Accuracy is measured in terms of the probability of correctly guessing the target operating system based on the results of a probing experiment. Furthermore, we argue that these probes are unlikely to be detected or modified by defensive devices. We provide accurate solutions using as few as a single probe packet.

We first provide, in Section 2, background material on operating system fingerprinting and theoretical results applying information gain to evaluate the 13 TCP probes used in Nmap version 4.21ALPHA4 [8]. Given the information gain evaluation of Section 2, we develop in Section 3 a set of 23 experiments to determine how few probes we can apply while still providing accurate classification. We empirically evaluate the accuracy of each of these experiments on several target systems. In Section 4 we argue that subsets of accurate probes are unlikely to be detected or modified by defensive devices. Finally we provide a discussion of alternative evaluations and related work in Section 5. An Appendix is included to summarize the analytical techniques developed in [11].

2. Evaluating Information Gain across Tests for Nmap Probes

In order to evaluate a fingerprinting test, we compare how accurately we could guess the classification of a target system before and after performing the test. The difference is called the information gain. The test with the highest information gain provides the most discriminative power in fingerprinting. Information gain

is built on the principles of information theory [20] and is an important tool in building decision tree classifiers [15][17][19]. Information gain is used to select the next test at each step in growing a decision tree. Decision tree classifiers have been used in many fields.

Prior to fingerprinting a target system, we can guess the operating system based on the a priori distribution of operating system classifications, over all possible classifications. After performing a fingerprinting test we can guess the operating system based on the a posteriori distribution of operating system classifications. Let X be a random variable that describes the classification of the operating system of a target system. The *entropy* in X is the amount of uncertainty there is in classifying an unknown system. Let $Test_i$ be a random variable that describes the result of applying test i to the probe responses of a target system. Knowing the value of $Test_i$ might tell us something about the value of X. This can be captured in the *conditional entropy* of X given $Test_i$. A measure of the amount of information we gain about X if we know the value $Test_i$ is called the *mutual information*, or *information gain,* of X and $Test_i$. This can be expressed as the difference between the entropy in the classification before taking the test and the conditional entropy in the classification, conditioned on the value of the test. The fingerprinting test with the highest information gain removes the most uncertainty about the OS classification of a target system.

In [11] we detail a method that uses information gain to evaluate fingerprinting tests. This method is summarized in an appendix below. That paper tackles several hurdles in order to apply information gain in this context. The first hurdle is that information gain is generally computed from collections of training samples of test results from known systems. However, a fingerprinting tool stores information about known systems in a digested signature database rather than as raw training samples. This removes and obscures distribution information. Since a signature database is once-removed from the training samples used to create the database, we must derive calculations to take advantage of the knowledge represented in the signature database and make assumptions about the knowledge that has been lost. Our calculation also resolves issues concerning the use of data that is represented as disjunctive lists and ranges, and the handling of missing test values.

2.1 Nmap Probes

By default, Nmap version 4.21ALPHA4 sends a total of 16 probes (excluding re-transmissions) to a target system and applies tests to the probe responses. The test values are combined into a fingerprint, also known as a signature. The fingerprint of a target system is compared against reference fingerprints in a signature database in order to find matches to help classify the operating system of the target system. Nmap's 16 default probes include six TCP SYN packets to an open port on the target machine (Pkt1-6), three TCP packets with various flags to an open port (T2-T4), three TCP packets with various flags to a closed port (T5-T7), one TCP packet to an open port with the Explicit Congestion Notification (ECN) control flags set, two ICMP ECHO packets (IE), and one UDP packet sent to a closed port to elicit an ICMP port unreachable packet. In this paper we focus on the 13 TCP probes. We do not study UDP and ICMP probes because (1) they are more easily blocked by defensive devices, and (2) our information gain evaluation reveals that they are of marginal value. More detail about the evaluation of ICMP and UDP probes are provided in [10] and [11].

R	Responsiveness
DF	IP don't fragment bit
T	IP initial time-to-live (TTL)
TG	Guessed IP TTL
W	TCP initial window size
S	TCP sequence number
A	TCP acknowledgement number
F	TCP flags
O	TCP options
RD	TCP checksum
TOS	IP type of service
Q	TCP miscellaneous quirks
SP	TCP initial sequence number (ISN) predictability index
GCD	TCP ISN greatest common denominator
ISR	TCP ISN counter rate
TI	IP header ID sequence generation
TS	TCP timestamp option generation

Table 1: Nmap Tests

Table 1 summarizes the tests applied to the responses of the 13 TCP probes of Nmap version 4.21ALPHA4. Pkts 1-6 serve a dual purpose. They are (1) used to determine TCP/IP properties that can only be derived by sequences of timed packets and (2) used as additional sources of TCP initial window size (W) and TCP options (O) data. These probes vary only in TCP options and TCP window fields. Pkt1 is also called T1 and its response is subject to the same tests as responses from probes T2-T7. The sequence tests include testing the TCP initial sequence number (ISN) generation algorithm (tests SP, GCD, and ISR). These tests require responses from at least four of the six

Pkt1-6 probes. Other sequence tests include IP header ID (IPID) sequence generation (TI), requiring responses from three of the six Pkt1-6 probes, and TCP timestamp option generation algorithm (TS), requiring responses from at least two of the six Pkt1-6 probes.

