Evaluating Deception and Moving Target Defense with Network Attack Simulation

Daniel Reti
German Research Center for Artificial Intelligence
Kaiserslautern, Germany
daniel.reti@dfki.de

Karina Elzer
German Research Center for Artificial Intelligence
Kaiserslautern, Germany
karina.elzer@dfki.de

Daniel Fraunholz
Independent Researcher
Munich, Germany
daniel.fraunholz@gmx.de

Daniel Schneider
German Research Center for Artificial Intelligence
Kaiserslautern, Germany
daniel.schneider@dfki.de

Hans Dieter Schotten*
German Research Center for Artificial Intelligence
Kaiserslautern, Germany
hans_dieter.schotten@dfki.de

ABSTRACT

In the field of network security, with the ongoing arms race between attackers, seeking new vulnerabilities to bypass defense mechanisms and defenders reinforcing their prevention, detection and response strategies, the novel concept of cyber deception has emerged. Starting from the well-known example of honeypots, many other deception strategies have been developed such as honeytokens and moving target defense, all sharing the objective of creating uncertainty for attackers and increasing the chance for the attacker of making mistakes. In this paper a methodology to evaluate the effectiveness of honeypots and moving target defense in a network is presented. This methodology allows to quantitatively measure the effectiveness in a simulation environment, allowing to make recommendations on how many honeypots to deploy and on how quickly network addresses have to be mutated to effectively disrupt an attack in multiple network and attacker configurations. With this optimum, attacks can be detected and slowed down with a minimal resource and configuration overhead. With the provided methodology, the optimal number of honeypots to be deployed and the optimal network address mutation interval can be determined. Furthermore, this work provides guidance on how to optimally deploy and configure them with respect to the attacker model and several network parameters.

CCS CONCEPTS

• **Security and privacy → Network security**.

KEYWORDS

network security, cyber deception, moving target defense, honeypot

*Also with Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern.

MTD '22, November 7, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9878-7/ 22/11…$15.00
https://doi.org/10.1145/3560828.3564006

ACM Reference Format:
Daniel Reti, Karina Elzer, Daniel Fraunholz, Daniel Schneider, and Hans Dieter Schotten. 2022. Evaluating Deception and Moving Target Defense with Network Attack Simulation. In *Proceedings of the 9th ACM Workshop on Moving Target Defense (MTD '22), November 7, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3560828.3564006

1 INTRODUCTION

In the past decades, many effective strategies for improving the security of Information Technology (IT) have come forward. Whether on the network level such as firewalls, Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) or on the system level such as kernel hardening, antivirus and User Access Control. Nonetheless, many reports show that networks and systems are still being compromised despite the security measures in place. There are multiple reasons that are complex to mitigate, such as zero-day-exploits or malicious or naïve insiders. Thus, it is becoming clearer that concepts for defense-in-depth and resiliency need to be researched and ultimately deployed to aid the overall level of security. In parallel, the field of cyber deception is gaining relevance, as it addresses multiple problems in state of the art cyber defense. Cyber deception is seeking to distract adversaries from real targets, and present simulated targets instead, such as honeypots. This also addresses the traditional asymmetry between attackers and defenders, meaning that the defender has to be successful all of the time and the attacker is only required to be successful once. In this paper, a network attack simulation is used to quantify the effectiveness of the presence of honeypots and Moving Target Defense (MTD), i.e. mutated network addresses, in networks of various sizes. The quantification is done by defining three attack strategies and for each strategy it is measured how likely it is for an attacker to successfully exploit a vulnerable host, when the different deception setting are in place. The contribution of this paper is the following:

- A methodology for optimizing honeypot and MTD deployment and effectiveness based on an enhanced version of a network attack simulation platform
- Introduction of various configuration options for the simulation to enable researcher and practitioners to tweak our methodology to their needs

- A quantitative study, comparing of the impact of various network, honeypot and MTD parameters based on the presented methodology

The remainder of this work is structured as follows: Section 2 gives an overview on the state of the art in cyber deception research. In Chapter 3 our Methodology is described, followed by the simulation environment in Chapter 4. In Chapter 5 the results are presented and a conclusion is given in Chapter 6.

2 CYBER DECEPTION

The term *cyber deception* describes a defense strategy of cyber security that is based on simulating fake targets and hiding real targets. While many cyber deception strategies exist, this section focuses on the popular techniques of honeypots and moving target defense, as those are used in this work. An more extensive overview is presented by Fraunholz et al. [8].

