A Game-Theoretic Approach for Deceiving Remote Operating System Fingerprinting

Mohammad Ashiqur Rahman, Mohammad Hossein Manshaei, and Ehab Al-Shaer
Department of Software and Information Systems, University of North Carolina at Charlotte, USA
Emails: mrahman4@uncc.edu, manshaei@gmail.com, ealshaer@uncc.edu

***Abstract*—Remote Operating System (OS) Fingerprinting is a precursory step for launching attacks on the Internet. As a precaution against potential attacks, a remote machine can take a proactive counter-strategy to deceive fingerprinters. This is done by normalizing or mystifying the distinguishing behaviors in the packets. However, the unified modification causes significant performance degradation to benign clients. Using a game-theoretic approach, we propose a selective and dynamic mechanism for counter-fingerprinting. We first model and analyze the interaction between a fingerprinter and a target as a signaling game. We derive the Nash equilibrium strategy profiles based on the information gain analysis. Based on our game results, we design *DeceiveGame*, a mechanism to prevent or to significantly slow down fingerprinting attacks. Our game-theoretic approach appropriately distinguishes a fingerprinter from a benign client and mystifies packets to confuse the fingerprinter, while minimizing the side effects on benign clients. Our performance analysis shows that *DeceiveGame* can reduce the probability of success of the fingerprinter significantly, without deteriorating the overall performance of other clients.**

I. Introduction

Fingerprinting is the process of determining the OS of a remote machine. To exploit a vulnerability of a remote machine, an attacker must know the machine's platform and the running services in advance. Otherwise, it would take many more attempts to be successful, which would render the attacker susceptible to an intrusion detection system (IDS). In the fingerprinting process, the adversary usually sends probes to a target machine and analyzes the responses in order to get platform critical characteristics (e.g., OS type, version, installed patches, etc.) required for launching an attack.

Implementations of the networking protocol (TCP/IP) stack often differ among operating systems, even among different versions. Fingerprinters usually leverage the OS specific characteristics of the stacks (often named *OS signatures*) to identify the target's OS. There are two main techniques of fingerprinting: passive and active. A passive fingerprinter secretly listens to the outgoing traffic from the server (e.g., SinFP, p0f, etc. [1], [2]), while an active fingerprinter sends an arbitrary number of crafted probes and analyzes the responses (e.g., Nmap, XProbe2, etc. [3], [4]). In both cases, a number of tests are carried out on the outgoing packets from the target. The test results are compared with the known signatures in order to identify the OS platform of the target.

A number of counter-fingerprinting tools have been proposed in the literature, such as Scrubber, IP Personality, Morph, HoneyD, OSfuscate, IPMorph, etc. [5], [6], [7], [8], [9]. However, these tools experience many practical shortcomings. All of the existing counter-fingerprinting tools alter outgoing packets of each connection irrespective of the sender's behavior. This *exhaustive defense mechanism* results in a significant *performance degradation* mainly with respect to throughput and connection quality. These techniques make modifications on different TCP/IP fields in the packet header, among them some are critical for performance. For example, the modifications on *Initial Window Size*, *Time-To-Live* (TTL), and *Don't Fragment Bit* (DF) can make a significant throughput degradation. If the receive-window size is too small, then the sender will be blocked constantly as it fills out the receive window with packets before the receiver is able to acknowledge [10]. If the initial window size is too large, the sender will spend a long time to retransmit the entire window every time a packet loss is detected by the receiver [11]. Similarly, if the TTL value is too small, packets may not reach the distant destination. Throughput degradation can also happen if the DF bit is reset, because this breaks the maximum transmission unit (MTU) discovery. Moreover, the existing counter-fingerprinting techniques make significant computational overhead for defenders. IP-layer defense mechanisms (*e.g.*, scrubbing) require fragment reassembly and re-fragmentation. Any modification in IP header requires adjustment of the header checksum. All of these works increase the end-to-end communication latency significantly.

Contributions: In order to solve the aforementioned problems of counter-fingerprinting techniques, in this paper we propose a game-theoretic solution for counter-fingerprinting that performs dynamic sanitization on selective packets. Our solution increases the distortion of the fingerprinter's knowledge with acceptable overhead. We define a game model for the fingerprinting process and corresponding countermeasure between a fingerprinter and a target machine. Using a *signaling game* framework [12], we analyze the interactions between the fingerprinter and the target, while obtaining both pooling and separating equilibria. This analysis gives us the opportunity to find potential solutions for evading or delaying fingerprinting, and to get the best strategy for the target. Using our equilibrium analysis, we design a defense mechanism called *DeceiveGame*, to deceive fingerprinting. As our game model takes the performance of benign clients into consideration, *DeceiveGame* balances between security and overhead. We performed experiments to evaluate the performance of *DeceiveGame*, by comparing it to the non-strategic exhaustive counter-fingerprinting mechanism. We found that in some scenarios, our tool reduces the overhead by 60% compared to the exhaustive defense, while it keeps the probability of successful fingerprinting satisfactorily low.

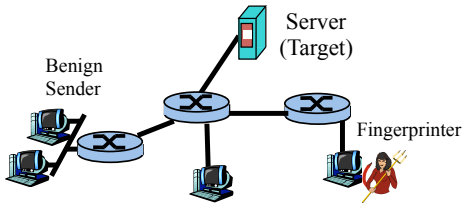

Fig. 1. Fingerprinting system model. A server (target) provides services to several hosts. A fingerprinter tries to figure out the OS of the target.

The rest of this paper is organized as follows: In Section II, we describe the system model along with different fingerprinting tests and probes. In Section III, we present the fingerprinting game model. In Section IV, we present the solution to the game. By using the game theoretic results, in Section V we propose the *DeceiveGame* mechanism. In Section VI, we show the evaluation results of our proposed mechanism. We discuss related works in Section VII and conclude the paper in Section VIII.

II. Fingerprinting: System Model

As shown in Fig. 1, we study a network where the *target* is a server that provides services (e.g., web services) to its client. Hence, many hosts communicate with the target to access the service. We call these hosts *senders*. We assume that an attacker tries to identify the OS of the target in order to use the information for launching attacks against the target. We call the attacker *fingerprinter*. A fingerprinter is also a sender. The target wants to defend against fingerprinting. Thus, the target (as a *defender*) needs to deal with benign senders and fingerprinters simultaneously, while it does not know explicitly about the sender type. In our model, a *probe* denotes a particular type of packet that is sent to the target by benign senders or fingerprinters. The attacker is able to attack different targets and it may have different levels of interest (or benefit) in them. However, the attacker has a certain interest in a particular target. Hence, in this paper, we focus on a single target. The attacker has a certain interest in a particular target, while the target can consider all senders as a single entity (i.e., the target can only care about received probes, not the number of senders). Hence, in this paper, we focus on modeling the interaction between one fingerprinter and one target.

