A Game Theoretic Investigation for High Interaction Honeypots

Osama Hayatle[1], Hadi Otrok[2], Amr Youssef[1]
[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 traps designed to resemble easy-to-compromise computer systems in order to deceive botmasters. Such security traps help security professionals to collect valuable information about botmasters' techniques and true identities. Depending on the complexity of services provided by honeypots, botmasters might be able to detect these traps by performing a series of tests. In particular, to detect honeypots, botmasters can command compromised machines to perform specific actions such as targeting sensor machines controlled by them. If honeypots were designed to completely ignore these commands, then they can easily be detected by the botmasters. On the other hand, full participation by honeypots in such activities has its associated costs and may lead to legal liabilities. This raises the need for finding the optimal response strategy needed by the honeypot in order to prolong its stay within the botnet without sacrificing liability. In this paper, we address the problem of honeypot detection by botmasters. In particular, we present a Bayesian game theoretic framework that models the interaction between honeypots and botmasters as a non-zero-sum noncooperative game with uncertainty. The game solution illustrates the optimal response available for both players. Simulation results are conducted to show the botmasters' behavior update and possible interactions between the game players. The obtained results can be utilized by security professionals to determine their best response to these kind of probes by botmasters.**

***Index Terms*—Honeypots, Anti-Honeypot Technology, Botnets and Game Theory**

I. Introduction

Nowadays, botmasters are motivated by financial gain [1] rather than fame and media attention. In particular, they aim to control a large number of computers, known as botnets [2], in order to use them in Internet-based criminal activities via SPAM, phishing, identity theft, and Distributed Denial of Service (DDoS) extortion. This switch in motivation has moved the combat between botmasters and security professionals, over the past years, from viruses and worms spreading/detection to battles with bots [3] where it is even possible to rent a botnet, or part of it, to conduct some illicit criminal activities [4].

Honeypots are traps designed to resemble easy-to-compromise systems in order to tempt botmasters to invade them. This will allow the collection of valuable information about botmasters' techniques, tools and even their true identities. Honeypots can be classified [5] according to the complexity of their services as (i) minimal servers which provide an open service port, (ii) restricted servers which provide basic interactions, (iii) simulated servers which provide complex interactions and (iv) full servers which provide full functional support.

If hackers are able to detect honeypots, then attacking honeypots may also provide hackers with valuable information about their observers [6], the same way that the honeypots were designed to provide security professionals with valuable information about botmasters. Thus, hackers are actively working on finding ways to detect honeypot traps. In fact, some anti-honeypot methods have already appeared on the web (e.g., see www.send-safe.com).

It should also be noted that it is not only hackers that are investigating potential weaknesses of honeynets. Some security professionals have recently looked at different weak sides of honeypots and investigated countermeasures against them (e.g., [7], [8]). These studies are aimed to warn the security community of potential pitfalls of current honeypot technologies, draw the attention of honeypot developers to possible limitations and deficiencies and help develop new honeypot systems with anti-detection techniques. Bethencourt *et al.* [7] proposed an attack technique to discover the location of Internet sensors, including honeypots, by probing their publicly published reports. Krawetz [5] introduced a commercial software, *Send-Safe*, that allows spammers to avoid honeypots by using the *Honeypot Hunter* tool which opens a mail server on the honeypot and connects back to itself. If the connection is established and the tool could not send itself the request, then this machine, which prevents outbound connections, is considered as a honeypot. Li *et al.* [9] argue that having uncertain number of honeypots decreases the botmaster revenue. If the botmster is uncertain about the percentage of honeypots in the botnet, then extra bots have to be used in order to compensate the lack of attack power caused by this uncertainty. The inclusion of these extra bots adds extra cost which reduces the botmasters' revenue. Eventually this may lead to break down this type of Internet-based criminal business. On the other hand, Zou and Cunningham [3] suggested a software and hardware independent methodology to disclose honeypots and remove them from botnets. In this methodology, the botmaster commands the bot in the compromised machine to execute some illicit actions (e.g., spamming, or making continuous