Probes T2-T7 vary in TCP flags, initial window size, and don't fragment bit setting. The responses to each of the T1-T7 probes are tested for responsiveness (R), IP don't fragment bit (DF), IP initial time-to-live (T), guessed IP initial time-to-live (TG), TCP initial window size (W), TCP sequence number (S), TCP acknowledgement number (A), TCP flags (F), TCP options field (O), TCP checksum (RD), IP type of service (TOS), and miscellaneous quirks (Q). Note that the IP initial time-to-live value test (T) requires both one of the T1-T7 probes and the ICMP response from the UDP probe to reconstruct the initial time-to-live value. This additional probe can be avoided by guessing the IP initial-time-to-live value (TG). The ECN probe is subject to the same tests as responses from probes T2-T7, as well as a congestion control (CC) test. A description of these probes and tests is provided in [8].

The different TCP options and initial window sizes sent in the 13 TCP probes can cause a target system to change the window size value in its response packet. Similarly, since TCP options fields are optional, many TCP/IP implementations differ in how they handle them. As shown below, TCP options and initial window size tests are important for accurate fingerprinting.

2.2 Using Information Gain to Minimize Probing Cost

We apply our information gain calculation to the tests of Nmap version 4.21ALPHA4 [8]. Table 2 depicts these results, grouped according to Nmap's 13 TCP probes. Each row corresponds to exactly one probe (except for the IP initial time-to-live (T) test which makes use of the ICMP response to a UDP probe to calculate initial time-to-live). Each column in Table 2 corresponds to a test on the response to that probe. Table 3 depicts the tests that are computed over more than one probe. The entries in these tables correspond to the information gain of the corresponding test computed based on the Nmap version 4.21ALPHA4 signature database. Note that the same type of test may have a different information gain value depending on the probe packet sent to the target. Values that are very similar for the same test may be attributed to noise in the signature database.

The Nmap version 4.21ALPHA4 signature database has 417 entries with total entropy prior to testing of 8.70. Values in Tables 2 and 3 are coded based on the percentage of total uncertainty that is removed by each test. Values in bold font remove at least 50% of the total uncertainty, while values in italicized font remove at least 25%. All other values remove less than 25% of the total uncertainty. The results in these tables assume a target system is equally likely to be any entry in the database and that all possible values of a test for a given entry are also equally likely. Other assumptions or a priori information about classification or test value distributions (e.g. normal distributions over ranges) can be accommodated by adapting the calculations in [11].

Fingerprinting tests with high information gain eliminate a lot of uncertainty about the target system and may be used to build effective fingerprinting tools. Tests with low information gain leave a lot of uncertainty about the target system and are only worthwhile if higher quality tests are too costly. Even so, they are unlikely to be useful independently.

Test cost may be expressed in terms of the number of probes needed for the test and the likelihood that a probe will be detected by IDS/NIDS. Each row in Table 2 corresponds to a collection of tests that cost one probe total, while the tests in Table 3 are tests that require between two and six probes. Our goal is to select the rows from Table 2 and, optionally, tests from Table 3 that provide accurate fingerprinting with low probability of detection. Information gain provides one analytical tool for making this optimization choice. In Section 3 we verify these analytical results with experiments on several target systems using a combination of probes.

From Table 2 we can see that the W and O tests to open ports provide the most information gain. These tests can be achieved with any of the Pkt1-6 probes, the ECN probe, or the T3 probe. The T2 and T4 probes provide less information and the probes to closed ports (T5-T7) provide very little information about W and O. Probes to closed ports often elicit TCP RST responses that can provide some information. Of the remaining tests that can be accomplished with one probe, only the time-to-live tests (T, TG) remove more than 25% of the classification uncertainty. The quality of these tests does not vary much over the applicable probes. To gain the benefits of the most discriminative tests we can choose the ECN, T1 or T3 probes. We can substitute any of the Pkt2-6 probes for the T1 probe, and apply tests R, DF, T, TG, S, A, F, RD, and Q without additional cost.