2.1 Honeypots

Honeypots are simulating hosts on a computer network which only serve as a decoy for attackers. Therefore any connection with a honeypot may be considered suspicious. A distinction can be made between low-, mid- and high-interaction honeypots[8]. While low interaction honeypots can reliably detect a host discovery scan and simple service interactions, a high interaction honeypot allows the attacker to take over and fully interact with the operating system. This allows researchers to learn about the behavior and capabilities of the attacker. A medium interaction honeypot is a compromise between high- and low interaction honeypot and typically allow interaction with services but not with the operating system. A distinction can also be made between production honeypots and research honeypots. While production honeypots are popular in corporate networks for attack detection and gaining insight on attacks, research honeypots can gather more attack specific data and involve higher maintenance costs. Many honeypot projects exist, differing in respect to which service they are emulating, how they are deployed and how much interaction is possible. Well known honeypot projects include Kippo [28], Honeyd [20], Glastopf [21] and Conpot [22], which simulate open SSH ports, TCP ports, Web Applications and industrial control systems, respectively.

Another variation of honeypots, which do not simulate a host, but rather specific details, have been coined *honeytoken* [6]. Such honeytoken are simulating a digital asset which looks interesting to the attacker. These can be for example decoy files, databases, credentials, or websites [30] [14]. Honeytoken, like honeypots, make the reconnaissance phase, the earliest stage of an attack, difficult by creating uncertainty and increasing the chance of being detected. This uncertainty was proposed to be exploited, by proposing to make production systems appear like honeypots, namely fake honeypots [23].

2.2 Moving Target Defense

Similarly to the objectives of honeypots, MTD is aiming to create uncertainty for the attacker and waste their resources. To achieve these objectives, in MTD the attack surface is constantly shifting. This can take various shapes and is already established for Address Space Layout Randomization (ASLR) to better protect the memory from buffer overflow attacks or in radio networks channel hopping is used against jamming [18]. Enabled through Software Defined Networking, constant mutation of network addresses, also referred to as random host mutation, has been proposed by different authors [9, 13, 15, 26, 27]. However, as with most security measures, the trade-off between security and performance needs to be taken into account [1, 16]. An extensive survey on MTD is presented in [24], highlighting the use of MTD against Advanced Persistent Threats and noting the extensive use of Artifical Intelligence and game theory in existing MTD approaches.

3 METHODOLOGY

In this section, the three attacker models are explained that have been defined for further examination. Next, the simulation environment used for the data collection is described. The environment includes a state of the art simulation tool as well as the changes which have been implemented to make Honeypots and MTD possible. The overall methodology to the used approach is described as well.

3.1 Attacker Model

Three attacker agents with different assumptions on their behaviour have been defined and implemented. Each agent had a different tactic to attack the network. This tactic is implemented through fixed behaviour rules. The rules are separated into different phases. Parts of the decisions are chosen randomly. These choices allowed for a comparison between the agents.

The actions an agent can chose from are the following:

- Subnet scan: Find out which addresses on the network are in use for host discovery
- Service scan: Find open TCP ports on a given host and derive an estimation on which services are present on that host.
- OS scan: Fingerprint the operating system and kernel version of the host.
- Vulnerability scan: Probe the target services for known vulnerability fingerprints
- Process scan: Used for privilege escalation, read which processes are running on a host.
- Exploit/Privilege Escalation: Chooses one of many predefined exploits
- Wiretapping: A post-exploitation step to capture unencrypted credentials from the network traffic.

The first agent is the 'careful' agent. This agent pursues a comprehensive horizontal scan, by first gaining an overview of the whole network and all attack possibilities before deciding on what action will be used for the attack. The 'careful' agents' actions are separated into the following phases:

(1) The agent does a full set of all possible scans for all the existing hosts. This includes the subnet, service/port, vulnerability and Operating System (OS) scan.
(2) The agent chooses a random host. Then they pick the best exploit to gain user or root access. If the agent already has user access, it will do a privilege escalation instead.
(3) If the exploit was successful, the agent will perform a process scan if they obtained user rights. If the agent obtained root privileges, then the agent will do wiretapping.

(4) If Moving Target Defense is turned off or the agent does not detect it, they will repeat the steps from phase two until the end of the simulation. If Moving Target Defense is turned on, the agent will start with phase one.

