AHEAD: A New Architecture for Active Defense *

Fabio De Gaspari
Dip. di Informatica
Sapienza University
Roma, Italy
degaspari@di.uniroma1.it

Sushil Jajodia
CSIS
George Mason University
Fairfax, VA, USA
jajodia@gmu.edu

Luigi V. Mancini
Dip. di Informatica
Sapienza University
Roma, Italy
mancini@di.uniroma1.it

Agostino Panico
Dip. di Informatica
Sapienza University
Roma, Italy
panico@di.uniroma1.it

ABSTRACT

Active defense is a popular defense technique based on systems that hinder an attacker's progress by design, rather than reactively responding to an attack only after its detection. Well-known active defense systems are honeypots. Honeypots are fake systems, designed to look like real production systems, aimed at trapping an attacker, and analyzing his attack strategy and goals. These types of systems suffer from a major weakness: it is extremely hard to design them in such a way that an attacker cannot distinguish them from a real production system. In this paper, we advocate that, instead of adding additional fake systems in the corporate network, the production systems themselves should be instrumented to provide active defense capabilities. This perspective to active defense allows containing costs and complexity, while at the same time provides the attacker with a more realistic-looking target, and gives the Incident Response Team more time to identify the attacker. The proposed proof-of-concept prototype system can be used to implement active defense in any corporate production network, with little upfront work, and little maintenance.

CCS Concepts

•**Security and privacy** → **Intrusion detection systems; Network security;** *Software and application security;*

Keywords

Active Defense, Intrusion Detection System, Cyber Deception, Honeypot, Honeytoken

*This work has benefited in part from the activities of the NATO Research Task Group IST-152 on Intelligent Autonomous and Trusted Agents for Cyber Defense and Resilience. This work was partially supported by the Army Research Office under grants W911NF-13-1-0421 and W911NF-13-1-0317, and by the Office of Naval Research under grant N00014-13-1-0703.

SafeConfig'16, October 24 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2994475.2994481

1. INTRODUCTION

In recent years, the escalation of the power that attackers can muster and the rapid evolution of their strategies, resulted in classical reactive defense techniques (such as antivirus and intrusion detection and prevention systems) losing most of their efficacy. Indeed, while these tools still serve some purpose, they did not keep up with the increase in sophistication of attacks and are far from being sufficient to secure computer systems. Active defense is an alternative approach to protecting computer systems based on the concept of actively hindering and deceiving an attacker, rather than trying to block him out [17, 18, 19, 20]. Differently from classical defense systems that are designed to close all possible attack vectors in order to keep attackers outside target systems, active defense techniques are designed to confound and slow down an attacker that already has access to the system. One of the most well-known active defense measure is honeypots [22]. Honeypots are mock systems designed to look like real production systems. The goal of honeypots is to provide an attacker with an attractive target to attack, keeping the real production systems safe. Additionally, honeypots are instrumented to log all the actions of the attacker, so that the Incident and Response Team (*IRT*) can better understand his goal and how to react. While honeypots are an improvement compared to classical defense techniques, they also suffer from some major disadvantages. Indeed, properly configuring a honeypot so that an attacker can not distinguish it from a real production systems is extremely challenging [15]. Many published works analyze this problem and propose different solutions to make honeypots look more real [15]. Unfortunately, these works are all purely theoretical and, in general, are not directly applicable to real world scenarios. Indeed, there are several techniques an attacker can employ to easily detect both low-interaction and high-interaction honeypots [21]. In this paper, we propose an alternate approach to active defense called Attackers Hindered by Employing Active Defense (*AHEAD*). Instead of trying to build mock systems that look like real systems, the real production systems themselves should be instrumented to provide active defense and deception capabilities. The AHEAD approach allows to slow down (*annoyance*) the attack, providing IRT with enough time to monitor and identify (*attribution*) the intruder while he is still trying to map the target production system. It is worth noting that AHEAD does not increase the exposure of the network to the attacker, and does not require to expose a higher number of production systems to external clients. Indeed, AHEAD simply provides pre-existing production systems with active

