Dempster-Shafer Evidence Combining for (Anti)-Honeypot Technologies

Osama Hayatle[1], Amr Youssef[1], Hadi Otrok[2]
[1]Concordia Institute for Information Systems Engineering
Concordia University, Montreal, Quebec, Canada
[2] Electrical and Computer Engineering Department
Khalifa University of Science, Technology & Research, Abu Dhabi, UAE
Email:{youssef, o_hayat}@ciise.concordia.ca, Hadi.Otrok@kustar.ac.ae

Abstract

Honeypots are network surveillance architectures designed to resemble easy-to-compromise computer systems. They are deployed with the aim to trap hackers in order to help security professionals capture, control, and analyze malicious Internet attacks and other activities of hackers. A botnet is an army of compromised computers controlled by a bot herder and used for illicit financial gain. Botnets have become quite popular in recent Internet attacks. Since honeypots have been deployed in many defense systems, attackers constructing and maintaining botnets are forced to find ways to avoid honeypot traps. In fact, some researchers have even suggested equipping normal machines by misleading evidence so that they appear as honeypots in order to scare away rational attackers. In this paper, we address some aspects related to the problem of honeypot detection by botmasters. In particular, we show that current honeypot architectures and operation limitations may allow attackers to systematically collect, combine and analyze evidence about the true nature of the machines they compromise. In particular, we show how a systematic technique for evidence combining such as Dempster-Shafer theory can allow botmasters to determine the true nature of compromised machines with a relatively high certainty. The obtained results demonstrate inherent limitations of current honeypot designs. We also aim to draw the attention of security professionals to work on enhancing the discussed features of honeypots in order to prevent them from being abused by botmasters.

Index Terms

Honeypots, Anti-Honeypot Technology, Botnets, Dempster-Shafer Theory

I. Introduction

A botnet is an army of compromised computers [1] controlled by a bot herder, also known as the botmaster, and used to perform illegal Internet-based malicious activities such as spam spreading, identity theft, and distributed denial of service (DDoS) attacks. Botnets have introduced a new kind of organized underground business that allows criminals to rent a botnet or part of it [2] to be used for illicit activities. This potential financial revenue has raised botnets to become one of the most important threats in the Internet security world [3]. One way to reveal the secrets of this underground world is through the use of honeypots, which are traps designed to lure attackers by providing what appear as easy-to-compromise machines and networks. By trapping attackers into these honeypots, security professionals are able to capture, control, and analyze malicious attacks and other activities of the hackers underground communities. The wide deployment of honeypots had forced attackers who construct and maintain botnets to find ways to avoid these honeypot traps since falling into these traps may lead to their detection or reduction in their financial gain [4]. If hackers are able to detect honeypots, then attacking honeypots may also

provide hackers with valuable information about their observers, the same way that honeypots were designed to provide the security professionals with valuable information about botmasters. Thus, hackers are presumably actively working on finding ways to detect honeypot traps. In this newly emerging battle between honeypots and botnets, it is not hard to imagine that it would not take long for online hackers to come up with new anti-honeypot detection techniques that allow botmasters to systematically disclose the true nature of compromised machines. In fact, some anti-honeypot methods have already appeared on the web (e.g., see www.send-safe.com). Consequently, studying these potential approaches, which is the main focus of this paper, should allow security professionals to develop techniques that better hide honeypots and evade detection by botmasters [5] [6].

Zou and Cunningham [3] suggested an independent methodology to disclose honeypots and remove them from botnets. In this methodology, the botmaster commands bots in the compromised machine to execute some illicit actions (e.g., spamming, or making continuous web requests that look like a DDoS attack.) The target of these actions are machines owned by the botmaster which serve as sensors to detect the correctness of the attack. The work in [3] assumes that honeypots do not respond to these kind of commands by the botmasters. However, if honeypots are designed to completely ignore these commands, then they can be easily detected by the botmasters. On the other hand, full participation by the honeypots in such activities has its associated cost and may lead to legal liabilities. In practice, it sounds more logical to assume that honeypot operators would develop and follow a response strategy that allow them to evade easy detection by botmasters without completely sacrificing their liability.

Motivated by the fact that attackers are becoming more aware of honeypots and that they generally want to avoid being trapped into them, Rowe *et al* [7] suggested equipping normal machines by misleading evidence to make attackers consider them as honeypots. A rational attacker who believes to be interacting with a honeypot would usually withdraw and leave the machine to avoid being captured. It should be noted, however, that the work in [7] does not consider the possibility that attackers can also collect and analyze evidence that support the fact that they are interacting with a normal machine.

In this work, we address the possibility of systematically determining the true nature of a compromised machine by botmasters. In particular, we show that current honeypot architectures and operation limitations allow attackers to systematically collect, combine and analyze evidence about the true nature of the machines they compromise.

In particular, we show how a systematic techniques for evidence combining such as Dempster-Shafer theory can allow botmasters to determine the true nature of compromised machines with a relatively high certainty. In particular, we demonstrate honeypot detection methodology, in which botmasters can detect honeypots by collecting different forms of evidence from different sources and then systematically combine them using the Dempster-Shafer theory of evidence [8]. The main objective of this work is to alert the security community to the following inherent weaknesses in current honeypot design and network defense approaches:

- Attackers can use systematic methodologies to collect evidences from honeypots and combine them in order to calculate a belief value for the true nature of the machines they are interacting with. This enables botmasters to avoid honeypots and remove them from their botnets.