We assume that the fingerprinter should launch and complete the fingerprinting in a period of time smaller than T_M. We define T_M as the maximum time bound for fingerprinting. The reason behind this assumption is that we consider other defense mechanisms like *moving target defense* working on the target, in addition to the counter-fingerprinting mechanism. In the case of a moving target defense mechanism, the target's logical identification is intelligently or randomly changed (e.g., the change of IP address [13], [14]). We assume that the target moves at each T_M interval. If there is no secondary defense mechanism for the target, then T_M can be very large. In a particular time instance, several senders can connect to the target. The sessions of some senders do overlap. A sender can start communication (*i.e.*, fingerprinting) at any time during a particular T_M. Hence, T_F is the period during which the fingerprinting process continues. T_F cannot be more than T_M. Though there is variable communication latency, we assume that the sending time is the same as the receiving time. Table I summarizes the notation used throughout this paper.

TABLE I. A list of symbols.

Symbol	Definition
n, m	Number of tests and probes, respectively
x	Amount of effort (or time) put for fingerprinting
$\mathbb{N}$, $\mathbb{M}$	Sets of tests and probes, respectively
$\mathbb{N}_x$, $\mathbb{M}_x$	Sets of tests and probes used within time x, respectively
g_i	Potential information gain from i^{th} test
G	Average information gain required for successful fingerprint
d_i	It denotes whether or not test i is defended by the target
$\pi(x)$	Probability of success in fingerprinting in x effort
$\sigma(x)$	Cost of fingerprinting in x effort
$\phi(x)$	Information gain possible from the probes sent in x effort, if no defense is taken against them
$\theta(x)$	Belief of the target about the sender type (being a fingerprinter)
$\psi(x)$	Potential defense cost due to taking defense against $\mathbb{N}_x$
Q	Acceptable performance degradation (due to defense)

A. Fingerprinting Tests

We assume that a fingerprinter utilizes n types of *tests* for fingerprinting. Let, $\mathbb{N} = \{1, 2, 3, \cdots, n\}$ is the set of tests. For each test type $i \in \mathbb{N}$, we define the following properties:

1) *Information gain* (g_i): This property denotes the potential information gain [15] from the result of the test i, which in turn expresses the strength of the test in fingerprinting. We briefly describe the calculation of information gain in the appendix. A test is a simple or complex mechanism of checking one (or more) TCP/IP header field value(s), sometimes response behaviors, with known results. Usually a particular test is possible from different probes. These probes often give different information gains for the test. However, we do not count some of these probes as they give significantly lower gains compared to the remaining probes. The probes that we count for a test usually give very similar gains. Hence, we consider the average of these gains as the gain expected from the test. Note that g_i basically represents the information gain considering how much uncertainty the test i can remove if the test is done alone. A test may not give the same information gain given prior successful tests. This can happen, especially, when two or more tests depend on the same TCP/IP fields or response behaviors. We address this issue later in the paper.
2) *Credibility* (v_i): The credibility value ($0 \leq v_i \leq 1$) shows whether the test i is plausible as a TCP/IP packet. The credibility depends on the credibility of the probe(s) used for this test. Moreover, if a test requires a group of probes, the test's credibility is considered as low. Because, the specific order or the fixed collection of the probes can give an easy sign of a fingerprinting act.
3) *Defensibility* (b_i): It is a boolean value denoting whether the test i can be defended, so that the sender does not get potential information gain from the test. There can be some tests, which are not possible to defend, *i.e.*, the way of hiding the information associated to the tests is not possible (or known) to the target.
4) *Impact on performance* (q_i): Defending different tests makes different impact on the performance of the sender and target. For example, defending against *TCP initial sequence number* does not degrade the sender's performance, whereas defending against *initial TTL* might have

TABLE II. A LIST OF TESTS USED TO FINGERPRINT REMOTE OPERATING SYSTEM.

ID (i)	Name	g_i	v_i	b_i	q_i	Related Probes	Comment
1	DF: IP don't fragment bit	1.1	1	1	1	Any of {1-6}	
2	T: IP initial time-to-live (TTL)	2.5	0.9	1	1	Any of {1-12, 15}	
3	W: TCP initial window size	4.7	1	1	1	Any of {1-6, 8}	
4	S: TCP sequence number	1	1	1	0	Any of {1, 7, 8, 12}	
5	A: TCP acknowledgment number	1.2	1	1	0	Any of {7-9, 11, 12}	
6	F: TCP flags	1	1	0	-	Any of {7-9, 11, 12}	q_i is undefined, since $b_i = 0$
7	O: TCP options' order	5	1	1	0	Any of {1-6, 8}	
8	RD: TCP RST data checksum	0.7	1	1	0.5	Any of {1-9, 12}	Costly defense mechanism
9	CC: Explicit congestion notification	0.3	0.9	0	1	{15}	
10	CD: ICMP response code	0.9	0.5	1	0.5	{13, 14}	ICMP packet
11	SP: TCP initial sequence number (ISN)	3	0.5	1	0	Any 4 of {1-6}	Multiple (4) probes

TABLE III. A LIST OF PROBES USED BY A FINGERPRINTER.

j	Name	f_j	Comment
1-6	Pkt1-6	1	SYN, different TCP Options (*e.g.*, MSS, SACK, NOP, WScale, Timestamp) are selected in a order
7	T2	1	DF, Options (MSS, SACK, timestamp)
8	T3	1	SYN, FIN, URG, PSH, same Options as T2
9	T4	1	DF, ACK, same Options as T2
10	T5	0	SYN, same Options as T2, to closed ports
11	T6	0	Similar to T4 but to closed ports
12	T7	0	FIN, URG, PSH, same Options as T2, to closed ports
13-14	IE1-2	0	ICMP (different values for TOS and Code), DF
15	ECN	1	SYN, ECE, CWR, ECN, Options (MSS, SACK, etc.)

very bad impacts on the performance. We refer this as the *defense cost*, which is crucial in case of benign senders. We assume three qualitative values for q_i, *i.e.*, $\{1, 0.5, 0\}$, that define high, medium and no impact, respectively.

Table II presents a number of tests. We cite these tests from [3], since the tests done by Nmap are comprehensive and well known. The table also shows the properties of the tests. For example, the test 7 has information gain 5 and high credibility ($v_i = 1$). This test is defensible ($b_i = 1$) and it will not cause any performance degradation ($q_i = 0$).