	R	DF	T	TG	W	S	A	F	O	RD	Q
Pkt 2					**4.76**				**5.39**		
Pkt 3					**4.74**				**5.07**		
Pkt 4					**4.75**				**5.36**		
Pkt 5					**4.76**				**5.29**		
Pkt 6					**4.76**				**4.40**		
ECN	0.09	1.03	*2.57*	*2.57*	**4.61**				**4.89**		0.23
Pkt1/T1	0.68	1.01	*2.55*	*2.55*	**4.71**	0.19	0.29	0.29	**5.27**	0.62	0.62
T2	0.89	1.05	1.81	1.80	1.04	1.13	0.95	1.05	0.02	0.93	0.44
T3	0.71	1.49	*2.76*	*2.76*	**4.51**	1.14	1.31	1.61	**4.33**	0.68	0.26
T4	0.44	1.30	*2.73*	*2.73*	1.48	0.52	1.26	0.76	0.02	0.47	0.02
T5	0	0.98	*2.57*	*2.57*	0.18	0.44	0.20	0.23	0	0.08	0.04
T6	0.30	1.23	*2.67*	*2.66*	0.46	0.44	1.23	0.70	0.02	0.38	0.02
T7	0.55	1.36	*2.77*	*2.77*	0.72	0.90	1.52	0.74	0.02	0.59	0.04

Table 2: Information Gain for Single-Probe Tests in Nmap Version 4.21ALPHA4, Grouped by Probe Packets (each row corresponds to one probe packet)

SP (4)	**GCD** (4)	**ISR** (4)	**TI** (3)	**TS** (2)
3.02	1.45	*2.62*	1.62	*2.67*

Table 3: Information Gain for Multi-Probe Tests (number of probes in parentheses)

Several of the sequence generation prediction tests depicted in Table 3 remove greater than 25% of the classification uncertainty. Each has a varying cost. The SP, GCD, and ISR tests costs at least four probes, while the TI test requires at least three probes and the TS test requires at least two. The costs of these tests overlap each other and the Pkt1-6 test costs. If four probes are used for SP, GCD and ISR then no additional probes are needed for TI and TS. If used for sequence prediction, Pkt1-6 incur an additional cost in terms of delay.

In [11] we further derive information gain over the subfamilies of signature entries corresponding to Microsoft Windows, Linux, and a collection of embedded systems (routers, firewalls, and switches). There are interesting differences that lead to variations over which probes are most effective for detecting systems within these subfamilies. While the W and O tests remain the most discriminative over these sub-families, the W test is more discriminative than the O for Windows and embedded systems while the opposite is true for Linux, by a substantial margin. The TTL tests are discriminative for embedded systems and less so for Linux and Windows. The ISN tests are discriminative for Windows and embedded systems but not discriminative for Linux. The T3 probe is more useful in differentiating Linux versions than it is in general.

The composition of a signature database can have a strong effect on our information gain metric. We review this effect by comparing the tests from the first generation and second generation Nmap, using their respective databases. First generation Nmap has four times as many entries as the current second generation database (1684 vs. 417). In addition to database size, the distribution over types of systems (e.g. Linux, Windows, embedded) changes across databases, as does the distribution of newer versus older systems. The second generation database is skewed toward newer systems. W, O and ISN-based tests remain the most discriminative tests across both databases. However, O is significantly more discriminative in the second generation database than in the first. This may be attributed to the larger proportion of Linux systems in the second generation database. This may also be attributed to changes made between first and second generation Nmap. First generation Nmap does not test the performance-improving selective acknowledgment (SACK) option or the value of the window scale option. Similarly, ISN-based tests are less discriminative in the second generation database than in the first. This may be attributed to the larger proportion of modern OS's (e.g. Linux 2.6.X, Microsoft Windows XP) in the second generation database. Modern OS's have more random ISN generation algorithms, making this test less useful for fingerprinting. Finally, note that in [10] we evaluate Xprobe [2] tests. Xprobe has a signature database of 224 signatures dating from 2005 and earlier. Despite the smaller database and date of the database, the results are similar. In particular, window size, options ordering, and TTL prove especially discriminative.

3. Experimental Evaluation of Nmap Probes

Given the information gain evaluation above, we developed a set of experiments to determine how few probes we can apply while still providing accurate classification. We chose three target machines from current and slightly dated general computing platforms: (1) Microsoft Windows NT 4.00.1381 SP4, (2) Linux Fedora Core 4 kernel 2.6.11, and (3) Microsoft Windows XP Professional SP2. The results of 23 experimental combinations of Nmap probes against each target machine are reported in Table 4. Each row of this table is a probing experiment made up of tests from between 1 to 16 probe packets. The number of probes used in each experiment is given in brackets. We do not include the cost of the UDP probe for test T, as the test TG provides equivalent results without the UDP probe.

Each column of Table 4 corresponds to a target system. The values indicate how accurately the probing experiment classified the target system when choosing from among all 417 possible classifications in the signature database. This includes choosing from among different versions of an OS (e.g. Linux 2.4.22 vs. Linux 2.6.18) and even the same version of an operating system on machines that yield differing reference signatures due to differences such as drivers or hardware.