- Using fake honeypots to protect real machines may not work in practice. Attackers can overcome the idea of misleading them, i.e., planting false information into real machines, by systematically collecting evidence that support the fact that they are interacting with real machines.

The rest of the paper is organized as follows. In section II we refer to some of the researches related to honeypot detection and Dempster-Shafer theory in different fields. In section III we briefly review some relevant details of the Dempster-Shafer theory of evidences and show how it can be used to calculate the belief of a system state based of available evidences. In section IV we discuss different types of evidences that can be collected during different phases of interaction between the botmaster and compromised machines. Our model and simulation results that show the use of our model to differentiate honeypots from normal machines are presented in section V. Finally, our conclusion is presented in section VI.

II. Related Work

In this section, we present the research related to honeypot detection and Dempster-Shafer theory.

A. Honeypot Detection

Different techniques have been researched to expose honeypots through exploiting their characteristics to distinguish them from real machines. The work in [9], [10] and [11] focus on the virtual nature of honeypots. Most of honeypots are deployed as virtual machines to reduce costs and facilitate management. However, virtual environment can be detected by probing their distinguishable characteristics such as service availability [9], response time [10], and network links latencies [11]. Virtual machines are deployed on top of an other operating system, i.e. host system, so they trend to show longer response time, comparing to real machines. This enables detecting virtual environments by probing some of their services and comparing the response time to another physical machine. In this work, we consider detecting the existence of virtual environment as an evidence of being a honeypot; regardless of the detection technique. In [3], Zou and Cunningham exploited the unwanted risk of legal liability for honeypot operators when allowing their honeypots to participate in illicit actions. In this methodology, botmasters command the honeypot to participate in harmful actions (e.g. spamming, spreading malware, or performing DDOS attacks). However, the targets of these attacks are not real victims, they are machines controlled by the attacker and used as sensors to measure the tested machine's execution of botmasters' commands. The main idea of this work is that honeypot can not distinguish such test from actual attacks. Security professionals will have to choose between being liable for participating in illegal actions or being disclosed by the botmaster. In this work, we use this methodology to collect evidences about the true nature of the targeted machines. When a machine contributes in executing the botmaster's commands it will have a higher belief of being a normal machine rather than a honeypot. To exploit attackers' abilities of detecting honeypots, Rowe *et al* suggested equipping normal machines with misleading evidences that make attackers consider them as honeypots [7]. An attacker who finds evidences that a particular machine is a honeypot would avoid it to escape being detected. Different types of fake evidences can be planted into normal machines such as increasing the fraction of system files, mimicking the memory traces of the honeypot kernel module (e.g. Sebek), delaying the response to some commands to deceive time measuring tests, and others. It should be noted, however, that this work does not consider the possibility of collecting and analyzing evidences

that support being a normal machine. In our work, attackers can collect evidences and assign them belief values that support being a honeypot or normal machine depending on the evidence parameters.

B. Using Dempster-Shafer Theory

Chen and Venkataramanan [12] used Dempster-Shafer (DS) theory for intrusions detection in ad-hoc network. Their methodology is based on using independent nodes to collect different evidences about the suspected node. To avoid having misleading evidences by dishonest nodes, they built a reputation system in which the belief related to the evidence is calculated based on the node reputation. This way, even when a dishonest node reports a wrong evidence about some other node, the weight of this evidence will have less impact on the final decision. After assigning the belief values for the evidences, authors used Dempster rule of combining evidences to calculate the final belief about the suspected node. They compare their work to other voting techniques, such as majority voting and mean average and found that combining evidences by using Dempster-Shafer is more efficient in the existence of suspicious nodes that are trying to change the voting results. Otrok *et al.* [13] discussed three models for host-based intrusion detection systems. Authors used Bayesian model, Dempster-Shafer model, and a hybrid model that uses both Bayesian and Dempster-Shafer to calculate the belief of a user type (i.e Normal or Malicious) based on the evidences collected from his actions. From above work, we can see that Dempster-Shafer is a promising theory in the area of security for combining evidences and calculating belief values.

III. Honeypot Detection Using DS Theory

In this section, we will describe the Dempster-Shafer theory (DS) [14] in brief, and then introduce our DS-based model for detecting honeypots.

A. Dempster-Shafer Theory of Evidences

In mathematics, the Dempster-Shafer (DS) theory [14] is used to calculate the belief degree, within the range [0,1], for a given hypothesis by combining evidences from different sources. Each evidences must have a belief mass assignments that reflects the support for each possible proposition of the system.

Let $\mathbb{S}$ be the finite set of system propositions, and its power set is $\mathbb{P} = 2^{\mathbb{S}}$. We define a basic belief assignment (BBA) $m : \mathbb{P} \mapsto [0, 1]$ such that $m(\phi) = 0$ and $\sum_{A \in \mathbb{P}} m(A) = 1$.