978-1-4577-2053-6/12/$31.00 ©2012 IEEE

web requests that look like a DDoS attack). The target of these actions is a machine (or more) owned by the botmaster which serves as a sensor to detect the correctness of the attack. The work in [3] assumes that honeypots do not respond to these kinds 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 [6]. This raises the need for considering the following question: *What is the best response strategy that the honeypot can follow in order to evade detection by botmasters without completely sacrificing the liability?*

Furthermore, the execution of such tests by botmasters is bounded to different types of constraints such as firewall settings that might prevent normal bots from always cooperating [10]. Such constraints can lead botmasters to misjudge normal bots as honeypots and drop them from the botnet. To avoid such a false negative decision, botmasters need to run their tests several times in order to build a belief evaluation of the machine nature and then, only when the evaluation belief reaches a specific threshold, then botmasters should remove the machine from the botnet. Thus, the question that raises in this context is: *How to differentiate between normal machines and honeypots with a small number of test commands?*

In this paper, we develop a Bayesian game theoretic framework that models the interactions between honeypots and botmasters in the above setup. In particular, we formulate the interaction between botmasters and the honeypots as a non-cooperative, non-zero-sum game with incomplete information [11] about the honeypot. In this game, botmasters aim to determine their opponents' true nature using the posterior belief function. Botmasters test their opponents by commanding them to send malicious traffic to one or more target machines controlled by the botmasters and which work as sensors. To make it harder for the honeypots, botmasters mix test commands with real attack commands and repeat the test multiple times. Up to the authors' knowledge, our game theoretic model is among the first efforts that formulate the interaction between botmaster and honeypot from the botmasters' standpoint.
In summary, our contribution in this paper is a game theoretic model that:

- Guides security professionals for the best contribution strategy that considers the tradeoff between identity disclosure and liability.
- Formally analyzes the moves of botmasters against honeypots.
- Investigates the botmasters' best strategy to differentiate honeypots from normal machines.

The rest of this paper is organized as follows. Section II presents the game model and explains the players' strategies and payoffs. The formulated game is discussed in Section III and the simulation results are presented in Section IV. Finally, our conclusion is given in Section V.

II. The Game Model

In the this game, we have two players: the botmaster and the honeypot. Throughout the game, the botmaster sends attack commands to the honeypot who responds by participating in the requested attack with a specific contribution percentage (ranging from 0 to 1). Botmasters have incomplete information about the nature of their opponents. In other words, botmasters do not know a-priori the true nature of the machine they are interacting with, i.e., whether it is a normal machine or a honeypot. To disclose the opponent's identity, botmasters command the compromised machines to perform specific actions that can be measured by sensor machines controlled by them.

Depending on the nature of the test attack, the botmasters' sensors can use different methods to determine the correctness of the attack as follows:

- Infection Test: In this method, the botmaster commands the bot to infect other machines, where some of these machines act as the attack sensors. When the sensors receive the attack vector, they compare its digest (e.g. its MD5 hash) with a previously saved value. If both values are equal, the machine passes the test.
- DDoS Test: The botmaster commands the compromised machine to continuously send web requests to a server that is controlled by the botmaster. From the bot/honeypot perspective, this looks like a DDoS attack. The web server verifies receiving these requests and if the compromised machine sends enough requests, it is considered as a normal bot.
- Spamming Test: The botmaster commands the compromised machine to send emails to group of email addresses which are owned by the botmaster. A machine that sends enough number of such spam emails is considered as a normal bot.

A. The Players Strategies

The botmaster has three strategies:

1) Test: When choosing this strategy, the botmaster commands the bot to attack the sensor machine using one of the test attacks discussed above. A normal machine would respond to this test by attacking the target machine (the sensor), while a honeypot will have some constraints on fully responding to such commands. To prevent a single point of failure, the botmaster may use multiple sensors which may also increase the accuracy of the test.
2) Attack: In this case, the botmaster commands the bot to attack a real target, for example, by launching a DDoS attack or infecting other machines to recruit more bots.
3) Not Interacting: In this case, the botmaster chooses not to perform any activity with the compromised bot.

The honeypot has two strategies:

1) Contribute: In this case, the honeypot executes the attack commands received from the botmaster. While this helps the honeypot to collect more information about the

botmaster and stay for a longer time as a member of the botnet, participating in such illicit activities has its own cost in addition to possible legal liability for damages caused by this attack.