In order to evaluate the accuracy of each probing experiment we must establish what we mean by a "correct" result. We first run the full set of 16 Nmap probes against our target machine and call this result the "correct" baseline classification. Nmap reports an accuracy percentage for each reference signature that is roughly the number of tests that match the signature divided by the total number of tests, assuming no weighting of tests. To take into account noise in signature entries and test results, we take each signature entry that receives an accuracy percentage of at least 95% and call them all equivalently correct signatures for a target machine. For each experiment reported in Table 4, we take as the classification output all signature entries that receive an accuracy percentage of at least 95%. In other words, there are a set of signatures that we consider "correct" classifications based on the full set of probes and a set of signatures that are considered output classifications for each experimental set of probes. We compare these two sets to yield a new accuracy result as described below. We experimented with other thresholds and found that 95% was reasonable across all experiments. A few experiments in which a threshold of 93% yielded perfect results are marked * in Table 4. In a few experiments there were no signatures that achieved an accuracy percentage greater than 95%. In those cases we took the signatures with the highest accuracy percentage below 95% and considered them the "correct" result. These cases are marked ** in Table 4.

The accuracy of each experiment is reported in terms of the probability of correctly guessing the target OS after probing. In other words, if we randomly choose from among the signatures one that has an accuracy greater than 95% after probing and report it as the classification of the target system, what is the probability of being correct? More specifically, let B_t be the baseline set of correct signatures for target system t (i.e. signatures that receive an accuracy percentage of at least 95% using the full set of 16 Nmap probes). Let E_t be the set of signatures returned as output (with at least 95% accuracy) using experiment E on target system t. The accuracy of experiment E is the probability that we would guess one of the correct signatures if drawing uniformly from the output signatures, namely

$$\frac{|B_t \cap E_t|}{|E_t|}$$

For example, if there are 6 correct baseline signatures and 10 output signatures in an experiment and 4 of these signatures are the same as baseline signatures we have an accuracy of 4/10 or 0.40. If we randomly choose any signature as our target system classification from the 10 returned signatures we have a 0.40 probability of being correct. Note that the accuracy of the full set of 16 Nmap probes is 1.0 by definition.

There is very little probability guessing the target system classification correctly without probing. There are 417 entries in the Nmap 4.21ALPHA4 signature database and the baseline sets (using the 95% accuracy threshold) for our three target systems have 6, 6, and 10 signatures, respectively. Prior to any testing the probability of guessing correctly for the NT and Fedora systems are 6/417 = 0.014 and for the XP system it is 10/417 = 0.024. Thus, we are very unlikely to randomly guess the target system classification without probing. This assumes that each entry in the signature database is equally likely a priori. These probabilities may be modified if distribution information is available.

	Microsoft Windows NT 4.00.1381 SP4	Linux Fedora Core 4 kernel 2.6.11	Microsoft Windows XP Pro SP2
All, Nmap 4.21ALPHA4	1.00	1.00	1.00
1. All, Except W [16]	1.00	0.25	0.48
2. All, Except O [16]	0.86*	0.12	0.91
3. All, Except W,O [16]	0.75	0.07	0.43
4. Pkt1/T1 (W,O) [1]	0.67*	0.86	1.00
5. Pkt1 (W,O,TG) [1]	0.67*	0.86	1.00
6. Pkt1 (W,O,T,TG) [1]	0.50*	0.75	1.00
7. Pkt1 (ALL) [1]	0.50*	0.75	1.00
8. Pkt1 (R,DF,T,TG,S,A,F,RD,Q) [1]	0.14	0.00	0.24
9. T3 (ALL) [1]	0.50*	1.00	1.00
10. ECN (ALL) [1]	0.50*	0.86	1.00
11. T2 (ALL) [1]	0.15	0.00	0.27
12. T4 (ALL) [1]	0.50*	0.00	0.30
13. T5 (ALL) [1]	0.13	0.04	0.26
14. T6 (ALL) [1]	0.50*	0.00	0.30
15. T7 (ALL) [1]	0.14	0.05	0.28
16. Pkt1-6 (SP,GCD,ISR) [4-6]	1.00	0.06	0.00
17. Pkt1-6 (TS) [2-6]	0.00	0.00	0.34
18. Pkt1 (W,O), Pkt1-6 (TI) [3-6]	1.0	0.75	1.00
19. Pkt1(W),Pkt1-6(TI) [3-6]	0.83*	0.11	0.24
20. Pkt1 (W,O,TG), Pkt1-6 (TS) [2-6]	0.67*	0.86**	1.00
21. Pkt1 (W,O,TG), Pkt1-6 (TS, TI) [3-6]	1.00	0.86**	1.00
22. Pkt1 (All), Pkt1-6 (TS,TI) [3-6]	1.00	0.86**	1.00
23. Pkt1 (W,O), Pkt1-6 (TS,TI), T3(All) [4-7]	1.00	1.00**	1.00

Table 4: Probability of Correctly Classifying Target Operating System From 417 Possible Classifications For Each Probing Experiment Using 95% Accuracy Threshold (*1.00 for accuracy threshold of 93%; **No matches above 95% accuracy threshold, but choosing from among the top matches would yield correct classification with this probability.)