B. Fingerprinting Probes

We assume that there are m types of probes. Let, $\mathbb{M}$ is the set of probes, *i.e.*, $\mathbb{M} = \{1, 2, 3, \cdots, m\}$. We denote the credibility of a probe j using f_j. This property shows whether the probe j is credible enough, so that the target would like to respond to it (*e.g.*, whether the probe is a valid TCP/IP packet). For example, if a probe j is sent towards a closed port, its credibility can be assumed to be zero. Any probe without three-way handshaking are also not valid ($f_j = 0$), while TCP SYN based probes are valid TCP/IP packets (*i.e.*, $f_j = 1$). The test's credibility depends on the probe(s) used for the test. Usually the response of a probe can give more than one test result. Hence, the potential gain possible from the probe j is $\sum_{i \in \mathbb{N}_j} g_i$. Here, $\mathbb{N}_j$ is the set of tests ($\mathbb{N}_j \subseteq \mathbb{N}$), which are carried out from the probe j.

Table III shows some of the probes used in [3] with their properties. The set $\mathbb{M}$ includes all kinds of packets in addition to the probes used by a fingerprinter. The information gain may be possible from the responses to the packets sent by a benign sender. Since the benign sender does not do fingerprinting, the accumulated gain from its packets is typically less than that from the probes used by the fingerprinter.

III. FINGERPRINTING: GAME MODEL

In this section, we introduce our game-theoretic model for fingerprinting. Game theory models strategic situations, in which the success of one player in making choices depends on the choices of other players [12]. The key point of our game-theoretic analysis is to consider the strategic behaviors of the target as well as the sender (especially from a fingerprinter's point of view). Moreover, game theory will help us to deal with the lack of knowledge about the type of the sender, *i.e.*, whether it is benign or malicious.

We model fingerprinting using a *signaling game* with a target and a sender (*i.e.*, a benign sender or a fingerprinter) as players. Our choice of the signaling game is based on the *dynamic* and *incomplete information* nature of the fingerprinting attack where the action of one player is conditioned over its belief about the type of the opponent as well as its observation on the actions of other player. As it is shown in Fig. 2, we can represent the signaling game, similar to two connected trees where branches and roots represent the available actions for a given player and the belief about the type of the opponent respectively. We define a fingerprinting game for each connection (or a set of connections that are suspected for coming from a single botnet). We assume that if several connections exist, there are as many fingerprinting games running in parallel. We analyze a single fingerprinting game in the following. We describe different strategies for the fingerprinter and the target. Next, we address how the belief of the target about the type of the sender is updated. Finally, we introduce their payoffs.

A. Strategy Model

Fingerprinter's Strategy: In the fingerprinting game, the fingerprinter determines the amount of information gain it receives by the probes. Let x be the amount of time spent by a sender for communicating with the target. From the fingerprinter's perspective, this is the time already given for fingerprinting the target. We name x as *effort*. Hence, we use *effort* and *time* interchangeably throughout the paper. We assume that the fingerprinter's cost of sending probes is trivial with respect to the time spent for fingerprinting. The value of x is bounded within 0 and T_F.

We use $\hat{x}_i$ to denote the earliest time instance when the target receives a probe whose response potentially gives result for the test i. We know that a single probe can do multiple tests and multiple probes can do the same tests. Hence, $\mathbb{N}_x$ ($\mathbb{N}_x \subseteq \mathbb{N}$) is the set of tests possible from the probes in $\mathbb{M}_x$. The sizes of $\mathbb{M}_x$ and $\mathbb{N}_x$ are nondecreasing with the time (*i.e.*,

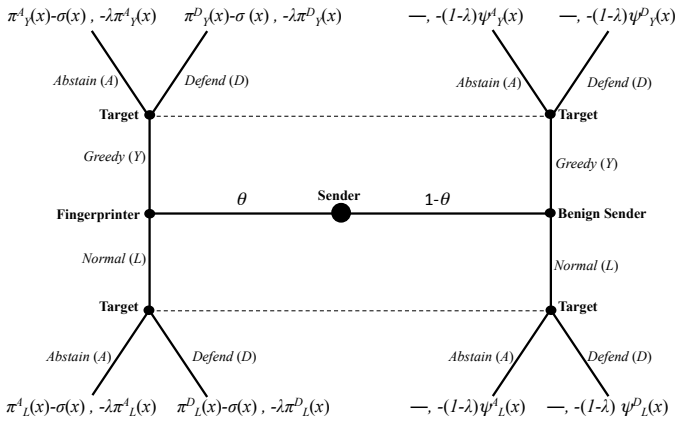

Fig. 2. Representation of a fingerprinting attack as a *signaling game*. The players are a target and a sender. The belief of the target about the sender type (*i.e.*, whether a benign sender or a fingerprinter) is modeled by θ (see Section III-B). Dashed lines show the uncertainly of the target about the type of the sender. The target observes the actions of the sender, *i.e.*, *normal* (L) and *greedy* (Y). The actions of the target (*i.e.*, *abstain* (A) and *defend* (D)) are represented on each leaf of the tree (see Section III-A). The leaves of the tree represent the payoffs of the players (see Section III-C). The first value is the fingerprinter's payoff. Note that we do not show payoff for the sender if the target believes that it is a benign sender.

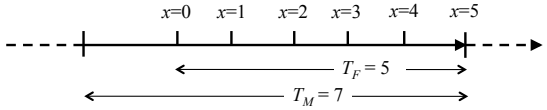

Fig. 3. An example showing the relation between T_M, T_F and x, when the maximum time for fingerprinting the target is $T_M = 7Sec$ and the time of the game between the fingerprinter and the target is $T_F = 5$ Sec.

x). The relationship between the probe set $\mathbb{M}_x$ and the test set $\mathbb{N}_x$ is expressed as below:

$$\mathbb{N}_x = \bigcup_{j \in \mathbb{M}_x} \{i_{j,1}, ..., i_{j,k}\}, \text{where } 0 \leq k \leq m \qquad (1)$$

Here, $\{i_{j,1}, ..., i_{j,k}\}$ is the set of tests that are possible from the probe j. We know that there are few tests that require the responses of multiple probes. A test of this kind is usually associated with a specific set of probes. Hence, for such a test i, $\mathbb{N}_x$ includes i, if and only if each of the probes required for computing i exists in $\mathbb{M}_x$. This is also fair to assume that, in order to defend the test i, it is enough to sanitize the response of one of the probes (*e.g.*, the last received one) associated to the test. To illustrate the above model we consider the case where $m = 7$ and $n = 10$, *i.e.*, $\mathbb{N} = \{1, 2, \cdots, 10\}$ and $\mathbb{M} = \{1, 2, \cdots, 7\}$, as shown in Fig. 3. In this example, we assume T_M is 7 seconds, T_F is 5 seconds, and a probe is sent per second. At $x = 1$, the sender sends (*i.e.*, the target receives) the probe 2, which can give result for the tests 1 and 4. Thus, $\hat{x}_1 = 1$ and $\hat{x}_4 = 1$, while $\mathbb{M}_1 = \{2\}$ and $\mathbb{N}_1 = \{1, 4\}$. At $x = 2$, the sender sends the probe 6, which can give results for the tests 3 and 4. So, $\hat{x}_3 = 2$, while already we saw $\hat{x}_4$. Then, $\mathbb{M}_2 = \{2, 6\}$ and $\mathbb{N}_2 = \{1, 3, 4\}$.