2) Not-Contribute: In this case, the honeypot chooses not to execute the attack commands.

B. Pay-off

In this subsection, we consider the different factors affecting the players' pay-off.

- Operating cost of the Honeypot/Bot: Even without any interaction between botmasters and honeypots, both have some cost associated with maintaining and running their software and machines. Throughout the rest of the paper, we use O_h and O_b to denote the operating cost of of the honeypot and the operating cost of the bot, paid by the honeypot operator and by the botmaster, respectively.
- Cost of Attack/Test: When the botmaster interacts with the honeypot, the latter reveals extra information about its true identity, tools and techniques. The revealed information differs depending on the botmaster's strategy, i.e., attack or test. Botmasters need to keep their actual next target as well as their newly developed hacking techniques hidden form security professionals, especially when preparing botnets for a future attack [12]. We use C_a and C_t to denote the cost of performing attacks and tests, respectively.
- Revenue/Loss by Honeypot Contribution: Each bot contribution in a real attack adds more value to it. In case of spamming, each bot has a specific capacity for sending spam emails. The more spam emails sent, the higher the revenue attained by the botmaster. In DDoS attacks, it is important to send enough traffic to the victim server in order to consume its computation and/or bandwidth resources. Typically, a larger number of bots participating in a DDoS attack increases its probability of success. Thus, when the honeypot contributes in a real attack, it adds more value to the botmaster's revenue. If the honeypot does not contribute, this will negatively affect the revenue of the botmaster who initially counts on this participation for the success of the attack. To reflet this payoff, we use R to denote the revenue added to the botmaster's pay-off when the honeypot contributes to real attacks.
- Revealing Honeypots: When a honeypot does not participate in the botmaster's tests, the botmaster has a greater chance of disclosing its true nature and consequently quitting interaction with it. This is considered as an extra reward to botmasters since it improves their business and provides better protection for them. We use E to denote this kind of benefit which is added to the botmaster when the honeypot does not participate in a test attack.
- Liability: When a honeypot participates in a real attack, it acquires the risk of being liable for this contribution. While in some situations taking this risk is prohibitively high, in other scenarios this may not necessarily be the case, for example, if the honeypot is administered in collaboration with law enforcement agencies or if some fees have to be paid to acquire and administer some legal permission in order to be allowed to respond in a very controlled way to such actions. To model the cost associated with this liability constraints, we assign a penalty L for honeypots participating in a real attack.

TABLE I
SUMMARY OF PLAYERS' PAY-OFF

C_a	Cost acquired by the botmaster when performing a real attack
C_t	Cost acquired by the botmaster when performing a test
R	Revenue gained by the botmaster when a machine participates in a real attack
E	Reward to the botmaster when a honeypot does not participate in a test
L	Honeypot liability/penalty for participating in an attack
O_h	The cost of maintaining and running the honeypot
O_b	The cost of maintaining and running the bot

Table I summarizes the payoff factors discussed above. Having the strategy 'Not-Interacting' means that the botmaster has compromised the machine but does not do anything with it. If this machine is a normal machine, then choosing this strategy means that the botmaster spent effort and time for nothing. Furthermore, if this machine is a honeypot, then the botmaster has revealed its identity and the bot code without any further objective. Thus it is intuitive to conclude that the botmaster will not choose such a strategy, and consequently the pay-off table would be reduced to Table II. Furthermore, we assume that:

- $C_a > C_t$: A real attack contains more important information than a test does. Although some information is common in both types of actions (e.g. source of the command or type of attack), a real attack may reveal the actual target, instead of a fake one, the used code and the vulnerabilities being exploited.
- $L > C_a$: If the liability is less valuable than information obtained from the botmaster, it is clear that a honeypot would always contribute in executing botmasters' commands to stay in their botnets and get more information. However, as argued in [3], this is not usually the case.
- $R > C_a$: Botmasters are financially motivated [1]. If the revenue obtained from a bot does not exceed the risk of revealing the attackers' information when interacting with it, then it would not be profitable for the botmaster to recruit this bot.