We can make many observations about low-cost fingerprinting from these experiments. As expected from our information gain results, probes T2, T4-T7 are not very useful by themselves (experiments 11-15), especially for Linux target systems. Similarly, the ISN prediction tests (experiment 16) are not very useful by themselves, except for Windows NT, where they are perfectly accurate. This is consistent with our information gain results in which ISN is discriminative overall but not for Linux systems. In this case Windows XP has similar behavior to Linux in generating ISNs.

Probes Pkt1/T1,T3, and ECN (experiments 7, 9, and 10) are each individually almost perfectly accurate predictors of target system classification at the cost of a single probe. Interestingly, Pkt1/T1 performs even better if we only test W and O, rather than the full set of tests (experiment 4). Experiments 5 and 6 show that adding the T and/or TG tests to W and O does not improve accuracy, in fact they decrease the accuracy. This is contrary to our information gain results that indicate that time-to-live is a discriminative test, independent of other tests. The results for Pkt1/T1 hold if any probe Pkt2-6 is substituted for Pkt1/T1. T3 is especially effective for Linux but may not be useable in practice, as discussed below.

To further investigate the value of W and O we see in experiments 1, 2, 3, and 19 that removing W and/or O tests from the original set of 16 Nmap probes severely reduces the accuracy of fingerprinting, especially for the Linux target system. For Windows the effects are not as pronounced. Removing O for those systems has less effect compared to removing W. For Linux, removing W and O together has a cumulative effect greater than removing each independently, demonstrating that these are non-overlapping effects.

If the accuracy of the low-cost single probe solutions, Pkt1-6 using W and O or ECN or T3, are not sufficient we can augment these probes. Experiment 23 shows that an effective addition is to combine T3 with W and O from Pkt1/T1 and TS and TI from Pkt1-6. This requires at least 4 probes and obtains results equivalent to the 16 probe Nmap detection for the target systems. Experiments 18 and 21 provide compromises by achieving almost equivalent accuracy with as few as 3 probes. To summarize, high accuracy can be achieved with as few as 1 probe packet and perfect accuracy can be achieved with no more than 3 or 4 probe packets.

4. Effective Stealth

In the previous section we identified low-cost sets of probes that provide highly accurate operating system fingerprinting. In order for these probes to provide effective stealth they must be able to reach the target system and elicit responses unblocked, unmodified, and undetected by defensive devices. Probes that begin with TCP SYN to an open port are less likely to be blocked (i.e. Pkts1-6 and ECN), while many ICMP packets are commonly blocked by default [22]. ICMP may further cause alerts by intrusion detection systems like Snort [18]. Malformed TCP packets (e.g. T3) are likely to scrubbed or dropped by defensive devices.

Smart et. al. [21][23] studied the problem of defeating TCP/IP fingerprinting and found that certain probes and responses used in fingerprinting tests could be modified or blocked without affecting TCP/IP performance. They designed a network scrubber that only allows packets that are part of a standard TCP three-way handshake. Packets that are not part of a valid three-way handshake include T2, T3, T4, T6 and T7. A network scrubber can also perform a canonical ordering of the TCP options. In similar work [12][16] propose normalizing TCP traffic to remove protocol ambiguities for use in network intrusion detection. A canonical ordering of TCP options would affect the information gain of the O tests. However, the option values cannot be scrubbed without affecting TCP performance [21][23], thus, retaining some information gain. Similarly, normalizing the initial window size (W) to defeat operating system fingerprinting may not be worth the performance trade-offs [23]. TCP/IP fingerprinting defeat has also been discussed in [4].

We have empirically demonstrated [10] that defensive devices like the PF network filter [13], despite having traffic normalizing features, are not commonly configured to defeat OS fingerprinting. From our results in Section 3 we empirically observe that, as long as initial window size and TCP options are not normalized, effective fingerprinting is possible with very few probes. Thus, Pkts1-6 and ECN can include the highly discriminative W and O tests and have a high likelihood of being able to reach a target system undetected. As long as initial window size and TCP options are not normalized, effective stealthy fingerprinting is possible.

5. Discussion

We provide both analytical and empirical support for building fingerprinting tools that use few probes yet provide effective operating system classifications. We make use of an information-theoretic measure of test quality to evaluate fingerprinting probes and use this evaluation to derive effective probe combinations that minimize probe packets. We demonstrate successful fingerprinting with as few as one packet. Furthermore, we use valid TCP SYN packets to open ports, which are less likely to be detected as fingerprinting probes than malformed packets or packets that are not part of a valid TCP three-way handshake.