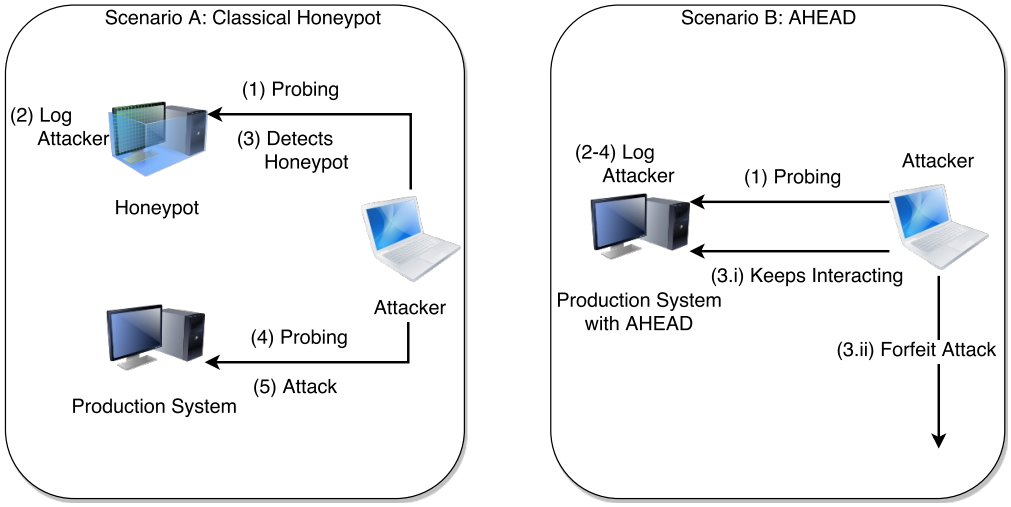

Figure 1: Comparison between a classical honeypot system and AHEAD.

defense capabilities, without requiring any modification to the architecture of the network or the production systems themselves. Moreover, AHEAD provides layered defense in case an attacker manages to penetrate the network. In fact, if an attacker obtains access to a production system machine on the outer perimeter of the network, the attacker will still be faced with AHEAD deployed on the inner systems of the network, which will considerably slow down his progress. Finally, AHEAD does not necessarily replace honeypots, but can be deployed in a corporate network alongside them.

This new perspective provides several advantages over classical honeypot-based defense systems.

Cost effectiveness. Honeypots are expensive. Indeed, beside the cost of the new physical honeypot machines that need to be added to the network, there is the cost of configuring, deploying and managing a considerably more complex network infrastructure. Moreover, honeypots need to be configured such that they look like real production systems, which considerably increases the costs and the complexity of the configuration. AHEAD does not suffer from this limitation, since it *is* a real production system and the only physical systems that are needed to implement it are the pre-existing production systems.

Reduced complexity. As we already discussed, honeypots are complex systems. Configuring and deploying several complex systems within an already complex corporate network might create new vulnerabilities due to potential oversights or unforeseen interactions between various components. Moreover, introducing additional systems also increases the complexity of maintaining the corporate network. AHEAD reduces the complexity of integrating active defense in a corporate network since it does not require complex re-engineering of the network. At the same time, AHEAD reduces maintenance complexity by virtue of the fact that it does not require additional machines in the network.

Easier Network Design. It is not easy to find the most effective network layout to maximize the chances of the attacker choosing the honeypot as target of the attacks [23]. AHEAD does not suffer from this drawback, since the production system itself becomes a honeypot. There is no need to make the production system attractive to try to lure the attacker to it.

To summarize, our contributions in this paper are as follow:

- We propose a new active defense approach based on deception, AHEAD, which avoids most of the common pitfalls of previous active defense approaches.
- We implement a proof-of-concept prototype of AHEAD to prove the feasibility of our approach.

The AHEAD approach is radically different from previous works and funded projects on honeypots. Rather, AHEAD can be seen as a generalization of the *Honey-Patches* approach [14], where vulnerabilities are patched in such a way that the failed exploit appears to succeed, while the attacker is in fact redirected to an ad-hoc environment. However, the Honey-Patches approach is limited only to known, patched vulnerabilities, reducing its applicability. Another relevant proposal is that of *shadow honeypots* [13]. This approach requires two distinct systems: the real production system and an instrumented shadow copy of the real system. In this scenario, network flows that are marked as suspicious by an anomaly detection sensor are forwarded to the shadow copy, while legit flows are directed to the real system. Shadow honeypot approaches rely on the assumption that malicious network flows will be forwarded to the shadow honeypot, rather than to the real system, and are therefore reliant on the correct classification by the anomaly detection sensor. Since anomaly detection systems often misclassify traffic [16], the effectiveness of this approach is limited. AHEAD does not suffer from this drawback, since the active defense tools are deployed on the production system itself, and therefore the attacker does not need to be redirected to it. When the attacker probes a target machine to find a suitable service to attack, he is slowed down and confounded by the high number of services offered by AHEAD.