Each hypothesis $A \in \mathbb{P}$ has two bounds; lower which is called belief (*Bel*) and upper which is called plausibility (*Pl*)

$$Bel(A) = \sum_{B|B \subseteq A} m(B) \tag{1}$$

$$Pl(A) = \sum_{B|B \cap A = \phi} m(B), \tag{2}$$

where the relation between equations (1) and (2) is

$$Pl(A) = 1 - Bel(\overline{A}), \tag{3}$$

where $\overline{A}$ represents the subsets that do not intersect with *A*.

If we have two independent evidences represented by their BBA values m_1 and m_2 from different sources; then we can combine them using Dempster rule of combination:

$$m_{1,2}(A) = \frac{1}{1-K} \sum_{B \cap C = A \neq \phi} m_1(B) m_2(C) \tag{4}$$

where

$$K = \sum_{B \cap C = \phi} m_1(B) m_2(C) \tag{5}$$

B. DS-based Model for Honeypot Detection

Any machine that botmasters intend to compromise could be either a honeypot or normal machine. Thus, the set of possible propositions is $\mathbb{S} = \{Honeypot, Normal\}$ and the power set is $\mathbb{P} = \{\phi, \{Honeypot\}, \{Normal\}, \mathbb{S}\}$. During the interaction with this machine, including the very first actions such as port scanning, botmasters collect predetermined evidences from the set of possible evidences $\mathbb{E} = \{E_1, E_2, ...E_n\}$ and determine their BBA values for each possible hypothesis $h \in \mathbb{P}$. BBA values can be determined following different methodologies based on the evidence type. For instance, some evidences can have binary representation of *exist* or *don't exist*, such as detecting virtual environments, which enables the attacker to set the BBA to one of two values only for each hypothesis. Other evidences can have multiple levels of supporting (e.g. penetrating can be very easy, easy, hard, or very hard) which need more detailed BBA values for each possibility. Upon receiving a new evidence, botmaster uses the equation in (4) to combine these evidences and conclude the degree of belief for this machine. As Dempster rule of combination is associative, the order of evidences is not important and can be combined in any sequence. Although it is possible to calculate the belief of each and every hypothesis, we will only consider calculating it for the proposition 'Honeypot'. After collecting all possible evidences, the final belief of being honeypot is compared with two thresholds, honeypot threshold (h_{th}) and normal machine threshold (n_{th}), where $1 > h_{th} > n_{th} > 0$. The investigated machine is considered as honeypot if the final belief is greater than h_{th}, and a normal machine if less than n_{th}. In other cases (i.e. the final belief is between the two thresholds) the nature of machine is indeterminable and botmaster needs to collect more evidences if wants to disclose its true nature. The following is a simplified pseudo code to execute the DS-based detection algorithm:

Example 1: Consider a case where the botmaster is investigating three machines and has collected three independent evidences from each one (different types of evidences are shown in section IV). The botmaster will follow algorithm 1 trying to determine the true nature of each machine. Botmaster starts by initializing the model parameters ($n = 3, h_{th} = 0.9, n_{th} = 0.1$) and assigning BBA values for each possible hypothesis of the three machines as in table I. Here, we directly assign these values for the sake of simplicity, the weight of different evidences will be discussed in section IV. After that, botmaster uses equation (4) to combine these evidences and calculate the belief of being honeypot for each machine. The final values of the honeypot belief of each machine are: *A=0.93, B=0.06 and C=0.6*. By comparing these values to the assigned thresholds (h_{th} and n_{th}), botmaster can conclude that machine **A** is a honeypot and machine **B** is a normal machine with a very high

Algorithm 1 Calculate Belief of Being Honeypot

Require: $n > 1, h_{th}, n_{th} \in [0, 1]$
  for $i = 1 \rightarrow n$ **do**
    Collect Evidence i
    Set BBA values for evidence(i)
    $i = i + 1$
  end for
  Calculate the belief of being honeypot *Bel(H)*
  if $Bel(H) \geq h_{th}$ **then**
    Machine is Honeypot
  else
    if $Bel(H) \leq n_{th}$ **then**
      Machine is Normal
    else
      Machine is indeterminable
    end if
  end if

TABLE I
BBA VALUES FOR EVIDENCES USED IN EXAMPLE 1

Machine	Evidence 1		Evidence 2		Evidence 3	
	Honeypot	Normal	Honeypot	Normal	Honeypot	Normal
A	0.8	0.2	0.7	0.3	0.6	0.4
B	0.3	0.7	0.1	0.9	0.6	0.4
C	0.6	0.4	0.6	0.4	0.4	0.6

confidence. However, machine **C** true nature can not be determined based on available evidences and botmaster needs to collect more evidences to disclose it.

IV. TYPES OF EVIDENCE

In this section we present and discuss some examples for different types of evidences that can be collected by botmasters in order to evaluate their belief about the true nature of compromised machines. The presented evidences are not the only possible ones that can be used by the botmaster to disclose the true nature of a machine, it is very possible to use other evidences and/or customize the suggested ones [3] [15] [16] [17] to fit with different types of attacks and different levels of botmasters' abilities. We focus on these evidences particularly as they are related to different phases of machine compromising as illustrated next.