Fyodor [7] provides a discussion of fingerprinting tests, including the history and influences of fingerprinting prior to Nmap. Taleck [22] discusses additional TCP fingerprinting tests. Arkin [1] provides a study of the use of ICMP in fingerprinting and implements these techniques in Xprobe [2]. Zalewski [24] and Auffret [3] describe fingerprinting using passively captured packets instead of active probes. Passive fingerprinting using a single SYN or SYN ACK packet is available in p0f [24]. The information gain metric presented here can help establish the quality of the tests used in these tools and guide the improvement of these tools to be accurate stealthy alternatives to active probing. The analytical tools presented here can be used to evaluate newly developed fingerprinting tests as well as to re-evaluate the quality of existing tests as TCP/IP implementations and the distribution of deployed operating systems change over time.

In [10] we empirically evaluate the robustness of fingerprinting tests to defensive devices. Nmap includes a system for weighing the comparative value of its fingerprinting tests, called MatchPoints [8]. MatchPoints are heuristic estimates of fingerprinting test quality that combine notions of classification value and reliability. Fyodor [9] mentions that some fingerprinting tests (e.g. TOS, SP, ISR) are given low weights if they are commonly affected by network conditions or otherwise viewed as unreliable.

Beverly [5] develops a method to classify operating systems based on passive observation of TCP/IP headers. That study builds classifiers to combine TCP/IP header fields using probabilistic learning. Burroni and Sarraute [6] build an operating system classifier using neural network learning techniques. In that work they try to improve on Nmap's classification results by building and combining a set of hierarchical classifiers based on Nmap's fingerprinting tests. Their classifiers are learned from a dataset created by randomly sampling entries in the Nmap signature database. In our work, we derive probabilities directly from signature database entries, rather than through sampling. The approaches in [5][6] both differ from ours in that we

use signature databases to understand the information provided by fingerprinting tests used by open source tools, rather than build classifiers that combine these tests. In future work we will apply this understanding toward creating improved operating system classifiers.

In the experimental evaluation in this paper, other alternative measures of "correctness" are possible. One alternative is to insert exact fingerprints of the target machines into the database and use those as the correct signatures. However, those fingerprints would differ from the generalized fingerprints in the database and may skew the results. Other alternatives include modifying the threshold accuracy percentage or considering only perfect matches.

Information gain is not a perfect measure of test quality. One well-known weakness is that information gain tends to overestimate the quality of tests that have many possible values. We discuss the impact of this weakness in [11]. Alternatives to information gain that are less biased toward multi-valued tests, but retain other weaknesses, include gain ratio [17] and minimum description length [14].

An interesting application of this work is to develop targeted attacks that use the same probe packets to both test for a specific open port and fingerprint the operating system, simultaneously. Since effective probes are TCP SYN packets, the attacker can complete the three-way handshake and proceed to fingerprint the service on that port before completing the attack. Following-up the TCP SYN probe with a complete handshake further reduces the chance of detection or blocking.

6. REFERENCES

[1] O. Arkin, "ICMP Usage in Scanning: The Complete Know-How," June 2001, <http://www.sys-security.com/archive/papers/ICMP_Scanning_v3.0.pdf>.

[2] O. Arkin, F. Yarochkin, and M. Kydyraliev, "The Present and Future of Xprobe2: The Next Generation of Active Operating System Fingerprinting," Sys-Security Group, July 2003, <http://sys-security.com/blog/published-materials/papers/>.

[3] P. Auffret, "SinFP," Jan. 2007, <http://www.gomor.org/sinfp/>.

[4] D. Barroso Berrueta, "A Practical Approach for Defeating Nmap OS-Fingerprinting," 2003, <http://www.zog.net/Docs/nmap.html>.

[5] R. Beverly, "A Robust Classifier for Passive TCP/IP Fingerprinting," Proc. 5th Passive and Active Measurement Workshop (PAM '04), (Juan-les-Pins, Fr., 2004), pp. 158–167.

[6] J. Burroni and C. Sarraute, "Using Neural Networks for Remote OS Identification," Proc. Pacific Security Conf. (PacSec '05), (Tokyo, Japan, 2005).

[7] Fyodor, "Remote OS Detection via TCP/IP Stack Fingerprinting," Insecure.Org, June 11, 2002, <http:// insecure.org/nmap/nmap-fingerprinting-old.html>.

[8] Fyodor, "Remote OS Detection via TCP/IP Fingerprinting (2nd Generation)," Insecure.Org, Jan. 2007, <http://insecure.org/nmap/osdetect/>.

[9] Fyodor, Personal communications, May 2007.

[10] L. Greenwald and T. Thomas, "Understanding and Preventing Network Device Fingerprinting," Bell Labs Technical Journal Special Issue on Security in IP-based Networks, 12:3, Fall 2007.

[11] L. Greenwald and T. Thomas, "Evaluating Tests used in Operating System Fingerprinting," LGS Bell Labs Innovations Technical Memorandum TM-071207, July 2007, <http://lgsinnovations.com> (under Bell Labs tab).