The remainder of this paper is organized as follows. In Section 2 we discuss a usage scenario of AHEAD, highlighting its main differences with respect to traditional approaches. In Section 3 we present the abstract architecture of the AHEAD framework. In Section 4 we illustrate our

proof-of-concept prototype of the AHEAD system and highlight its main features. Finally, in Section 5, we draw our conclusions.

2. USAGE SCENARIO

In this section we describe a usage scenario of AHEAD. Let us consider an attacker who wants to attack some production systems on a target network. We distinguish the two scenarios depicted in Figure 1: *(A)* a network protected by a classical honeypot and *(B)* a network protected with AHEAD. Before performing any attack, the attacker will have to perform reconnaissance on the network in order to identify valuable targets. Let us assume that, in order to reach this goal the attacker performs a network scan to identify existing systems and services.

(A) Classical honeypot. In scenario (A), the network scan will eventually reach the honeypot (step 1). At this point, if the configuration of the honeypot is realistic enough, the attacker will start attacking one of the available services provided by the honeypot. The attack is detected by the honeypot (step 2), and the IRT will be notified that something anomalous is going on in the network. Unfortunately, the attacker will eventually realize that the target is indeed an honeypot (step 3) and will move on to attacking one of the remaining systems (step 4). In this scenario, the limitation of the honeypot approach from the point of view of the IRT is that the interaction between the attacker and the honeypot is extremely limited in time, often in the order of seconds. Indeed, after the attacker leaves the honeypot and moves on to another system, the IRT loses the chance to monitor the attacker and devise a proper identification and defense strategy.

(B) AHEAD. On the other hand, when AHEAD is employed, the attacker will have to sift through fake services and mock vulnerabilities (step 1) in order to try to compromise the production system, forcing him to interact with AHEAD for a considerably longer time (step 3.i). This provides the IRT with enough time to analyze the logs generated by AHEAD (step 2) and decide how to counter the attack, as well as provide more material to analyze the strategy of the attacker after the attack has concluded (step 4). This additional information allows IRT to improve the attribution of the attack and the security of the corporate network and systems, adapting them to ever-evolving attack strategies. Moreover, AHEAD can also work as a deterrent. Indeed, if the attacker realizes the real production system is heavily monitored and instrumented, he might also choose to forfeit the attack in order to protect himself (step 3.ii). In both cases, the corporate network is protected.

3. THE ARCHITECTURE OF THE AHEAD FRAMEWORK

From an architectural perspective, the AHEAD framework is composed by two components: the *AHEAD Controller* and *AHEAD-pot*. The AHEAD Controller includes an administrator interface; it is the single point of interaction with the AHEAD-pots and allows to interact with them over a secure channel. The AHEAD-pot is the component effectively implementing the active defense countermeasures, and is deployed on the production systems. In a real-world scenario, several AHEAD pots are deployed in a corporate network, covering all components of the information system (see Figure 2). The AHEAD pots are encapsulated in a sandbox and therefore do not interfere with the production services, while at the same time having a low overhead on the production system itself. Additionally, the AHEAD framework is tightly integrated with pre-existing security information and event management (SIEM) systems: the AHEAD-pots constantly send activity logs to the SIEM systems, allowing the IRT to improve other security components already deployed (e.g., intrusion detection/prevention systems, firewall). The feedback from the AHEAD-pots is also used by the IRT to identify what additional active defense modules need to be deployed in the Pots themselves, so that the system can dynamically adapt to emerging threats.