A. Difficulty of Compromising

Computers operators usually fortify their machines with different defences, such as anti-virus, firewalls and program patches, which make these machines hard to compromise. On the other hand, honeypots, by design, are supposed to appear like easy-to-compromise machines because they aim to attract a large number of attackers [18]. Some interactive honeypots use sophisticated interactions to enforce attackers reveal more of their tools [19], which gives the attacker a feeling that these machines are not easy targets. Nevertheless, the difficulty of compromising is a good indicator of being a honeypot or a normal machine. On the other hand, difficulty is a relative issue, and even in the extreme scenarios (very easy or very hard to compromise), an attacker cannot be certain about the type of the machine. Thus, attacker can assign a belief function that relates the exploitation

difficulty level to the two possible machine types. Furthermore, attackers can consider different factors in determining the difficulty of exploitation such as:

- Exploited vulnerability: A system that is exploited by using a well-known vulnerability is considered to be an easy-to-compromise. Vulnerability date and the existence of patch for it are important in determining the degree of this factor.
- Number of required trials to penetrate the system: A system that requires more attempts to be penetrated is considered to be harder to compromise and is more likely to be a normal machine rather than a honeypot.
- Number of open ports: A machine with many opened ports is so inviting and is more likely to be a honeypot trap. Honeypot operators would increase the number of opened ports in order to attract more attackers to the trap. Attackers can scan open ports in different ways [15] and they can even use available free software [20] to do so.

The importance of this evidence comes from the fact that some factors can be measured even before the actual attack (e.g. port scanning) and the possibility to have more than one evidence at the same time, such as considering the exploited vulnerability as one evidence and number of needed trials as a second evidence.

B. Virtualization

It is common, and also preferable, for honeypots to run as virtual machines rather than real ones since this allows honeypot administrators to run multiple honeypots on a single physical machine and also makes it easier to control and repair honeypots [21]. Detecting virtual environments [16] can provide botmasters with some evidences about the true nature of the machine since some virtualization features are more likely to be used with honeypots than with normal machines. It should be noted, however, that determining the appropriate type of evidence associated with virtualization is technically involved, especially since many modern applications such as cloud computing [22] have made it common to have normal computer systems running in virtual environments. A rational botmaster will consider the targeted environments when determining the value of this evidence. For example, it is more likely to have productive virtual systems for business users rather than for home users. On the other hand, the absence of virtualization can highly support the hypothesis of being a real machine.

C. Software Diversity

Normal computers typically have different software that are required to perform day-to-day tasks. For example, personal computers usually have MS-Office, OpenOffice or iWork for word processing, spreadsheets and presentations. It is also intuitive to find some Internet browsing software (e.g., Internet Explorer, Chrome or Firefox) and many other useful utilities. Security-related software is also essential for any real computer system, this includes anti-virus, IDS, and anti-malware. For web servers it is common to have software such as Apache, MySQL and MS IIS. The absence, or lack, of these common software strongly supports the hypothesis of being a honeypot. Attackers can thus attempt to collect evidences that support the existence, or absence, of such software and assign them different basic belief values depending on the observed degree of software diversity. The importance of such evidences comes from the easiness of detecting installed software (e.g. listing the contents of '$C:\backslash ProgramFiles$' in Windows systems) which allows for quick determination of the belief degree assigned to this evidence.

D. System Activities

Normal computers are used by legitimate users to perform different tasks. Users may run and terminate programs, create, update and delete files or surf the web. These activities can be detected in different ways such as monitoring CPU utilization, memory, or hard disk usage. One can also observe running processes by using simple line commands such as '*ps*' in Linux systems. A system that does not show such activities is more likely to be a honeypot rather than a normal machine. The existence of human actions, such as keyboard strokes, is an indicator that a given machine is being used by legitimate users. Although high interaction honeypots can simulate many of these activities [19], still the absence, or lack, of such activities can be considered as a strong evidence that supports the likelihood of being a honeypot. Botmasters can use many tools to detect system activities and determine the possibility that these activities support the hypothesis of being interacting with a normal or honeypot machine. AS normal systems may have some idle time, e.g. off hours in companies or user absence at night time, it is important for the botmaster to measure the targeted system activities during suitable time intervals or by taking the average of these activities over a relatively long period to avoid the effect of idle weekends and night time. Despite the many factors that determines the use of this type of evidences, botmasters can find it very important to easily support machine type as normal when observing a lot of system activities.