[12] M. Handley, C. Kreibich, and V. Paxson "Network intrusion detection: evasion, traffic normalization, and end-to-end protocol semantics," in Proc. 10th USENIX Security Symp., Washington, DC, Aug. 2001.

[13] D. Hartmeier, "PF: The OpenBSD Packet Filter," OpenBSD, Oct. 2006, <http://www.openbsd.org/faq/pf/>.

[14] I. Kononenko, "On biases in estimating multivalued attributes," Proceedings of the 14th International Joint Conference on Artificial Intelligence (pp. 1034--1040), Morgan Kaufmann. Hill, 1997.

[15] T. Mitchell, Machine Learning. McGraw Hill, 1997.

[16] V. Paxson and M. Handley, "Defending against NIDS evasion using traffic normalizers," 2nd Int. Workshop Recent Advances in Intrusion Detection, Sept. 1999.

[17] R. Quinlan, "Induction of decision trees," Machine Learning 1 (1986) 81—106.

[18] M. Roesch, "Snort," Jan. 2007, <http://www.snort.org>.

[19] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall/Pearson Education, Upper Saddle River, NJ, 2003.

[20] C. E. Shannon, "A Mathematical Theory of Communication," Bell Sys. Tech. J., 27:3 (1948), 379–423, 27:4 (1948), 623–656.

[21] M. Smart, G. R. Malan, and F. Jahanian, "Defeating TCP/IP Stack Fingerprinting," Proc. 9th Usenix Security Symposium (USENIX '00), (Denver, CO, 2000), pp. 229–240.

[22] G. Taleck, "SYNSCAN: Towards Complete TCP/IP Fingerprinting", CanSecWest , Vancouver B.C., Canada, 2004.

[23] D. Watson, M. Smart, and G. R. Malan, "Protocol Scrubbing: Network Security Through Transparent Flow Modification," IEEE/ACM Trans. Networking, 12:2 (2004), 261–273.

[24] M. Zalewski, "the new p0f: 2.0.8," Sept. 9, 2006, <http://lcamtuf.coredump.cx/p0f.shtml>.

7. Appendix: Information Gain as a Metric for Evaluating Fingerprinting Tests

In the following discussion we outline the calculation of information gain using signature databases as data, including the handling of disjunctive lists and ranges of values. More detail on these calculations is available in [11]. Let X be a random variable that describes the classification of the operating system of a target system. Let X take on n possible values, each with an a prior probability $p(x_j),\ 1 \le j \le n$. The *entropy* in X is the amount of uncertainty there is in classifying an unknown system. This can also be referred to as the information content of knowing the correct classification. It can be expressed as:

$$H(X) = -\sum_{j=1}^{n} p(x_j) \log_2 p(x_j)$$

Let $Test_i$ be a random variable that describes the result of applying test i to the probe responses of a target system. Let $Test_i$ take on n_i values, each with probability $p(test_{ik}),\ 1 \le k \le n_i$. Knowing the value of $Test_i$ may tell us something about the value of X. This can be captured in the *conditional entropy* of X given $Test_i$. Conditional entropy can be expressed as:

$$H(X \mid Test_i) = -\sum_{k=1}^{n_i} p(test_{ik}) \sum_{j=1}^{n} p(x_j \mid test_{ik}) \log_2 p(x_j \mid test_{ik})$$

A measure of the amount of information we gain about X if we know the value $Test_i$ is called the *mutual information*, or *information gain,* of X and $Test_i$. This can be expressed as:

$$H(X; Test_i) = H(X) - H(X \mid Test_i)$$

The fingerprinting test that tells us the most about the operating system classification of a target system is the one that removes the most uncertainty about the classification, namely the test with the highest information gain.

To calculate information gain we need the probability of each classification, $p(x_j),\ 1 \le j \le n$, the probability of each test value, $p(test_{ik}),\ 1 \le k \le n_i$, for test $Test_i$, and the conditional probability of each classification with respect to each test value, $p(x_j \mid test_{ik}),\ 1 \le j \le n$ and $1 \le k \le n_i$. Given a collection of training samples and the assumption that the data are representative of the frequency with which classifications and test values occur in practice, these probabilities can be calculated directly. However, we assume access to a signature database rather than a collection of training samples. Signature databases remove much of the information about distributions over classifications and distributions over test values that are represented in sets of training samples. To make up for this lost information, we need to re-express $p(test_{ik})$ and $p(x_j \mid test_{ik})$ in terms of $p(x_j)$ and $p(test_{ik} \mid x_j)$. These latter quantities are more easily measured from a signature database or other sources.

Through a combination of marginalization and the product rule we obtain:

$$p(test_{ik}) = \sum_{j=1}^{n} p(x_j) * p(test_{ik} \mid x_j)$$

Thus, we can calculate the probability of each test value by summing, over all classifications (entries in the signature database), the multiplication of probability of that classification times the probability of the test value given the classification.