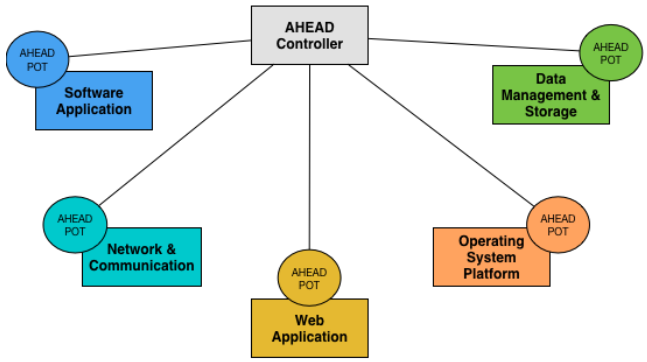

Figure 2: Integration of AHEAD in the architecture of a typical corporate information system.

In order to be useful in a real corporate information system, the AHEAD framework must satisfy a number of important requirements, which include: scalability, maintainability, security, modularity, expandability and interoperability. In Section 4, we illustrate how our proof-of-concept prototype achieves this requirements.

4. IMPLEMENTATION OF THE PROPOSED PROTOTYPE

In order to show the feasibility of AHEAD, we implemented a proof-of-concept prototype that satisfies the above mentioned requirements. The proposed prototype can run embedded in a production environment and covers all the subsystems of the corporate IT Infrastructure illustrated in Figure 2.

We based our prototype on the Docker platform [5]. In our architecture, Docker provides our prototype with a first layer of security through the use of *containers*. Indeed, our AHEAD-pot prototype runs inside a Docker container, which is akin to a sandbox: an attacker entering the container can not interact with other applications running in that machine, but only with applications running within the container itself. In our case, that is only the AHEAD-pot prototype, since it is the only application running inside the container. This guarantees that, during an attack, an attacker can not reach the real production services through one of the deployed active defense tools (see Figure 3). Moreover, our prototype provides additional security through a second layer of isolation between the attacker and the real

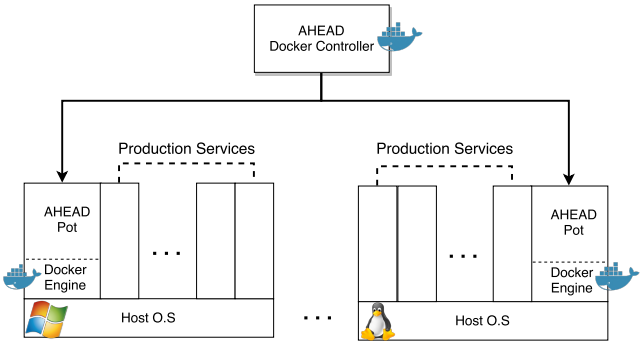

Figure 3: Overview of the architecture of the AHEAD prototype.

applications running in the machine. Indeed, each individual active defense tool running in our AHEAD-pot prototype is designed to be attacked and already implements containment measures against attackers. An attacker would have to escape the containment of both the active defense tool and Docker, before reaching the real production system (see Figure 4).

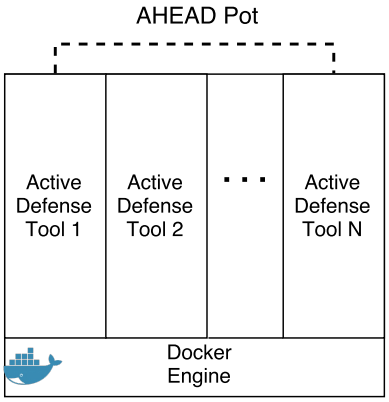

Figure 4: Architecture of an AHEAD-pot, running on top of the Docker engine.

Therefore, our AHEAD-pot prototype does not open additional venues for attack, but rather strengthens the security of the production system on which it is deployed. Beside security, Docker also satisfies the other requirements listed in Section 3: Docker is not platform dependent, it scales on-demand and it allows for easy configuration of the containers. Moreover, many containers can be deployed in parallel so modularity and extendability requirements are also satisfied. Finally, a preliminary analysis of our proof of concept implementation shows that AHEAD requires a small amount of resources of the production system. A more accurate analysis of the overhead introduced by AHEAD is deferred to future work.

For our proof-of-concept prototype, we focused only on the annoyance and attribution aspects of active defense. Our prototype is designed to slow down an attacker and to obtain evidence that can be used to identify him. However, the prototype can be easily extended to include additional countermeasures pertaining to other aspects of active defense, like attacking back.