We define the set of actions for the sender (i.e., a fingerprinter) as $s_F = \{Greedy, Normal\}$. When the fingerprinter plays $Greedy$, it is avaricious for information and it sends probes to get more information from the target. On the other hand, with the $Normal$ strategy the fingerprinter sends the probes that can give little or no information gain. Note that the expected behavior from a benign sender is $Normal$, because it communicates to the target only to receive a service. We define a simple threshold-based mechanism to distinguish these two strategies. In this respect, we define G as a portion of the total potential information gain (*i.e.*, $\sum_{i \in \mathbb{N}} g_i$) as the average gain that is required to identify the OS of the target. We use G^B and G^F to denote the total expected gain within the game period with respect to a *benign* sender and a *fingerprinter*, respectively. Obviously, according to the expected behavior, $G^B < G \leq G^F$.

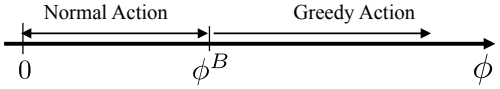

Fig. 4. Strategy of the sender. The sender is playing $Greedy$ if it asks information more than ϕ^B. Otherwise it plays $Normal$.

Let us assume that a fingerprinter selects a probe from the probe set at each step (at a particular x during T_F) of the game. Considering G^B, we define $\phi(x)$ as the observed behavior from the sender at x. We also define $\phi^B(x)$ as the expected behavior from a benign sender. The accumulated potential information gain asked by the sender till x is represented by $\phi(x)$, where $\phi(x) = \sum_{i \in \mathbb{N}_x} g_i$. The expected behavior $\phi^B(x)$ is computed as follows:

$$\phi^B(x) = \sum_{i=0}^{x} \phi_0^B r^i \qquad (2)$$

Here ϕ_0^B is the initial value, while $\phi_0^B r^i$ is the expected increment of gain at time i. The value of r ($0 < r < 1$) is approximately computed considering the equality: $\sum_{i=0}^{T_M} \phi_0^B r^i = G^B$.

The reason behind taking this equality is that with the time, the increase in asking gain (*i.e.*, the rate of asked gain) is expected to be reduced due to the using of same types of packets multiple times. The target expects the total asked gain within a particular time (effort) to be limited. As shown in Fig. 4, we assume that if $\phi(x) > \phi^B(x)$, the sender plays $Greedy$, otherwise it plays $Normal$.

Target's Strategy: We define the set of actions for the target as $s_T = \{Defend, Abstain\}$. The action $Defend$ means that the target defends the test, *e.g.*, by sanitizing the response of the associated probe (or by sending confusing or misleading response), so that the sender does not receive information from the test. In the case of $Abstain$, the original response remains unmodified and the sender receives information associated to the test. The term $d_i(x)$ denotes the target's strategy against the test i at x. We consider $d_i = 1$ when target decides to $Defend$, while $d_i = 0$ in case of $Abstain$. The target's strategy is to select one of these actions, so that (i) the success in fingerprinting by the sender is low (as far as possible) when it is a fingerprinter, and (ii) the defense cost (*i.e.*, the performance degradation experienced by the sender) is reasonably low, if it is a benign sender.

The target may require to take action against a particular test i more than once, since multiple probes often have the same test in common or multiple times the same probe might

be sent. We consider that the subsequent defense strategy for the same test will remain the same (*i.e.*, $d_i(x_i) = d_i(\hat{x}_i)$, where $x_i > \hat{x}_i$) during T_F. Moreover, since a probe usually give results for more than one test, the target often require to take decisions for multiple tests at a time.

The fingerprinting game is played at each time instance x when a probe is received at the target. Then, the optimal strategy of the target will be defined (i.e., whether play *Defend* or *Abstain*) by finding the equilibrium of the signalling game. We present the equilibrium in Section IV-B.

B. Belief Model

In our fingerprinting game, the target does not know whether the sender is a fingerprinter or a benign sender. The strategy of the target depends on its belief about its opponent, as shown in Fig. 2. We define $\theta(x)$ $(0 \leq \theta(x) \leq 1)$ as the belief of the target about the sender type at x. A larger value of θ denotes a higher possibility of being a fingerprinter.

The value of θ is updated with x, particularly after receiving a probe (*i.e.*, watching the sender's action). In order to update $\theta(x)$, we use $\phi(x)$.

$$\theta(x) = \min(1, \frac{e^{(\beta+\phi(x))/G^F} - 1}{e - 1}) \tag{3}$$

Note that we take into account the initial value of the belief (*i.e.*, $\theta(0)$) using the nonnegative value β. The larger is β, the higher is the initial belief about the sender towards being a fingerprinter. Typically, β is very small, even zero, compared to G^F. We consider the exponential function of the potential information gain to compute belief, so that a small increase in gain has more effect on belief if accumulated gain is already high. Because, though the probes used by benign senders also have information gain, the accumulated gain is expected to be low. Other convex function can be used given the target's objectives. The more the target is keen about its security, the higher the initial value of θ. Fig. 5 shows the impact of ϕ and β on θ. A list of known (or expected) benign and malicious hosts can be used to set the prior belief about a sender.

C. Payoff Model

Benefit: Let P be the benefit that the fingerprinter will receive if it succeeds in fingerprinting, and $\pi(x)$ be the *probability of success* after giving x effort. Then, the *expected benefit* for a fingerprinter is $\pi(x)P$. The value of P captures the importance of the target. We assume that both players perceive the same importance of the target. In our model, we assume that P is equal to 1. Hence, the expected benefit of the fingerprinter is $\pi(x)$. We now describe the calculation of $\pi(x)$.

If a probe for a test i is sent at $\hat{x}_i$ effort, we use the term $\pi_i(\hat{x}_i)$ for representing the probability of success received from the test. We already mentioned that the potential information gain from a test may depend on the prior tests. We use the notation $g_i(\hat{x}_i)$ representing the information gain received from the test i, given the tests done prior to $\hat{x}_i$. Hence, $\pi_i(\hat{x}_i)$ is computed as follows:

$$\pi_i(\hat{x}_i) = \frac{g_i(\hat{x}_i) - g_i(\hat{x}_i)d_i(\hat{x}_i)b_i}{G} \tag{4}$$

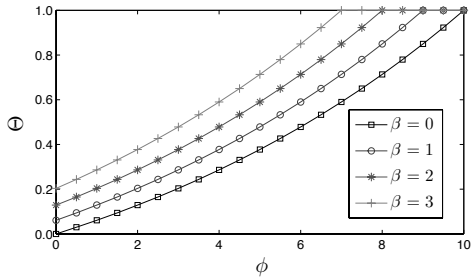

Fig. 5. The impact of ϕ and β on θ. Here, we consider $G^F = 10$.