Under the above reasonable assumptions, it is straightforward to show that there is no pure strategy Nash equilibrium for this game. In what follows, we use a mixed strategy to solve this game. Let p denote the probability that the botmaster chooses to launch a test and q denote the probability of the honeypot chooses to participate in executing the botmaster's commands.

C. Utility Function

The utility function of botmasters is affected by the uncertainty of their opponents. Botmasters use the posterior belief

TABLE II
PAY-OFF TABLE WHEN THE BOTMASTER INTERACTS WITH A HONEYPOT

Strategy	Contribute	Not-Contribute
Test	$-C_t - O_b, C_t - O_h$	$E-C_t-O_b, C_t-E-O_h$
Attack	$R-C_a-O_b, C_a-L-O_h$	$-C_a - O_b, C_a - O_h$

TABLE III
PAY-OFF TABLE WHEN THE BOTMASTER INTERACTS WITH A REAL BOT

Strategy	Contribute	Not-Contribute
Attack	-O_b+R, 0	-O_b,0

function, $\mu_t(\theta_h)$, in calculating the pay-off associated with their strategies where $\mu_t(\theta_h)$ reflects the botmasters' belief about the type of the machine that they interact with, i.e., whether it is a honeypot or a normal machine. When $\mu_t(\theta_h)$ approaches 1, the botmaster becomes certain that the machine is a honeypot. At the beginning of the evaluation process, botmasters set $\mu_0(\theta_h)$ to reflect their a-priori belief about the nature of the machine. In other words, $\mu_0(\theta_h)$ is set to the value that reflects the botmaster's belief about the percentage of honeypots in the botnet. A high value (close to 1) for $\mu_0(\theta_h)$ can be used by a suspicious botmaster who may have some reasons to believe that the botnet is penetrated by a large percentage of honeypots. Conversely, setting $\mu_0(\theta_h)$ close to 0 implies that the botmaster is almost certain that the botnet is not infiltrated by any honeypots. Then the botmaster updates its belief as follows:

$$\mu_{t+1}(\theta|a) = \frac{\mu_t(\theta_h) \times P(a|\theta_h)}{\mu_t(\theta_h)P(a|\theta_h) + (1-\mu_t(\theta_h))P(a|\theta_n)} \quad (1)$$

where $\mu_{t+1}(\theta_h|a)$ denotes the belief of having the opponent type as 'Honeypot' at time $(t+1)$ after observing the strategy a, $\mu_t(\theta_h)$ denotes the value of the belief function at time t, and $P(a|\theta_h)$, $P(a|\theta_n)$ denotes the probability of observing strategy a from a honeypot and from a normal machine, respectively. Let $P_t(a = Contribute|\theta_n) = U$. One can find a good estimate for U by evaluating the expected percentage of working time for a normal machine (e.g., we may assume that $U = \frac{h}{24} \times \frac{d}{7}$, where h denotes the average number of working hours in a day and d denotes the average working days in a week.) U can also be affected by other factors such as firewall settings, temporarily failures of normal machines and geographical and time zones distribution of the botnet.

After updating the belief function, the botmaster calculates the utility function as:

$$\begin{aligned} U_a &= \mu(\theta_h)[-pq(C_t + O_b) + p(1-q)(E - C_t - O_b) \\ &\quad +(1-p)q(R - C_a - O_b) + (1-p)(1-q) \\ &\quad (-C_a - O_b)] + (1 - \mu(\theta_h)[U(R - O_b) \\ &\quad +(1-U)(-O_b)] \\ &= -\mu(\theta_h)UR + UR - \mu(\theta_h)C_a + \mu(\theta_h)pC_a - O_b \\ &\quad +\mu pE - \mu(\theta_h)pqE - \mu(\theta_h)pqR - \mu(\theta_h)pC_t \\ &\quad +\mu(\theta_h)qR \end{aligned}$$