E. Outgoing Traffic

Typically, honeypots do not allow malicious outgoing traffic due to the fear of legal liabilities associated with participation in real attacks [3]. As suggested by Zou and Cunningham [3], botmasters may abuse this fear of legal liability to develop simple test techniques that measure the machine ability to send outgoing traffic. To determine the true nature of a compromised machine, the botmaster may command it to perform what looks like an illicit activity, e.g. spreading malware to other machines, participate in a DDoS attack or send spam emails. However, the targeted victims of these actions are controlled by the botmaster and used as sensors to measure the contribution of the tested machine. Based on the observed response, attackers can convert these measurements into a belief function value that reflects the likelihood of being a normal machine or a honeypot. A key point in here is that honeypots cannot distinguish between real attacks and tests. On the other hand, the lack of response does not necessarily indicate a honeypot. A normal machine may sometimes become unable to respond to such test commands because of a firewall setting, connectivity problems or simply because it is switched off. So attacker may need to repeat such tests and consider each response as a separate evidence that supports being a normal machine or a honeypot with different belief values. Attackers can also collect multiple evidence by performing multi-step test and measure the contribution of the machine in each step separately. For example, to invade a new victims, the bot performs multiple steps such as collecting email address then sending phishing emails with links to websites that contain the malicious code, as in Storm [23] and Waledac [24] bots. Other bots such as Agobot [25] perform port scan to identify possible victims, and then they perform different types of scans that include vulnerability scan, back door scan and brute force password scan to detect possible entries to the victim. Finally, they send their malicious code through these discovered vulnerabilities. Botmasters can use these steps to delude honeypots and measure their ability to perform different types of illicit actions. The result of each step will be assigned with a belief

value of the machine nature. For example, if the botmaster requested the machine to send 100 packets to one target and the sensor reported receiving only 10 of them, this results in higher belief value of being honeypot than the belief value of being a normal machine. On the other hand, if the sensor reported receiving 90 packets out of 100, the belief value of being normal machine will be higher than the one corresponding to being a honeypot. From the attacker perspective, evidences associated with the "Outgoing Traffic" have two advantages. First, they are completely controlled by the botmasters who can decide when and how to make the machine generate these evidences and repeat them as much as needed. Second, the "outgoing traffic" concept is of special importance as it can indicate the belief of machine type when the bot is unable to send the evidences collected locally in that machine. For instance, suppose the bot has collected some evidences regarding the system activities in a compromised honeypot and was not able to send them back to the botmaster due to the limitation of sending outgoing traffic in the honeypot. In this case, the absence of the outgoing traffic can be considered as an evidence by itself [3]. On the other hand, the other set of evidence makes it easier to disclose low interaction honeypots and have other advantages such as:

- The "difficulty of compromising" evidence can be determined without the need of any extra effort, it is somehow a *free* evidence.
- Unlike the evidence of Outgoing Traffic, these evidences do not require the botmaster to issue any special commands through the botnet which reduces the possibility of being detected.
- These evidences make it easier to disclose low interaction honeypots.

Finally, it should be noted that we are not limiting available evidences to the above five types. These types are presented as *examples* and it is not hard to imagine other types of evidence that can be collected and analysed during different phases of online attacks [17].

V. Simulation and Findings

Throughout this sections, we assume that attackers are able to collect different evidences during and after the machine compromising process. Attackers follow the steps described in section III-B to update their belief about the nature of the machine they are interacting with. As the Dempster-Shafer rule of combination is associative, they can combine evidences one by one and generate accumulative values until they reach the required level of confidence (i.e. predetermined thresholds) about the nature of the machines under test. Table II lists the suggested different machines specifications used in generating the BBA values for evidences. Throughout the simulation, we refer to a high interaction honeypot as 'H.I.H' and to a low interaction honeypot as 'L.I.H'. We assume that the H.I.H is well prepared to mimic the characteristics of a real machine which makes the attacker assigns BBA values similar to what an actual normal machine would have. However, as determined by [3] the outgoing-traffic test will give results that can distinguish between real machines and honeypot. In what follows, we present our simulation results for different scenarios that show how our suggested DS model can be used by attackers to determine the true nature of compromised machines.

TABLE II
SPECIFICATIONS OF MACHINES

Machine	Compromising	Virtualizing	Software Diversity	System Activities	Outgoing Traffic
Normal	Difficult	No	High	High	Yes
H.I.H	Difficult	No	High	High	No
L.I.H	Easy	Yes	Low	Low	No

TABLE III
RANGES OF BBA FOR DIFFERENT EVIDENCES USED IN SIMULATION 1

Machine	E1-E4		E5	
	Honeypot	Normal	Honeypot	Normal
Normal	0-0.1	0.6-0.9	0-0.1	0.6-0.9
H.I.H	0.1-0.3	0.3-0.7	0.6-0.9	0-0.1
L.I.H	0.6-0.9	0-0.1	0.6-0.9	0-0.1