Here, $g_i(\hat{x}_i)$ is the potential information gain from the test i, if it is successful, taking into account the already successful (not defended) tests till $\hat{x}_i$ ($\{k \mid (k \in \mathbb{N}_{\hat{x}_i}$ and $(d_k(.) = 0)\}$). If we assume that the the dependency is negligible, *i.e.*, the information gain is independent of the prior successful tests, then $\forall_x\ g_i(x) = g_i$. This will simplify Equation (4) as follows:

$$\pi_i(\hat{x}_i) = \frac{g_i - g_i d_i(\hat{x}_i) b_i}{G} \tag{5}$$

The target selects the optimal $d_i(\hat{x}_i)$ at the equilibrium, which depends on b_i, v_i, q_i, $\hat{x}_i$, and the *belief* (θ) about the sender type. The belief follows the prior behavior of the sender based on the probes received by the target. We compute the cumulative probability of success after x effort as follows:

$$\pi(x) = \sum_{i \in \mathbb{N}_x} \pi_i(\hat{x}_i), \text{ where } \hat{x}_i \leq x \tag{6}$$

We consider a *zero-sum benefit model*. The more benefit, *i.e.*, the probability of success, the fingerprinter receives ($\pi(x)$), the less benefit the target obtains ($-\pi(x)$).

Cost: Let $\sigma(x)$ be the cost incurred by the fingerprinter after a given x effort. The cost of processing or transmitting a packet is very trivial to be considered in the fingerprinting cost. Since all the probes and tests are already known, the fingerprinter only cares about the time it gives for fingerprinting. We define $\sigma(x)$ as a simple linear concave function of x as follows:

$$\sigma(x) = \alpha_0 + \alpha_1 x \tag{7}$$

Here, α_0 is the initial cost of attacking a target and α_1 is the cost per time unit. These coefficients take the same unit as of the budget. If we take the initial cost as 0 ($\alpha_0 = 0$) and the cost per time unit as 1 ($\alpha_1 = 1$), then the cost is just the time. The cost equation can be quadratic or even exponential. However, these types of cost equations will make a fingerprinter to fingerprint within a short time, *i.e.*, a higher probing rate, which is subject to catch by an IDS.

We know that defending a test i may incur cost (*i.e.*, the impact on performance), which is denoted by q_i. Hence, the defense cost at $\hat{x}_i$ for the test i is:

$$\psi_i(\hat{x}_i) = \frac{q_i d_i(\hat{x}_i) v_i}{Q} \tag{8}$$

Here, Q represents the defense cost that is intolerable in terms of the minimum level of performance. The use of v_i in computing $\psi_i(x)$ expresses the fact that less credible probes are not expected to be sent by a benign sender. Hence, there

is no need to consider the defense cost due to defending the tests associated to these probes. The cumulative defense cost after x effort is computed as follows:

$$\psi(x) = \sum_{i \in \mathbb{N}_x} \psi_i(\hat{x}_i), \text{ where } \hat{x}_i \leq x \tag{9}$$

Payoff: We model the fingerprinter's payoff by u_F as follows:

$$u_F(x) = \pi(x) - \sigma(x) \tag{10}$$

Note that both $\pi(x)$ and $\sigma(x)$ are normalized values between 0 and 1. We also model the target's payoff by u_T as shown in the below:

$$u_T(x) = -\lambda\pi(x) - (1-\lambda)\psi(x) \tag{11}$$

We use λ ($0 \leq \lambda \leq 1$) to denote the preference of defense over the defense cost. If the target does not have a specific choice, then λ is 0.5.

In Fig. 2, we show the payoffs of the fingerprinter and the target at a particular x for different combinations of strategies. In the case of *Greedy* (Y) strategy of the sender at x, let us assume that the target chooses *Abstain* for the test corresponds to Y. Then the received benefit by the fingerprinter is $\pi_Y^A(x) = g_Y/G$, since $d_Y = 0$ in Equation (4). In this case, the target has no defense cost, *i.e.*, $\psi_Y^A(x) = 0$ (see Equation (8)). If the target chooses *Defend* (*i.e.*, $d_Y = 1$), then the fingerprinter receives the benefit, $\pi_Y^D(x) = (g_Y - g_Y b_Y)/G$ and the target pays the defense cost, $\psi_Y^D(x) = q_Y v_Y$.

IV. Analysis of the Fingerprinting Game

In this section, we first introduce the methodology for solving a signaling game. Then we present the equilibria of our game and their interpretations.

A. Analysis Methodology: Perfect Bayesian Equilibrium

To predict the outcome of the fingerprinting game, one could use the well-known concept of *Nash Equilibrium* (NE): A strategy profile constitutes a Nash equilibrium if none of the players can increase its payoff by unilaterally changing its strategy. In the case of *incomplete information games* (*e.g.*, signaling games), the players are unaware of the payoffs of their opponents. Hence, we adopt the concept of *Perfect Bayesian Equilibrium* [12].

Definition 1: A *perfect Bayesian equilibrium* consists of strategies and beliefs satisfying the following requirements:

1. At each information set, the player with the move must have a belief about which node in the information set has been reached by the play of the game.
2. Given their beliefs, the players' strategies must be sequentially rational.
3. At information sets on the equilibrium path, beliefs are determined by Bayes' rule and the players' equilibrium strategies.
4. At information sets off the equilibrium path, beliefs are determined by Bayes' rule and the players' equilibrium strategies where possible.

Moreover, an equilibrium is called a *separating equilibrium* in signaling game, if each sender type sends a different signal. An equilibrium is called a *pooling equilibrium* if the same signal is sent by all types.

B. Fingerprinting Game: Results

Considering the above definition of the perfect Bayesian equilibrium, we solve the fingerprinting game to find possible *separating* and *pooling* equilibria. Theorem 1 and Theorem 2 identify the best strategies for the players in the fingerprinting game. Due to the limited space, we refer the reader to our technical report available in [16] for the proof of theorems.

Theorem 1. $[(Greedy, Normal),\ (Defend, Abstain)]$[1] *is the only separating equilibrium of the fingerprinting game.*

Theorem 1 shows that at the separating equilibrium the target defends (*i.e.*, plays $Defend$) if the sender (expected to be a fingerprinter) plays $Greedy$. It plays $Abstain$ if the sender (expected to be benign) plays $Normal$. Theorem 2 presents the pooling equilibrium along with necessary conditions. Here, the target believes that the sender plays $Greedy$ for each given type and the expected behavior of the target is $Defend$ at this equilibrium. In this case, the posterior probability of a sender being a fingerprinter is θ. If the senders of both types would play $Normal$ and the posterior probability of a sender being a fingerprinter would be assumed as μ, the expected behavior of the target would be $Abstain$.