Let q^* denote the optimum value of q, i.e., the optimum participation percentage by the honeypot to attack commands received from the botmaster, i.e., we have

$$U_a(p^*, q^*) \geq U_a(p, q^*).$$

We can determine q^* by calculating the first derivative of U_a with respect to p and equating it to zero. Thus we have

$$\mu C_a + \mu E - \mu q^* E - \mu q^* R - \mu C_t = 0 \Rightarrow$$
$$q^* = \frac{E + C_a - C_t}{E + R} \quad (2)$$

Similarly, the utility of the honeypot is given by

$$\begin{aligned} U_h &= pq(C_t - O_h) + p(1-q)(C_t - (E + O_h)) \\ &\quad +(1-p)q(C_a - (L + O_h))+ \\ &\quad (1-p)(1-q)(C_a - O_h) \\ &= C_a - pC_a - pE + pqE + pqL - qL - O_h + pC_t \end{aligned}$$

Let p^* denote the optimum value of p, i.e.,

$$U_h(p^*, q^*) \geq U_h(p^*, q).$$

Similar to the above analysis, we determine p^* by calculating the first derivative of U_h with respect to q and setting it to zero

$$p^*E + pL - L = 0 \Rightarrow p^* = \frac{L}{E + L} \quad (3)$$

Thus to maximize their utility, botmasters send attack commands with probability

$$\begin{aligned} P_a &= \mu(t) \times (1 - p^*) + (1 - \mu(t)) \times 1 \\ &= 1 - \mu(t) \times p^* \end{aligned} \quad (4)$$

and send test commands with probability

$$P_t = \mu(t) \times p^* \quad (5)$$

III. GAME DISCUSSION

From the game model above, it is clear that botmasters will eventually be able to disclose the true nature of their opponent by observing the opponent's responses and consequently updating their belief function about it. Botmasters are also able to optimize their utility by adjusting the percentage of test commands relative to actual attack commands, using equation (4) after updating their belief value. On the other hand, a honeypot is not able to determine the true nature of the botmaster's commands, i.e., whether the commands correspond to actual attacks targeting real victims or tests targeting sensors controlled by the botmaster. Consequently, the honeypot cannot update its best response strategy, q^*, over the time which indicates an inherent deficiency in the current honeynets' design. In what follows, we further discuss the impact of the parameters (E, L and R) used to calculate the optimal response strategy for both players.

- For situations where the honeypot penalty for participating in an attack is very high compared to the reward of the botmaster gained by discovering the honeypot true nature, e.g., in situations where legal liability is excessive, we have

$$p^* \approx 1, P_a \approx 1 - \mu(t), \text{ and } P_t \approx \mu(t).$$

Thus, in these situations, botmasters will use the belief value $\mu(t)$ to determine the percentage of test and attack commands. Higher values of the belief function lead to more test commands and less attack commands. In this case, the interaction between the botmaster and honeypot can be seen as an accelerated loop where the honeypot does not respond to tests and thus the belief value increases, which means that more tests are sent. This will significantly expedites the honeypot detection process.

- If the penalty of the honeypot for participating in real attacks is much less than the botmaster's reward for disclosing the honeypot, i.e., $L \ll E$, then p^* will assume a small value and the botmaster will tend to send less tests compared to the previous case. This is because the honeypot is expected to participate more in executing the botmaster's commands, which enables the botmaster to make use of this good-participating machine in executing real attacks rather than spending time and effort on sending many tests to disclose its true nature.

To simplify our analysis, throughout the rest of this section, we assume that $C_a \approx C_t$ and hence we have $q^* \approx \frac{E}{E+R}$. In general, q^* is effected by both the botnet size and honeypot penetration percentage. In large botnets, the revenue obtained by the botmaster when a given machine participates in an attack, R, would typically be small. Also, the reward E of removing a honeypot from botnet increases if the number of honeypot is small (e.g., if the botnet is infiltrated with only one honeypot then removing this honeypot will prevent all the information leakage from this botnet).

- For $E \ll R$, (e.g., in the case of a small botnet that is highly infiltrated with honeypots), q^* will have a small value, $q^* \ll 1$. In other words, honeypots will not participate in executing the botmasters' commands most of the time because there are so many honeypots gathering information about the botmaster and the botnet. Consequently, discovering and removing one of these honeypots does not have a big impact on the honeynet performance and hence honeypots tend to avoid the penalty of participating in real attacks by reducing their contribution percentages.
- For $E \gg R$, (e.g., in the case of a large botnet with few honeypots), the percentage of contribution increases because security professionals need to keep their honeypots in the botnet.