Making use of Bayes rule we can express the probability of a classification given a specific test value, $p(x_j \mid test_{ik})$, as the following ratio:

$$p(x_j \mid test_{ik}) = \frac{p(x_j) * p(test_{ik} \mid x_j)}{\sum_{j=1}^{n} p(x_j) * p(test_{ik} \mid x_j)}$$

These equations allow us to calculate information gain as long as we have the distribution over classifications $p(x_j)$ and distributions over test values given a known classification $p(test_{ik} \mid x_j)$. There is not enough information in a signature database to tell us anything directly about $p(x_j)$. We can, however, make use of information in a signature database to calculate

$p(test_{ik} | x_j)$. To do so we must take into account the four types of database entries:

1. tests that match a single discrete value
2. tests that match one of a disjunctive set of values
3. tests that match one of a range of values
4. tests that match a disjunctive set of discrete values or ranges

Prior to fingerprinting a target system, we can guess the operating system based on the a priori distribution of operating system classifications, $p(x_j)$, over all possible classifications. After performing a fingerprinting test we can guess the operating system based on the a posteriori distribution of operating system classifications, $p(x_j | test_{ik})$. This a posteriori distribution is conditioned on the test result. For tests with a single discrete value per operating system, the tests partition the database into mutually exclusive sets. $p(x_j | test_{ik})$ can then be computed by just considering the set in which x_j falls. When we consider tests with disjunctions or ranges of values, the resulting sets are not mutually exclusive. Each classification may contribute to the probability of more than one test value per test. We must consider each set that a classification can be in (i.e. each value it takes on) and combine the probability of each set in order to derive $p(x_j | test_{ik})$.

We first consider the case in which a test has one discrete value, $test_{ik}$, per classification, x_j. In this case, $p(test_{ik}|x_j)=1$ for that value and $p(test_{ik}|x_j)=0$ for all others. The remaining three cases require information about distributions over test values given a classification. If this information is indicated in the signature database we can use it here. Without that information we assume that each test value specified in a classification is equally likely. Other assumptions or a priori information about test value distributions (e.g. normal distributions over ranges) can be accommodated.

Let $size_{ij}$ be the number of values that $Test_i$ can take on in classification entry x_j. If $Test_i$ is disjunctive this is the sum of discrete values; if $Test_i$ is a range this is the size of the range; if $Test_i$ is a combination of disjunctive values and ranges this is the sum of sizes of each disjunct. If we assume that each test value is equally likely, then $p(test_{ik} | x_j)=1/size_{ij}$ for each test value $test_{ik}$ that occurs in the classification entry for classification x_j and zero for all other test values. We then have:

$$p(test_{ik}) = \sum_{x_j \text{ includes test}_{ik}} p(x_j) * \frac{1}{size_{ij}}$$

One way to interpret this is that each classification contributes a fractional value to the total probability of each test value, weighted by the probability of the classification and the probability of the value within the classification. Note that this subsumes the first case, where $1/size_{ij}=1$ for each test value $test_{ik}$ that occurs in the classification entry for classification x_j and zero for all other test values. We then have $p(x_j | test_{ik})=0$ for each value $test_{ik}$ that is not included in x_j, and for each value $test_{ik}$ that is included in x_j:

$$p(x_j | test_{ik}) = \frac{\frac{p(x_j)}{size_{ij}}}{\sum_{x_j \text{ includes test}_{ik}} \frac{p(x_j)}{size_{ij}}}$$

Recall that there is not enough information in a signature database to tell us anything directly about $p(x_j)$. If we assume that all classifications x_j are equally likely we have $p(x_j | test_{ik})=0$ for each value $test_{ik}$ that is not included in x_j, and for each value $test_{ik}$ that is included in x_j:

$$p(x_j | test_{ik}) = \frac{\frac{1}{size_{ij}}}{\sum_{x_j \text{ includes test}_{ik}} \frac{1}{size_{ij}}}$$

Let $sumsize^{-1}_{ik} = \sum_{x_j \text{ includes test}_{ik}} size_{ij}^{-1}$. For uniformly distributed classifications and uniformly distributed test values per classification, conditional entropy can be expressed as:

$$H(X | Test_i) = -\frac{1}{n} \sum_{k=1}^{n_i} sumsize_{ik}^{-1} \sum_{j=1}^{n} \frac{size_{ij}^{-1}}{sumsize_{ik}^{-1}} \log_2 \frac{size_{ij}^{-1}}{sumsize_{ik}^{-1}}$$

We may then calculate information gain from a signature database as follows:

$$H(X; Test_i) = \log_2 n + \frac{1}{n} \sum_{k=1}^{n_i} sumsize_{ik}^{-1} \sum_{j=1}^{n} \frac{size_{ij}^{-1}}{sumsize_{ik}^{-1}} \log_2 \frac{size_{ij}^{-1}}{sumsize_{ik}^{-1}}$$