Theorem 2. $[(Greedy, Greedy),\ (Defend, Defend),\ \theta]$ *and* $[(Normal, Normal),\ (Abstain, Abstain),\ \mu]$ *are the pooling equilibrium of the fingerprinting game, if the following conditions hold:*

1. $\theta/(1-\theta) \geq (1-\lambda)\ q_Y\ v_Y/(\lambda\ g_Y\ b_Y/G)$
2. $\mu/(1-\mu) \leq (1-\lambda)\ q_L\ v_L/(\lambda\ g_L/G)$
3. $g_L = 0$ *or* $b_Y = 0$

V. *DeceiveGame* Mechanism

The existing counter-fingerprinting mechanisms do not follow any dynamic strategy in defense mechanism. They always take same strategy, *i.e.*, defend a particular probe irrespective of its potential impact, the possible sender type, and the target's belief. We design a mechanism, named *DeceiveGame*, that follows optimal strategy selection based on the equilibrium analysis of the fingerprinting game to deceive attackers. *DeceiveGame* can be implemented on the target or between the sender and the target similar to firewall. This mechanism operates at the network layer, because it requires to work on the TCP/IP headers of the packets. *DeceiveGame* intercepts packets, applies necessary modifications to the outgoing packets, and forwards the packets to the sender.

A. Strategy Selection Mechanism

DeceiveGame follows Algorithm 1 in order to find the strategy against a received probe. Though a benign sender's behavior is mostly $Normal$, it can sometimes behave $Greedy$. On the other side, a fingerprinter can sometimes act $Normal$ to fool the target. Hence, it is important to consider the belief to obtain the optimal strategy. Since in the pooling equilibrium the belief is considered explicitly, we apply Theorem 2 in *DeceiveGame*. We divide $\mathbb{S}$, the set of tests that are possible

[1] The sender strategy profile (a, b) means that it plays a for the type θ and b for the type $1-\theta$. In case of the target, (a, b) means that it plays a following the $Greedy$ action and b following the $Normal$ action of the sender.

Algorithm 1 Strategy Selection

Require: Compute the sets $\mathbb{S}_1$ and $\mathbb{S}_2$, such that $\mathbb{S}_1 = S \cap \mathbb{N}_x$ and $\mathbb{S}_2 = \mathbb{S} \setminus \mathbb{N}_x$.
Require: Apply the same strategy (which has already been taken) for each test $i \in \mathbb{S}_1$.
Require: Sort $\mathbb{S}_2$ considering the following priority:

1) Tests with $b = 0$ comes in the beginning of $\mathbb{S}_2$, irrespective of g.
2) Tests with $b = 1$ goes to the end according to q and v. Tests with the least realized overhead (*i.e.*, qv) goes to the most right.
3) Tests with $b = 1$ and same qv are sorted in increasing order based on g.

Require: Let ϕ' be the current $\phi(x)$. Compute $\theta(x)$ and $\phi^B(x)$.

```
for i ∈ S_2 do
    Add g_i to φ'.
    if b_i is false then
        d_i = 0.
    else
        if (v_i = 0 or q_i = 0) then
            d_i = 1.
        else
            if (Sender Plays Greedy, i.e., φ' > φ^B) then
                d_i = 1, if the conditions of Theorem 2 are true;
                otherwise, d_i = 0.
            else
                if (Sender Plays Normal, i.e., φ' ≤ φ^B) then
                    d_i = 0, if the conditions of Theorem 2 are true;
                    otherwise, d_i = 1.
                end if
            end if
        end if
    end if
    Derive D such that D = {d_i | i ∈ S}.
    Update N_x considering S_2.
    Update φ(x).
end for
```

from the received probe, into two sets: $\mathbb{S}_1$ and $\mathbb{S}_2$. The first set consists of the tests, which are already seen in the earlier received probes, while the second set represents the new tests. Since the actions corresponding to the tests in $\mathbb{S}_1$ have been already selected and applied on the tests, the same strategy will be followed for them. Hence, we need to find the actions for the tests in $\mathbb{S}_2$. We sort the tests in $\mathbb{S}_2$ considering their properties as shown in Algorithm 1. We defend the packets, which are not valid ($v = 0$) or have no defense cost ($q = 0$). For the rest of the tests, we apply Theorem 2 to select the optimal strategies. Trivially, the value of μ is the same as θ.

Note that the values for different game parameters, such as G, ϕ_0^B, λ can be defined based on the guidelines and prior knowledge. For example, the small value of λ shows that the target is very sensitive to performance degradation. More discussions about the parameter selections can be found in [16]. In order to defend a probe, *DeceiveGame* modifies the appropriate fields (or behaviors) in the response to the received probe based on the strategy set $\mathbb{D}$.

B. Implementation Issues

The tests can be defended in different ways. In some cases, *DeceiveGame* can do normalization in the responses similar to the methodology of protocol scrubber [5]. In some cases, it can choose random values. For the probes having low credibility, *e.g.*, the probes sent without three way hand-shaking or towards closed ports, can be defended easily by sending no reply. Since these defense methods are not the aim of the paper, their description is not given here.

VI. Evaluation

We evaluated *DeceiveGame* in two stages. First, we analyzed the performance of the tool against conventional fingerprinting mechanisms. We also analyzed some important characteristics of the defense mechanism. Then, we verified whether our tool can evade Nmap [3].

TABLE IV. Parameter Values Used in Simulations

n	m	G	G^B	G^F	ϕ_0	β	Q	λ	T_M
16	20	7	0.5 $\times G$	1.5$\times G$	0.5 $\times$ G^B	0	3	0.5	25s

A. Performance and Characteristic Analysis

We analyzed *DeceiveGame* using emulation and simulation experimentations. Various fingerprinting scenarios were created where fingerprinters and benign senders send packets to the target. Experiments were done under three different options: (1) without defense (d = 0), (2) using exhaustive defense (d = 1), which represents the existing defense strategies, and (3) using our proposed *DeceiveGame*. We evaluated the results using the following metrics: (1) *effectiveness* that measures the probability of fingerprinting success, (2) *overhead* that measures the potential defense cost realized by the target as a result of sanitizing responses, and (3) *intrusiveness* that measures the number of defended probes.

Methodology: In our experiments, we created a random traffic that contained all the fingerprinting probes as shown in Table III. The probe type can be selected based on four fingerprinting models (*i.e.*, attack models) as follows: (1) *naive fingerprinter* that selects probes in an increasing order of information gain, (2) *greedy fingerprinter* that first selects the probes with higher information gain, (3) *random fingerprinter* that selects probes randomly from the unused tests using uniform distribution, and (4) *hybrid fingerprinter* that is a combination of the previous models. In hybrid model, a fingerprinter can start attacking with random model and next it can choose greedy model after spending 50% of its possible efforts. We used the random model in the experiments unless the fingerprinting model is explicitly specified. The properties of the tests and associated probes are the same as those shown in Table II and Table III, respectively. We assume that all other packets do not provide any information gain (*i.e.*, entail no fingerprinting test). We generate benign traffic using archives of web traffic [17]. Table IV shows the values that we considered for different game parameters. We already conducted simulations with other values and the results showed the similar behaviors as in the following.