The second agent is called the 'standard' agent. For this agent, the design aimed toward a more realistic approach. The agent focuses on vertically scanning and attacking one host before looking at the next one. The phases for the standard agent are the following:

(1) The agent performs a subnet scan to get the addresses of hosts connected to the network.
(2) The agent chooses a random host and only scans that host. They do a service, OS, vulnerability and process scan.
(3) Depending on the access level of the chosen host, the agent will perform an action from a list of exploits, privilege escalations or do wiretapping.
(4) If the action from phase three was successful, the agent goes back the phase two. Otherwise, the agent will continue in phase three until the host was successfully compromised the host or no exploits and privilege escalations are left. If Moving Target Defense is detected, the agent will start again from phase one.

The last agent being defined is the 'aggressive' agent. This agent does not scan hosts. Instead, the agent performs attacks without knowing if they will be successful. This could be the behavior of a computer worm, which tries to exploit the same vulnerability on as many hosts as possible without the necessity to do a port scan. These are the phases for the 'aggressive' agent:

(1) The agent performs a subnet scan.
(2) The agent chooses a random action from the list of exploits and privilege escalations.
(3) The agent will perform the action on all hosts in a random order.
(4) If the attack was successful, the agent will do wiretapping, and then it will go back to phase two.

4 SIMULATION ENVIRONMENT

To gather the data for analysis, NASim was used. Honeypots and Moving Target Defense were added as features. In this chapter, the implementation of these additions is described, as well as the parameters chosen for the simulation runs.

4.1 NASim

The NASim [2], short for Network Attack Simulator, is a program to simulate network penetration testing. It is a python program that implements Open AI Gym [5] to allow reinforcement learning. For this paper, this functionality has not been used. Instead, the capabilities of creating simulated computer networks and creating custom agents have been used. These agents have a clearer behaviour, which makes the comparison more comprehensibly.

NASim allows the creation of a network with variable amounts of subnets, hosts and firewalls. It includes scanning of subnets, services (port scans), processes and operating systems (OS). Furthermore, it implements attacks through exploits and privilege escalations. For each of these actions, it is possible to assign costs.

NASim defines a special type of host called a sensitive host. The agent's goal is to gain root access to all sensitive hosts. The simulation is considered successfully completed when the agent has exploited all sensitive hosts.

The modifications were built on an already expanded version of NASim. In this previously extended version, the authors already implemented specific vulnerabilities, a vulnerability scan as well as the concept of credentials and wiretapping as an attack for obtaining credentials.

4.2 Implementation of Honeypots and Network Address Mutation

For this paper, the functionality of NASim was expanded to include Honeypots. Honeypots are based on the implementation of sensitive hosts. If an agent exploits one honeypot host with either user or root privileges, the goal failed and the agent has lost.

Also, a second definition of the goal for the analysing was added. Instead of the agent needing to gain root privileges on all sensitive hosts, the second definition only needs one sensitive host to be successfully exploited for the agent to complete the goal.

Additionally to the Honeypot feature, a Moving Target Defense functionality was added. After a chosen time, hosts in the same subnet will randomly change their addresses. The time is defined by the actions taken and their associated costs. After this change, the agent does not know which hosts were already exploited.

To make more addressed available for the mutation, 'empty' hosts were introduced. 'Empty' hosts are not detectable or attackable. Their only purpose is to introduce additional addresses in the subnets. Thus only serve as an unused address on the network, and the exiting hosts can shift to theses adresses. Therefore the address space is bigger and the simulation is more realistic.

5 EXPERIMENTAL SETUP

For the test procedure, three different scenarios, based on different seed, were generated. It was decided which parameters of the simulation will be observed. Additionally, the termination conditions were set, defining when the agents would win and when they would lose.

First, a modified version of the random scenario generator from NASim was used to generate the scenarios. The scenario generator allows for great customisation of the environment with slight randomisation and without the need to create the scenarios manually.

The most important parameters for this paper are the ones which were decided to be changed and observed. These are the Honeypots in the network and the time until address mutation of the MTD takes place. These two parameters and their comparison among themselves and with each other build the main focus of our research work presented. The objective defines whether the attacker needs to successfully exploit one or all sensitive hosts to win. Furthermore, the number of active normal hosts in the network, the agents as described in section 3.1 and different random seeds that lead to different scenarios were identified. (Table 1)

The rest of the parameters were fixed throughout all simulations. The network included 256 addresses and two subnets, one only for the attacker. Since the focus was not layed on the firewall, no restrictions were included. The costs of each action was set to

Parameter	Description	Value
num_honeypots_options	Number of Honeypots	0, 2, 4, 6, 9, 10
movement_time_options	Movement Time	None, 25, 50, 75, 100
num_hosts_options	Number of normal hosts	10, 50
one_goal_options	Objective	True, False
seed_options	Random Seed	1234, 42, 24121997
agents	Attack behaviour of agents	careful, standard, aggressive