4.1 Active Defense Tools Employed

In this section, as an example, we present an overview of some of the tools we employed and integrated in our AHEAD-pot prototype. In particular, we describe how our prototype integrates these tools to cover all the subsystems present in a typical corporate information system (see Figure 2). Each of the following subsections describe the set of tools that the AHEAD Controller would deploy in the AHEAD-pots in order to protect a specific subsystem in the corporate network. For instance, Section 4.1.3 describes which tools would be deployed in an AHEAD-pot designed to protect the Data Management and Storage subsystem of the corporate infrastructure. It is worth nothing that, while the following tools are all implemented in a single Docker component, the AHEAD controller can dynamically deploy and configure a subset of these tools that is suitable for the desired purpose. For this proof-of-concept prototype, our selection of active defense tools was also inspired by the Harbinger distribution [6].

4.1.1 Software Application

Kippo. We provided our prototype with a fake SSH service through the tool Kippo [8]. With Kippo, our prototype is able to monitor the entire SSH session of the attacker, logging strategies and exploits used. We integrated Kippo with the SIEM system, providing real time alerts to the IRT.

Rubberglue. Our prototype uses the Rubberglue [10] tool to provide annoyance and attribution capabilities. Through Rubberglue, the prototype exposes an open port through which the attacker can connect. When the attacker sends traffic to this port, the system automatically redirects it back to the same port on the machine of the attacker. Effectively, the attacker ends up attacking his own machine, while our prototype logs all activity and forwards the logs to the SIEM. This tool can be easily extended to provide our prototype with attack capabilities.

4.1.2 Operating System Platform

Artillery. We employed Artillery [1] in our prototype to improve detection and identification rates. Through artillery, our prototype exposes a set of fake services that provide access to the sandboxed filesystem of the AHEAD-pot. These sections of the filesystem are heavily monitored in order to immediately detect any change to directories or files. Additionally, we integrated Artillery with Web Bug Server (see Section 4.1.3) to provide also attribution capabilities. Artillery is integrated with the SIEM and generates an alert upon detection of unexpected activities.

Dionaea. Through Dionaea [4], our prototype is able to trap and copy malware executables. By integrating Dionaea with other active defense tools, like Portspoof and Artillery, our prototype is able to obtain a copy of the malware executable used to attack the fake services these tools provide. The executable obtained can later be analyzed to patch vulnerable systems and to find evidence that can be useful in the attribution phase.

4.1.3 Data Management and Storage

CryptoLocked. With CryptoLocked, we provided our prototype with a failsafe system to monitor the files of the real production system, in case an attacker manages to obtain access to it (e.g., an insider). Our prototype uses CryptoLocked to monitor a set of fake files with attractive con-

tent in the filesystem of the production system. In case of modifications to these files, our prototype generates an alert for the SIEM system and implements contingency measures, such as encryption of sensitive files in the production system or shutdown of the machine.

Web Bug Server. We employed Web Bug Server [11] to improve the attribution capabilities of our prototype. We setup Web Bug Server to generate document files that look attractive to the attacker. These documents embed web beacons, that are invisible to a casual observer, by using techniques such as linked style sheets and 1 pixel images. Once the documents are opened by the attacker, the beacons send a request to the Web Bug webserver, exposing the real identity of the attacker.

4.1.4 Network Communication

Portspoof. In our prototype, we used Portspoof [9] to open fake services that are bound to a set of open ports in the production system. When an attacker interacts with one of these fake services, all his actions are monitored, logged and sent to the SIEM system so that the IRT can devise an appropriate defense strategy. Additionally, we integrated Portspoof with HoneyBadger (see Section 4.1.5) in order to disclose the real IP address of the attacker. This provides our prototype with both annoyance and attribution capabilities.

Conpot. We integrated the Conpot [2] industrial control system honeypot in our prototype. We setup Conpot listening on an open port accepting the de-facto standard Modbus protocol, so that our AHEAD-pot prototype can simulate a set of industrial control systems. This allows the prototype to monitor the activity and learn the motives of attackers targeting supervisory control and data acquisition (SCADA) systems. Like all other tools we employed, Conpot is integrated with the SIEM system in order to promptly alert IRT in case of an attack.