IV. Simulation Results

In this section, we present our simulation results that show the effect of different parameters on the temporal variation of the belief function of the botmaster as well as on the optimal probability of sending real attack commands by a rational botmaster.

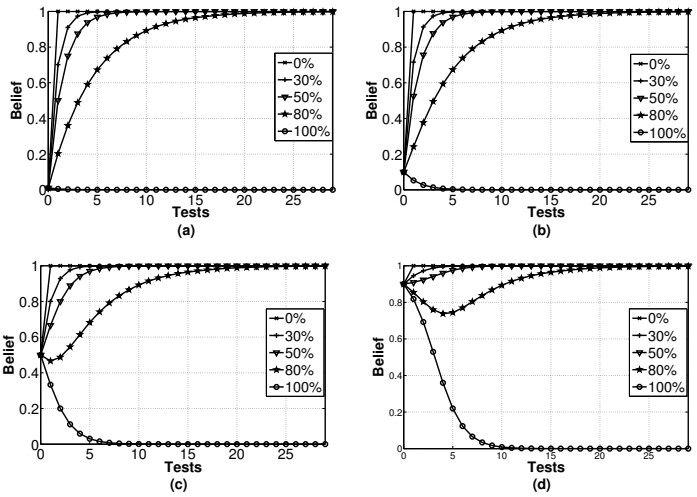

Fig. 1. Botmaster belief for $\mu_0(\theta_h) = 0.01, 0.1, 0.5, 0.9$ and $U = 0.23$

A. Belief Evaluation

Assuming that a normal machine responds to attack/test commands only 5 days/week, 8 hours/day and ignoring other temporarily networking problems such as possible firewall blocking, then $U = \frac{8}{24} \times \frac{5}{7} \approx 0.23$. Figure 1 shows the variation of $\mu_t(\theta_h)$ for different contribution percentages by the honeypot for botnets with different infiltration percentage (from low to high) as reflected by the botmaster initial value of the belief function $\mu_0(\theta) = 0.01$, 0.1 and 0.5 and 0.9, respectively. As depicted in the figure, the value of the initial belief does not have a big impact on the evolution trend of the botmaster's belief. It is also clear that when the honeypot does not respond to the botmaster command, or responds to it in small percentage of times less than U, then the botmaster is able to detect its true identity using a relatively small number of tests. On the other hand, it is interesting to note that increasing the participation percentage of the honeypot in the attacks is not always a good strategy to avoid detection by the bot master. In particular, when the honeynet responds to the attack/test commands by probability much greater than U, then the botmaster is also able to easily detect its true nature. In other words, to avoid detection by the botmaster, what is really important is to mimic the behavior of a normal machine. From the above simulations, we can conclude the following:

- To prolong its stay in a botnet, the honeypot should respond with probability $U \pm \epsilon$ for small values of ϵ. Furthermore, in order to improve its utility, the honeybot is better to respond with $U - \epsilon$.
- It is easier for the honeypot to evade detection in botnets where U has a small value, e.g., when the compromised machines are in different geographical locations and thus are not presumably all active on the same time.

B. The Impact of Changing the Honeypot Sequence of Actions

In this section, we study the impact of different distributions of honeypot actions (contribute and not-contribute) for the same contribution percentage. We consider a 50% contribution

percentage for the honeypot, i.e., the honeypot is assumed to participate in half of the commands received from the botmaster. We also assume that $U = 0.7$, i.e., a normal machine is expected to respond to 70% of the botmaster commands. We compare the effect of different sequence of actions for the honeypot by counting the number of tests required to reach a specific belief value threshold (a threshold belief value of 0.8 is be used in the following illustrating example).