A. Distinguishing Normal Machines from Honeypots

In this scenario, we assume that an attacker tries to determine the true nature of three different types of machines: a normal machine, a low interaction honeypot, and a high interaction honeypot. Table III shows the assumed ranges of basic belief values associated with different evidences collected by the attacker. In the table, $E1, E2, E3, E4$ and $E5$ represent evidence corresponding to difficulty of compromising, virtualization, software diversity, system activities, and contribution in the botmaster's tests (i.e. outgoing traffic) respectively. As shown in the table, we assume that high interaction honeypots are prepared to simulate a real machine by providing evidences ($E1 - E4$) in the same way as real machines. As mentioned above, $E5$ is of a special importance to the attackers who rely on the fact that honeypot operators have legal liability constraints that may prevent them from fully participating in real attacks [3]. As shown in the table, we assume that honeypots have low contribution percentage that support being honeypot more than supporting being a normal machine, while the high percentage of contribution for the normal machine provides evidences that support being normal more than it does for being a honeypot. Fig. 1 shows how the attackers' belief about the nature of the compromised machine varies as they collect more evidence. The simulation results were obtained by generating 1000 simulation scenarios where the value of evidences assume the ranges in Table III with a uniformly distributed probability. As depicted in the figure, attackers are able to detect low interaction honeypots after collecting only few evidences. The need of collecting more evidence becomes apparent when interacting with high interaction honeypots. As shown in the figure, the first collected set of evidences support being a normal machine decreasing the belief value. However, after collecting the fifth evidence, the belief of being honeypot starts to increase due to the fact that these evidences are collected by measuring the machine participation in attacker's tests, which eventually leads to determining the true nature of these machines as honeypots. From our simulation results, we conclude that if the first few evidences support being honeypot with high values, then rational attackers would not continue collecting more evidence and is more likely to stop interacting with these machines. However, if the first few evidences support being a normal machine, then rational attackers are more likely to try collecting more evidence (especially of type $E5$) in order to be sure that they are

TABLE IV
RANGES OF BBA FOR DIFFERENT EVIDENCES USED IN SIMULATION 2

Machine	Hypothesis	E1	E2	E3	E4	E5
Type 1	Honeypot	0.3-0.7	0-0.1	0-0.1	0-0.1	0-0.1
	Normal	0.1-0.3	0.6-0.9	0.6-0.9	0.6-0.9	0.6-0.9
Type 2	Honeypot	0.3-0.7	0.3-0.7	0-0.1	0-0.1	0-0.1
	Normal	0.1-0.3	0.1-0.3	0.6-0.9	0.6-0.9	0.6-0.9
Type 3	Honeypot	0.3-0.7	0.3-0.7	0.3-0.7	0-0.1	0-0.1
	Normal	0.1-0.3	0.1-0.3	0.1-0.3	0.6-0.9	0.6-0.9

not being deceived by high interaction honeypots.

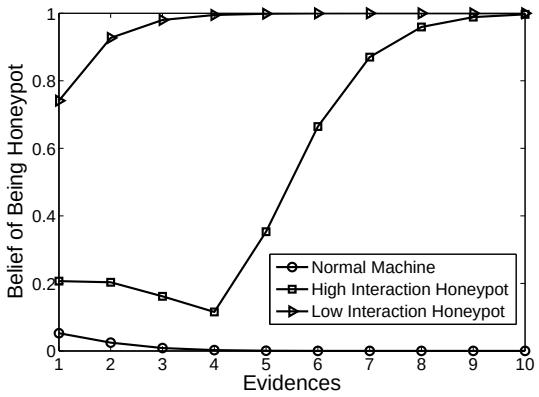

Fig. 1. Belief Evaluation Of Being Honeypot For Normal Machine Compared With High Interaction and Low Interaction Honeypots

DS rule for combining evidences is associative and consequently the order of calculated evidences does not affect the final belief value. What really affects is how much evidences support one hypothesis over the other and the BBA values for these evidences. Fig. 2 shows our simulation results for a scenario that shows how attackers can use the DS model to determine the true nature of three normal machines despite having different BBA values for some evidences. Table IV shows the values used in our simulation that represent the possibility of providing each evidence. In this example, we assume that the machines of type '1' are easy to compromise, machines of type '2' are easy to compromise and are deployed as virtual machines, and machines of type '3' are easy to compromise, deployed as virtual machines, and have no software diversity . All machines have a high contribution percentage which support being normal more than being honeypot. As depicted in the figure, by repeating the test multiple times, attackers are able to determine the assumed true nature of the three machines (i.e. normal machines).

Figures 1 and 2 show to what extent attackers can use a single flaw in honeypot design to reveal, and consequently avoid, honeypots. Even with the best equipped honeypot, as in first simulation, repeating the outgoing traffic test would eventually lead the attacker to uncover it. The legal liability of participating in executing botmasters' commands must have a special attention

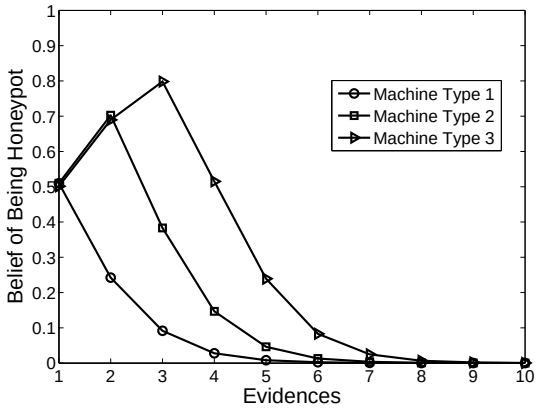

Fig. 2. The Effect Of Different Supporting Evidences Values In Detecting Three Normal Machines

from the security community to find appropriate solution to avoid, or reduce, attackers ability of exploiting this particular flaw.