Performance Analysis: Fig. 6(a) shows that the potential success of fingerprinting (*i.e.*, $\pi(x)$) in the case of *DeceiveGame* is close to that of the exhaustive defense mechanism and it is within the 20-25% difference. This success probability does not allow for a successful determination of the target. Note that repeating a probe multiple times does not increase the fingerprinting success as it does not contribute to the information gain. While *DeceiveGame* performs reasonably well in defeating fingerprinting, it remains highly efficient in reducing the defense cost (*i.e.*, $\psi(x)$) incurred by the exhaustive defense mechanism. Fig. 6(b) and Fig. 6(c) show the associated results. Fig. 6(b) shows that *DeceiveGame* reduces the overhead up to 60% as compared to the exhaustive mechanism. Fig. 6(c) shows that DeceiveGame defends fewer number of probes (as well as tests) than the exhaustive mechanism, especially in case of benign senders. These results prove that our mechanism is

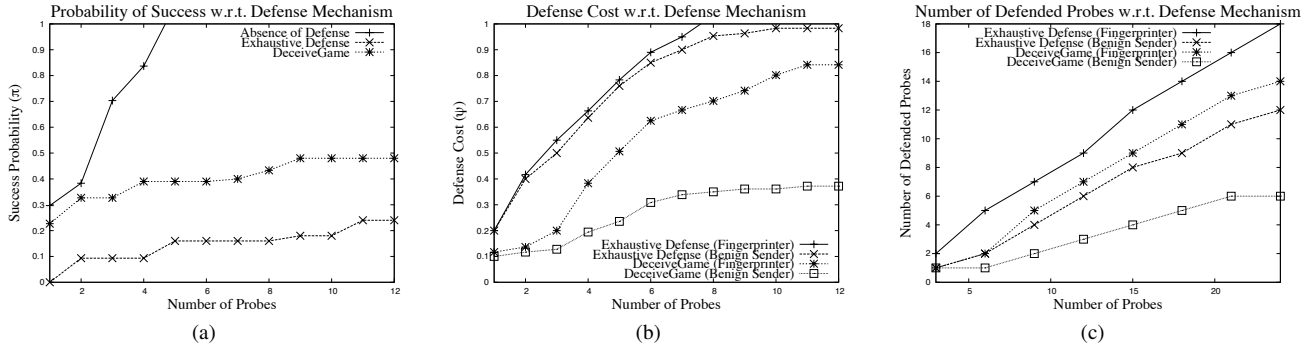

Fig. 6. (a) Comparison of general defense mechanisms w.r.t. success of fingerprinting, (b) Potential defense cost in different defense mechanisms, (c) Number of probes/tests defended in different defense mechanisms.

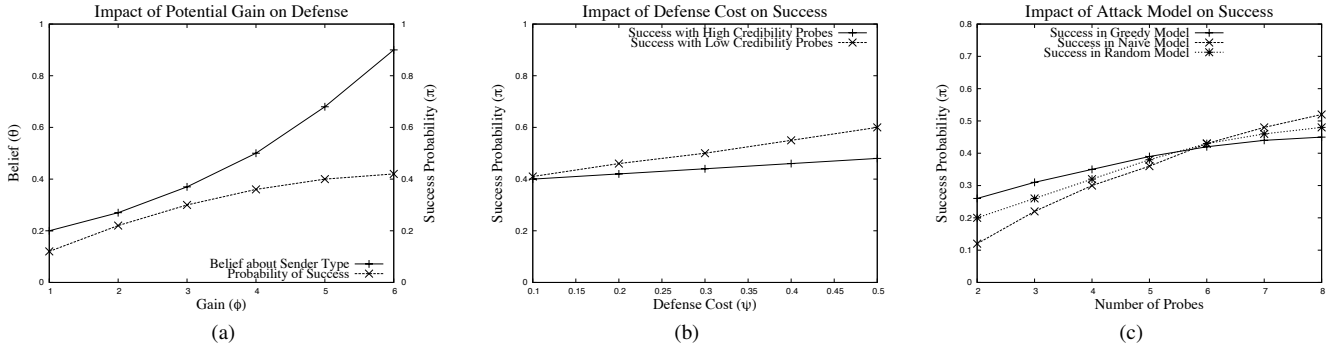

Fig. 7. Impact of (a) the potential information gain on the probability of success, (b) the potential defense cost on the probability of success, (c) different attack models (*i.e.*, the order of the probes based on potential gain) on the probability of success.

able to discriminate between benign senders and fingerprinters, and it adjusts the defense mechanism accordingly in order to outperform conventional fingerprinting countermeasure mechanisms.

Characteristic Analysis: We analyzed the impact of potential information gain of the fingerprinter on the target's belief about the sender type and thus on the fingerprinter's success. We observed (as shown in Fig. 7(a)) that with the increase of the potential gain, the belief (θ) that the sender is a fingerprinter. The impact of the potential defense cost on the success probability is shown in Fig. 7(b) in two different credibility levels. The more is the cost (*i.e.*, performance degradation) for defending a test, the higher is the success probability from the test. However, in case of a low-credible probe, this impact is insignificant (as discussed in Section III-C). The impact of attack model (*i.e.*, greedy, naive, and random) on the fingerprinting success is presented in Fig. 7(c). We observed that sending probes with higher information gain (e.g., in greedy model) obtain lower success probability in the long run. Because even a small increase in the potential gain might create a significant impact on the belief, given that the accumulated gain is high.

B. Accuracy Verification

We implemented the prototype of *DeceiveGame* using *C* programming language. We used *Win Divert* [18], a user-mode packet capture-and-divert package for windows, in this implementation. The implemented prototype was installed in a host, A, where Windows 7 OS was running. We installed Nmap [3] in another host, B. We tried to fingerprint A from B in two ways: with and without executing *DeceiveGame* on A. The outputs of both cases are presented in Table V. We found that *DeceiveGame* successfully disrupted Nmap from understanding the OS of A. In this experiment, we observed that the belief about the sender (i.e., θ) reached high (close to 1) within 3 probes sent by Nmap. As a result, most of the probes are defended.