Table 1: All Possible Varied Parameter Values

one. A distinction was made between the reward for the sensitive, honeypot and normal hosts. Three hosts were defined as sensitive hosts and for each scenario ten possible services, processes, exploits, privilege escalations and vulnerabilities were given. The number of Operating Systems was set to one and credentials have not been considered. (Table 2)

The simulations can terminate in three different ways:

- The agent wins, which means the agent successfully exploited all necessary sensitive hosts. This is dependent on the 'one_goal' parameter if the agent needs to exploit only one sensitive host or all three.
- The agent loses. This means the agent exploited a honeypot.
- The agent takes too long, which is defined by the 'step_limit' parameter.

Finally, after choosing all parameters, we ran the simulations. Each combination of parameters was run 100 times to compensate for randomness in the scenarios and agent behaviours.

6 RESULTS

As described in the previous chapters, the simulation was done by the three types of attackers in randomly generated network scenarios. Hereby different settings were tested, with a varying number of hosts, honeypots and network address mutation frequencies. In the following, the results from our experiments are presented from three different perspectives, starting with a comparison of the different attacker types, followed by the impact of honeypots and lastly the impact of moving target defense is considered. It is important to state the results reflect an assessment of honeypots and MTD considering an example configuration, and thus can be adopted to any network specifics using the proposed methodology to suit the needs of researchers and practitioners.

6.1 Agent Performance

The three different attacker types implemented in the simulation were a careful agent, who does a full scan of the network before it runs attacks, a standard agent who will scan single hosts and run an attack when a vulnerability is identified and the aggressive agent who will try a single attack on all hosts in the network. An agent will lose, when a honeypot is exploited or the step limit, as

Parameter	Description	Value
num_sensitive	Number of sensitive hosts	3
num_services	Number of services	10
num_os	Number of OS	1
num_processes	Number of processes	10
num_exploits	Number of exploits	10
num_privescs	Number of privilege escalations	10
num_vulns	Number of vulnerabilities	10
num_creds	Number of credentials	None
r_sensitive	Reward for sensitive hosts	1000
r_honeypot	Reward for honeypots	-1000
*_cost	Action costs	1
exploit_probs	Success probability for exploits	1.0
privesc_probs	Success probability for privilege escalations	1.0
uniform	Uniform distribution of hosts	True
base_host_value	Reward for normal host	1
host_discovery_value	Reward for host discovery	1
step_limit	Maximum number of actions permitted	3000
addresses	Number of network addresses	256
subnets	Number of subnets	2

Table 2: All Fixed Parameter Values

presented in Table 2, is exceeded. Therefore an increasing amount of honeypots in the network will decrease the chances of all three agents to win. But also the careful agent is highly affected as it does a horizontal scan and might identify a honeypot as a potential targets. As can be seen in Figure 2a, the winning probability of the standard agent is falling earlier with an increasing number of honeypots, while careful agent's winning probability dropping below the standard agent at around 4 honeypots and very close to it with 10 and above. In respect to MTD, as shown in 2b, moving target defense, with a lower movement interval of 25, the careful agent's winning probability is 0, as the IP mutation is happening before the agent chooses to attack. The aggressive agent is performing better for movement intervals below 60, and close for higher intervals.

The standard agents has throughout the highest winning chance, which is due to relatively quick actions on a randomly picked target.

In Figure 1, the agents are compared with honeypots and moving target defense enabled simultaneously. All three agents have a decreasing winning probability with lower mutation intervals and with an increasing amount of honeypots. It can be seen, that the standard agent is performing slightly better on average, with a winning probability of 44%, compared to 26% with the careful and 30% with the aggressive agent.

Focusing on the number of steps needed by each agent, it can be seen in Figure 3, that when honeypots and MTD are in place, that the aggressive agent has its maximum below 1000 steps, the careful agent performs poorly with its median at 3000 steps, which means that it doesn't succeed most of the time, and the standard agent, which performs the best with a median and a maximum below 100 steps. For reference, when neither honeypots nor MTD is enabled, all three agents have a median close to 100 steps. When only honeypots are enabled, the number of steps drops, as an interaction with the honeypot end the round. When only MTD is enabled, the median of the aggressive and the careful agent rise to the maximum, while the standard agent has a low median, while its third quartile is at the maximum value of 3000 steps.