B. Using Multi-step Test

It is possible for the attacker to consider several steps as a single test when interacting with a machine by using the outgoing traffic methodology, i.e. by measuring the machine contribution of executing a fake attack test. However, if a normal machine executes only parts of the test but is unable to complete it due to connectivity problems or firewall settings, then it is considered as 'not-contributing', which may make attackers consider it as a honeypot. On the other hand, security professionals can design a high interaction honeypot such that it may contribute in the less severe steps of the attacks, e.g. scanning, without actually doing harmful actions such as infection. To determine the actual contribution of a machine, botmasters can employ DS to combine different evidences that represent the execution of different steps and allocate different basic belief assignment for each step based on its importance and potential legal liability. For example, consider the infection process of Asprox botnet which includes multiple steps [26]. The infection process can be divided into the following three steps:

- Searching for SQL Servers: The bot in the infected machines searches, using Google, for websites that are hosted on Microsoft SQL Server. Botmasters can customize this step by redirecting the search request to one of their sensors that act as a search engine and measures the execution of the search command.
- SQL Injection: The bot receives a reply from the search engine (sensor) that contains a list of potential victims (other sensors). The bot uses SQL attack tool to attack these sensors, which in turn will measure the contribution of the tested machine.
- Spamming: The bot sends emails to a list of users to lure them to visit the infected website. The botmaster can change the email addresses such that these emails will be received by the sensors to determine the contribution of the machine.

In this part of our simulations, we compare the contribution of three machine types:

1) Normal machines that highly participate in all steps.
2) Low interaction honeypots that poorly participate all steps.
3) High interaction honeypots that try to avoid detection and also reduce their potential liability by highly participating in

TABLE V
THE VALUES OF BBA FOR EACH STEP IN MULTI-STEPS TEST

Machine	Hypothesis	Step 1	Step 2	Step 3
Normal	Honeypot	0.2-0.3	0.0-0.1	0.2-0.3
	Normal	0.4-05	0.8-0.9	0.4-05
H.I.H	Honeypot	0.2-0.3	0.6-0.9	0.2-0.3
	Normal	0.4-05	0-0.1	0.4-05
L.I.H	Honeypot	0.6-0.9	0.6-0.9	0.6-0.9
	Normal	0-0.1	0-0.1	0-0.1

the first and last steps of each test (search and spam email)and poorly participate in the second step (SQL injection). The rationale behind these contribution decisions can be justified by the fact that the operators of these honeypots may consider the first step as a legal action (searching). Also, while the second step contains a direct attack, the last step can be argued as less jeopardizing, because in most cases the sent email will direct the victim to a safe website due to skipping the SQL injection action.

Table V shows the assumed basic belief assignment associated with evidences collected during these different steps. Botmasters can assign higher BBA values for step two as it represents an illegal action that honeypot operators prefer to avoid due to its legal liability. Fig. 3 shows how the botmaster is eventually able to determine the true nature of these three machine types. In this simulation, if a botmaster considered applying 5 tests (that is 15 sub-steps) and setting belief threshold of 0.9 for a machine to be considered as a honeypot and 0.1 for normal machine, it will be easy to recognize the different types of machines.

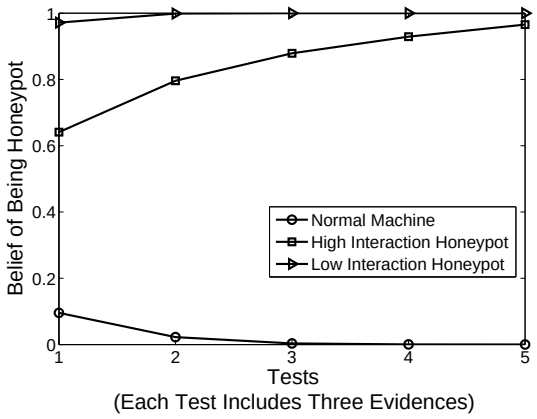

Fig. 3. Using multi-step test to determine the true nature of three machine types: normal, low interaction honeypots and high interaction honeypots

The simulation in figure 3 shows the importance of assigning the BBA following a reasonable approach. All different types of evidences must be evaluated based on several factors depending on its nature, legal liability, the scope of targeted machines and even the attacker himself. For example, an expert attacker would consider less importance for 'Difficulty of Compromising' evidence than what a novice attacker would do. An expert attacker will find more machines as easy targets, and thus the BBA for this evidence should not be very distinctive for him. Another example is when considering the 'System Activities' evidence,

it is important to take into consideration the time of test (day/night and weekdays/weekends) and the type of targeted users (home users or business). Some of these considerations might be easy to be automatically taken by the bot itself, e.g. checking system time, while others may need more manual customization by the attacker.

VI. Discussion and Conclusions

Honeypots are important tools for security professionals in their race against online attackers. If hackers are able to detect honeypots, then attacking honeypots may also provide hackers with valuable information about their observers, the same way that the honeypots were designed to provide the security professionals with valuable information about botmasters. In this work, we argued that botmasters can systematically collect, combine and analyze evidence throughout and after the machine compromise process in order to determine the true nature of compromised machines with a relatively high certainty. Our simulation results show that this technique can be powerful enough to determine the true nature of most compromised machines and honeypots. We hope that this work would shed some light on weak aspects of current honeypot designs which can be exploited by botmasters to generate and gather evidences that lead to disclosing honeypots and removing them from botnets. Honeypots developers can use this work as a guidance to better enhance the hiding capabilities of their honeypots by making more informed decisions before deciding their response to botmasters commands, e.g., by simulating the current state of the attacker belief and basing their decisions on this estimated belief values.