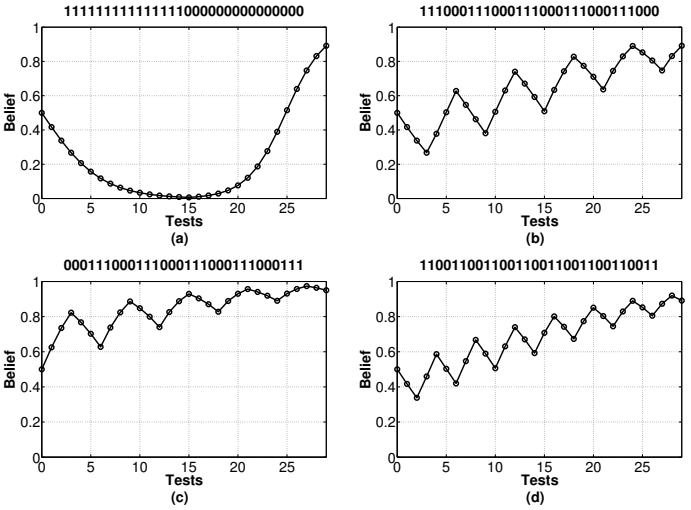

Fig. 2. Belief evaluation with different contribution schemes

Figure 2 shows the honeypot response as a series of ones (Contribute) and zeros (Not Contribute). Under our assumptions, the best strategy for the honeypot is to make all of its contributions at the beginning then to stop contributing as shown in Figure 2-(a). However, this is not possible in practice since the honeypot cannot predict how many tests the botmaster will perform. Figures 2-(b) and 2-(c) show similar sequence of actions where the honeypot starts with contribute in (b) and with not-contribute in (c). The belief value of 0.8 is reached after three tests only in (b) and fifteen tests in (c). The best *realistic* contribution decision in this scenario is in Figure 2-(d) '111000111000...' where the belief value reaches 0.8 after eighteen tests. This simulation shows that:

- Irrespective of the honeypot sequence of actions, the botmaster will eventually evaluate the same value of the belief function as long as the same percentage of contribution is used by the honeypot. The only effect of changing sequence of actions is to delay reaching a higher belief value, which may encourage the botmaster to send more real attack commands to the honeypot for a longer time.
- Honeynets need to enhance their hiding capabilities by making intelligent decisions before deciding their response to botmasters' commands. For example, if the target of the botmaster command is a popular reputable server, then this command would most likely correspond to a real attack. Consequently, future honeypots should be designed to achieve a better decision by inquiring some side channel information about the target of the botmaster command such as geographical location and IP reputation (e.g., by querying online services such as *WatchGuard ReputationAuthority* available at www.reputationauthority.org).

V. Conclusion

Honeypots are great tools for security professionals in their battle against botmasters. In this work, we developed a game theoretic framework to analyze the interaction between security professionals, who are trying to infiltrate botnets using honeybots, and botmasters who are trying to detect the presence of honeypots in their botnets. The results obtained from our analysis allow the security professional to optimally decide on the best response to probes sent by botmaster in order to disclose their honeypots. Our analysis also shows some inherent weakness in the current design of honeypots; since botmasters are able to probe the honeypots with test/attack commands, botmasters are able to update their belief about the machines they are interacting with and consequently optimally decide on the optimal mix between real attack commands and test attack commands. On the other hand, honeypots' decisions cannot be systematically optimized over the time because the true nature of the botmaster probes are not known to them.

References

[1] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *Proc. 14th ACM Conference on Computer and Communications Security*, pages 375–388, 2007.

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

[3] C. Zou and R. Cunningham. Honeypot-aware advanced botnet construction and maintenance. In *Proc. International Conference on Dependable Systems and Networks (DSN)*, pages 199–208, 2006.

[4] 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.

[5] N. Krawetz. Anti-honeypot technology. *IEEE Security & Privacy Magazine*, 2(1):76–79, 2004.

[6] 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.

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

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

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

[10] A. Srivastava and J. Giffin. Tamper-resistant, application-aware blocking of malicious network connections. In *Proc. 11th RAID, Berlin, Heidelberg*, 2008.

[11] K. Binmore. *Game Theory A Very Short Introduction*. Oxford University Press, 2007.

[12] R. Vogt, J. Aycock, and M. Jacobson. Army of botnets. In *Proc. Network and Distributed System Security Symposium NDSS07*, pages 111–123, 2007.