TABLE V. THE OUTPUT (PARTIAL) OF NMAP

OS Fingerprinting Command: nmap -O -v 152.15.106.207

Nmap Output without *DeceiveGame*:

MAC Address: D4:BE:D9:9A:74:3A (Dell)
Running: Microsoft Windows 7 | Vista | 2008
OS details: Microsoft Windows 7 Professional, · · · · · · · · ·

Nmap Output while *DeceiveGame* is running:

MAC Address: D4:BE:D9:9A:74:3A (Dell)
Too many fingerprints match this host to give specific OS details

VII. RELATED WORK

Different tools proposed in order to evade from fingerprinting. following either of two basic approaches: (i) converting ambiguous traffic from a heterogeneous group of hosts into sanitized packets, and (ii) providing the ability to have different (OS) personalities. The TCP/IP stack normalization approach was introduced with Scrubber [5], which is the first well-known counter-fingerprinting tool. It removes ambiguities from

TCP/IP traffic that gives clues to a host's operating system. It normalizes the outgoing traffic mainly on the fields, such as the order of TCP options, the pattern of initial sequence numbers, the initial window size. Changes of some fields like DF flag, window size, TTL affect the performance. IP Personality [7] introduced the idea of emulating different personalities to evade fingerprinting tools. Different counter-fingerprinting tools like [8], [19], [6], also follow the same idea. IP personality especially is a patch for Linux 2.4 kernel that gives an user the ability to change different characteristics of TCP/IP stack. However, the emulated TCP/IP stack can make the communication weaker than the original one.

IpMorph [6] is built combining the concept of Scrubber and IP Personality. Its objective is to hide the computer's identity from the fingerprinting tools and to impersonate the identity of a different less interesting system, so that the fingerprinter finds the system less appealing. The main drawback of this tool is its complex mechanism. OSfuscate [9] is a small counter-fingerprinting tool that allows to change some registry values in order to reconfigure the TCP/IP stacks. Its usability is limited only to Windows XP/Vista. However, none of these above mentioned tools is strategic in defense mechanism and they treat a benign sender similarly to a fingerprinter.

VIII. Conclusion

Defense against remote OS fingerprinting is the precaution against potential remote attacks. Since a sender can be a benign one rather than a fingerprinter, always defending can cause considerable performance degradation in the case of benign senders. Our proposed game-theoretic defense mechanism, *DeceiveGame* takes the sender type into consideration and performs selective defense actions based on the belief about the type. Therefore, the proposed mechanism is suitable for defense against an unknown opponent. Most importantly, *DeceiveGame* works differently against a fingerprinter and a benign sender. It keeps the fingerprinting success low in case of a fingerprinter, while it creates less performance degradation in case of a benign sender. We evaluated our tool prototype by simulating different known probes. We found that the tool outperforms conventional defense mechanisms by reducing the overhead up to 60%, while the probability of fingerprinting success remains reasonably low.

References

[1] Patrice Auffret. Sinfp, unification of active and passive operating system fingerprinting. In *Journal in Computer Virology*, volume 6, 2010.

[2] M. Zalewski. The new p0f: 2.0.8. 2006. Available in http://lcamtuf.coredump.cx/p0f.shtml.

[3] Fyodor. Remote os detection via tcp/ip fingerprinting (2nd generation). 2007. Available in http://insecure.org/nmap/osdetect/.

[4] O. Arkin and F. Yarochkin. A fuzzy approach to remote active operating system fingerprinting. 2003. Available in http://www.sys-security.com/archive/papers/Xprobe2.pdf.

[5] M. Smart, G. R. Malan, and F. Jahanian. Defeating tcp/ip stack fingerprinting. In *USENIX Security*, Aug 2000.

[6] G. Prigent, F. Vichot, and F. Harroue. Ipmorph: Fingerprinting spoofing unification. In *Journal in Computer Virology*, volume 6, Oct 2009.

[7] Roualland and Jean-Marc Saffroy. Ip personality. 2001. Available in http://ippersonality.sourceforge.net.

[8] X. Zhang and L. Zheng. Delude remote operating system (os) scan by honeyd. In *Workshop on Computer Science and Engineering*, Oct 2009.

[9] Adrian. Osfuscate 0.3. 2008. Available in http://www.irongeek.com.

[10] K. Poduri and K. Nichols. Simulation studies of increased initial tcp window size. In *Internet Draft by IETF*, 1998.

[11] Tcp optimizer, speed guide. 2011. Available in http://www.speedguide.net/tcpoptimizer.php.

[12] R. Gibbons. Game theory for applied economics. In *Princeton University Press*, 1992.

[13] J. Michalski. Network security mechanisms utilizing network address translation. In *Journal of Critical Infrastructures*, volume 2, 2006.

[14] E. Al-Shaer, Q. Duan, and J. H. Jafarian. Random host mutation for moving target defense. In *SECURECOMM*, 2012.

[15] L. Greenwald and T. Thomas. Evaluating tests used in operating system fingerprinting. In *LGS Bell Labs Innovations*, 2007.

[16] M. Rahman, M. Manshaei, and E. Al-Shaer. A game-theoretic solution for counter-fingerprinting. Technical Report, Available at http://www.manshaei.org/files/TR-DeceiveGame.pdf.

[17] The internet traffic archive. 2008. Available in http://ita.ee.lbl.gov/html/traces.html.

[18] Basil. Windivert 1.0: Windows packet divert. 2012. Available in http://reqrypt.org/windivert.html.

[19] K. Wang. Frustrating os fingerprinting with morph. 2004. Available in http://www.synacklabs.net/projects/morph/.

Appendix

The calculation process of information gain from a test presented in [15] is very briefly described here. Let X be a random variable that describes the classification of the OS of a target system. The entropy in X is the amount of uncertainty existing in classifying an unknown system. Let X can take n possible values, where n is the number of all possible operating systems. Each value has the probability $p(x_j)$, $1 \leq j \leq n$. So, the entropy is calculated as follows:

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

Let $Test_i$ be a random variable that describes the result of applying the test i to the probe response from a target system. The conditional entropy of X (*i.e.*, $H(X|Test_i)$) is calculated given that $Test_i$ is successful. The mutual information of X and $Test_i$ is the amount of information one gains about X if it knows the result, $Test_i$. This is termed as *Information Gain*. This can be simply defined as the difference between the entropy before taking the test and the entropy conditioned on the value of the test. The conditional entropy $H(X|Test_i)$ is computed as follows, where $Test_i$ is considered to take on n_i values, each ($test_{i_k}$) with probability $p(test_{i_k})$, $1 \leq k \leq n_i$:

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

Thus, the Information gain on X by $Test_i$ is

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

In our model, g_i denotes $H(X; Test_i)$. To calculate information gain one needs three probabilities: $p(x_j)$, $p(test_{i_k})$ with $1 \leq k \leq n_i$, and $p(x_j \mid test_{i_k})$ with $1 \leq j \leq n$ and $1 \leq k \leq n_i$. The information gain shown in Table II is derived by assuming that each OS is equally likely $p(x_j) = 1/n$. We know that OS distributions are not uniform usually. Though the given OS distribution might change the gain values [15], which in turn may change the course of playing the game, our game model and the solution process remain the same. Moreover, the administrator of the target network knows the OS distribution in the network, while the attacker may not know the correct one. Hence, the target has an advantage over the attacker in playing the game.