References

[1] P. Ramneek. Bots and botnet - an overview. SANS Institute InfoSec Reading Room, August 2003.

[2] M. Akiyama, T. Kawamoto, M. Shimamura, T. Yokoyama, Y.Kadobayashi, and S. Yamaguchi. A proposal of metrics for botnet detection based on its cooperative behavior. In *Proc. 2007 International Symposium on Applications and the Internet Workshops*, pages 82–85, 2007.

[3] Cliff Changchun Zou and Ryan Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *DSN*, pages 199–208, 2006.

[4] Z. Li, Q. Liao, and A. Striegel. Botnet economics: Uncertainty matters. In *Proc. The 7th Workshop on the Economics of Information Security, WEIS*, 2008.

[5] J. Bethencourt, J. Franklin, and M. Vernon. Mapping internet sensors with probe response attacks. In *Proc. USENIX Security Symposium*, pages 193–208, 2005.

[6] T. Holz L. Oudot. Defeating honeypots: Network issues (part 1 and 2). *SecurityFocus InFocus Article*, 2004.

[7] N.C. Rowe, B.T. Duong, and E.J. Custy. Fake honeypots: A defensive tactic for cyberspace. In *Information Assurance Workshop, 2006 IEEE*, pages 223 –230, June 2006.

[8] Bin Zhang and S.N. Srihari. Class-wise multi-classifier combination based on dempster-shafer theory. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 2, pages 698 – 703 vol.2, Dec. 2002.

[9] S. Mukkamala, K. Yendrapalli, R. Basnet, M.K. Shankarapani, and A.H. Sung. Detection of virtual environments and low interaction honeypots. In *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, pages 92 –98, June 2007.

[10] P. Defibaugh-Chavez, R. Veeraghattam, M. Kannappa, S. Mukkamala, and A.H. Sung. Network based detection of virtual environments and low interaction honeypots. In *Information Assurance Workshop, 2006 IEEE*, pages 283 –289, June 2006.

[11] Xinwen Fu, Wei Yu, Dan Cheng, Xuejun Tan, K. Streff, and S. Graham. On recognizing virtual honeypots and countermeasures. In *Dependable, Autonomic and Secure Computing, 2nd IEEE International Symposium on*, pages 211 –218, 29 2006-oct. 1 2006.

[12] T.M. Chen and V. Venkataramanan. Dempster-shafer theory for intrusion detection in ad hoc networks. *Internet Computing, IEEE*, 9(6):35 – 41, Nov.-Dec. 2005.

[13] Hadi Otrok, Benwen Zhu, Hamdi Yahyaoui, and Prabir Bhattacharya. An intrusion detection game theoretical model. *Information Security Journal: A Global Perspective*, 18(5):199–212, 2009.

[14] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[15] Wikipedia. Port scanner — wikipedia, the free encyclopedia, 2012. [Online; accessed 31-January-2012].

[16] Peter Ferrie. Attacks on more virtual machine emulators. Symantec Advanced Threat Research, 2006. [Online; accessed 30-January-2012].

[17] T. Holz and F. Raynal. Detecting honeypots and other suspicious environments. In *Information Assurance Workshop, 2005. IAW '05. Proceedings from the Sixth Annual IEEE SMC*, pages 29 – 36, June 2005.

[18] I. Mokube and M. Adams. Honeypots: Concepts, approaches, and challenges. In *Proc. The 45th Annual Southeast Regional Conference, SIGAPP: ACM Special Interest Group on Applied Computing. ACM*, pages 321–326, 2007.

[19] Gérard Wagener, Radu State, Alexandre Dulaunoy, and Thomas Engel. Self adaptive high interaction honeypots driven by game theory. In *SSS*, pages 741–755, 2009.

[20] Radmin. Advanced port scanner 1.3 (free). http://www.radmin.com/products/previousversions/portscanner.php, 2006. [Online; accessed 30-January-2012].

[21] Niels Provos. A virtual honeypot framework. In *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 1–1, Berkeley, CA, USA, 2004. USENIX Association.

[22] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A berkeley view of cloud computing. In *Technical Report No. UCB/EECS-2009-28*. University of California at Berkley, USA, Feb. 10, 2009.

[23] Wikipedia. Storm botnet — wikipedia, the free encyclopedia, 2011. [Online; accessed 1-February-2012].

[24] Gilou Tenebro. W32.waledac, threat analysis. Symantec Research, 2010. [Online; accessed 01-February-2012].

[25] Paul Barford and Vinod Yegneswaran. An inside look at botnets. In Mihai Christodorescu, Somesh Jha, Douglas Maughan, Dawn Song, and Cliff Wang, editors, *Malware Detection*, volume 27 of *Advances in Information Security*, pages 171–191. Springer US, 2007. 10.1007/978-0-387-44599-1_8.

[26] R. Borgaonkar. An analysis of the asprox botnet. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 148 –153, July 2010.