4.1.5 Web Application

Weblabyrinth. In our prototype, we implemented active defense for web applications through integration of Weblabyrinth [12] with the production web server. Through Weblabyrinth, our prototype can generate an intricate maze of connected web pages aimed at trapping and confounding malicious spiders. For instance, by including a Weblabyrinth link in the *robot.txt* file of the production web server, we can lure malicious spiders into crawling through all the fake webpages. Additionally, the logs generated are automatically forwarded to the SIEM system.

HoneyBadger. In order to provide our prototype with attribution capabilities, we integrated the HoneyBadger tool [7]. Through HoneyBadger, our prototype can identify the physical location of a web attacker through a combination of several techniques. Additionally, we improved the default HoneyBadger tool to include client-side technologies (e.g., JavaScript), honeytokens and to use powershell to provide more accurate identification of the attacker.

Decloak. Our prototype supports robust attribution through the integration of Decloak [3]. Decloak allows our prototype to obtain the IP address of a web attacker, independently from potential proxy settings, through a combination of HTML tags and JavaScript code. We integrated Decloak with the fake services offered by Portspoof and Artillery.

5. CONCLUSIONS AND FUTURE WORK

In this paper we presented AHEAD, a new approach to active defense. AHEAD allows to reduce costs, complexity and to ease the integration of active defense in corporate information systems and networks, while at the same time providing attackers with a more attractive and realistic target than classical honeypots. Additionally, we described and discussed the strengths of our prototype implementation of AHEAD. In our future work, we plan on employing machine learning techniques to improve AHEAD with the ability to automatically deploy additional active defense tools based on the behaviour of the attacker. This will allow AHEAD to learn from previous attacks and provide it with the ability to select the best countermeasure against a given type of attack. Finally, in our future research we intend to address how to protect the functionality of AHEAD against an attacker who obtained control over the machine where AHEAD is running.

6. REFERENCES

[1] Artillery. https://github.com/shoreditch-ops/artillery.

[2] Conpot. https://github.com/mushorg/conpot.

[3] Decloak. https://github.com/cmlh/decloak.

[4] Dionaea. https://github.com/rep/dionaea.

[5] Docker platform. https://www.docker.com/.

[6] Harbinger distribution. http://www.blackhillsinfosec.com/?page_id=4419.

[7] Honeybadger. http://github.com/honeybadger-io/honeybadger-ruby.

[8] Kippo. https://github.com/desaster/kippo.

[9] Portspoof. https://github.com/drk1wi/portspoof.

[10] Rubberglue. https://github.com/adhdproject/adhdproject.github.io/blob/master/Tools/Rubberglue.md.

[11] Webbugserver. https://github.com/adhdproject/adhdproject.github.io/blob/master/Tools/WebBugServer.md.

[12] Weblabyrinth. https://github.com/mayhemiclabs/weblabyrinth.

[13] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting targeted attacks using shadow honeypots. In *Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14*, SSYM'05, pages 9–9, 2005.

[14] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser. From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, pages 942–953, 2014.

[15] M. L. Bringer, C. A. Chelmecki, and H. Fujinoki. A survey: Recent advances and future trends in honeypot research. In *International Journal of Computer Network and Information Security*, IJCNIS, 2012.

[16] R. Di Pietro and L. V. Mancini. *Intrusion Detection Systems*, volume 38 of *Advances in Information Security*. Springer, 2008.

[17] S. Jajodia, K. A. Ghosh, V. Subrahmanian, V. Swarup, C. Wang, and S. X. Wang, editors. *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*. Springer, 2013.

[18] S. Jajodia, K. A. Ghosh, V. Swarup, C. Wang, and S. X. Wang, editors. *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. Springer, 2011.

[19] S. Jajodia, P. Shakarian, V. Subrahmanian, V. Swarup, and C. Wang, editors. *Cyber Warfare: Building the Scientific Foundation*. Springer, 2015.

[20] A. Kott, C. Wang, and F. R. Erbacher, editors. *Cyber Defense and Situational Awareness*. Springer, 2014.

[21] N. Provos and T. Holz. *Detecting Honeypots*, chapter in book: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, 2007.

[22] Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley Longman, 2002.

[23] S. Tapaswi, A. Mahboob, A. S. Shukla, I. Gupta, P. Verma, and J. Dhar. Markov chain based roaming schemes for honeypots. *Wirel. Pers. Commun.*, pages 995–1010, 2014.