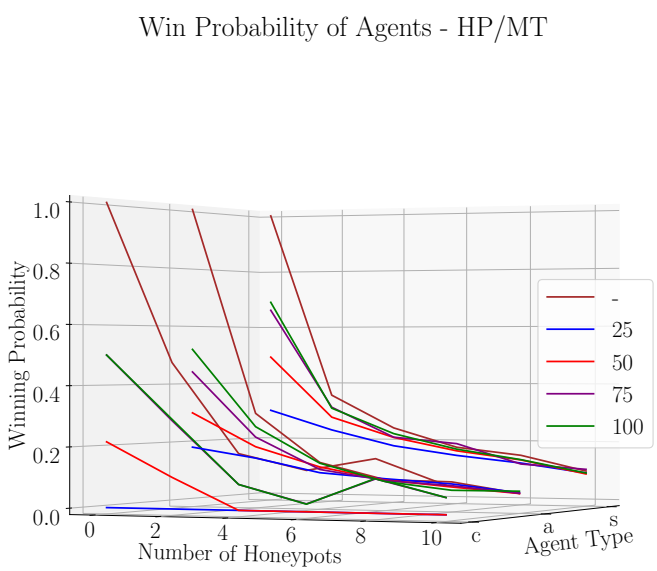


Figure 1: Winning probability for different IP changing times of the standard agent (s), aggressive agent (a) and careful agent (c) over an increasing number of honeypots (z-axis)

6.2 Honeypots

In the previous section, the impact of honeypots on the different attack strategies was discussed. In this section various honeypot configurations are compared for a single attack strategy. In Figure 4,

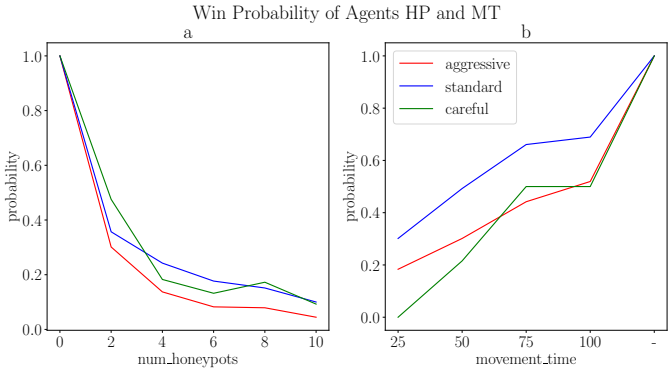


Figure 2: Comparison of the winning probability for the three agents for a) an increasing number of honeypots and b) an increasing IP changing interval.

the impact of 0 to 10 honeypots is compared for different mutation frequencies of 25, 50, 75 and 100 time steps. With a number of 2 honeypots, the winning probability of the agent is already decreased from 100% to below 40% for all mutation intervals. Employing 8 honeypots further decreases the probability to below 20% and 10 honeypots to below 10%. An observation can be made, that with an increasing number of honeypots the mutation interval is having less influence on the winning chance. This also implies that less perturbations are needed the more honeypots are in place. To see the effect of honeypots without host mutation, 5a shows the probability to find and exploit all vulnerable hosts, for 0 to 10 honeypots in a network of 10 and 50 hosts respectively. This show that even in a small network of 10 hosts, the employment of 2 honeypots is enough to decrease the winning probability from 77% down to 34%. As depicted in Figure 5b, a comparison was made between the chance of exploiting all vulnerable hosts and exploiting at least one. While for compromising only one sensitive host the probability is 82% compromising all three hosts is only 41%. For two honeypots the chance for compromising one host decreases to 55% and for three host decreases to 5%.

6.3 Moving Target Defense

For the MTD, the evaluation was made by varying the mutation intervals, as can be seen in Figure 6a, for network sizes of 10 and 50 hosts, respectively. The results show that employing moving target defense every 25 time steps for a network of 10 hosts has the highest chance of winning with 78%, while the a larger network of 50 host increases difficulty to find the vulnerable target and decrease the winning chance to 48%. Increasing the movement time to 50 increases the winning probability to 33% and 23% respectively. For 10 hosts, the mutation interval of 75 is already very close to the reference of having no MTD in place. For larger networks of 50 hosts, the increase of movement intervals does slightly increase the chance of winning, with the highest movement interval still being 33% lower than the reference of no MTD.

For only one goal, as depicted in 6b, the MTD does not decrease the winning chance below 40% for any interval, as shown in Figure 6b. Meanwhile compromising all three sensitive hosts is below 1%

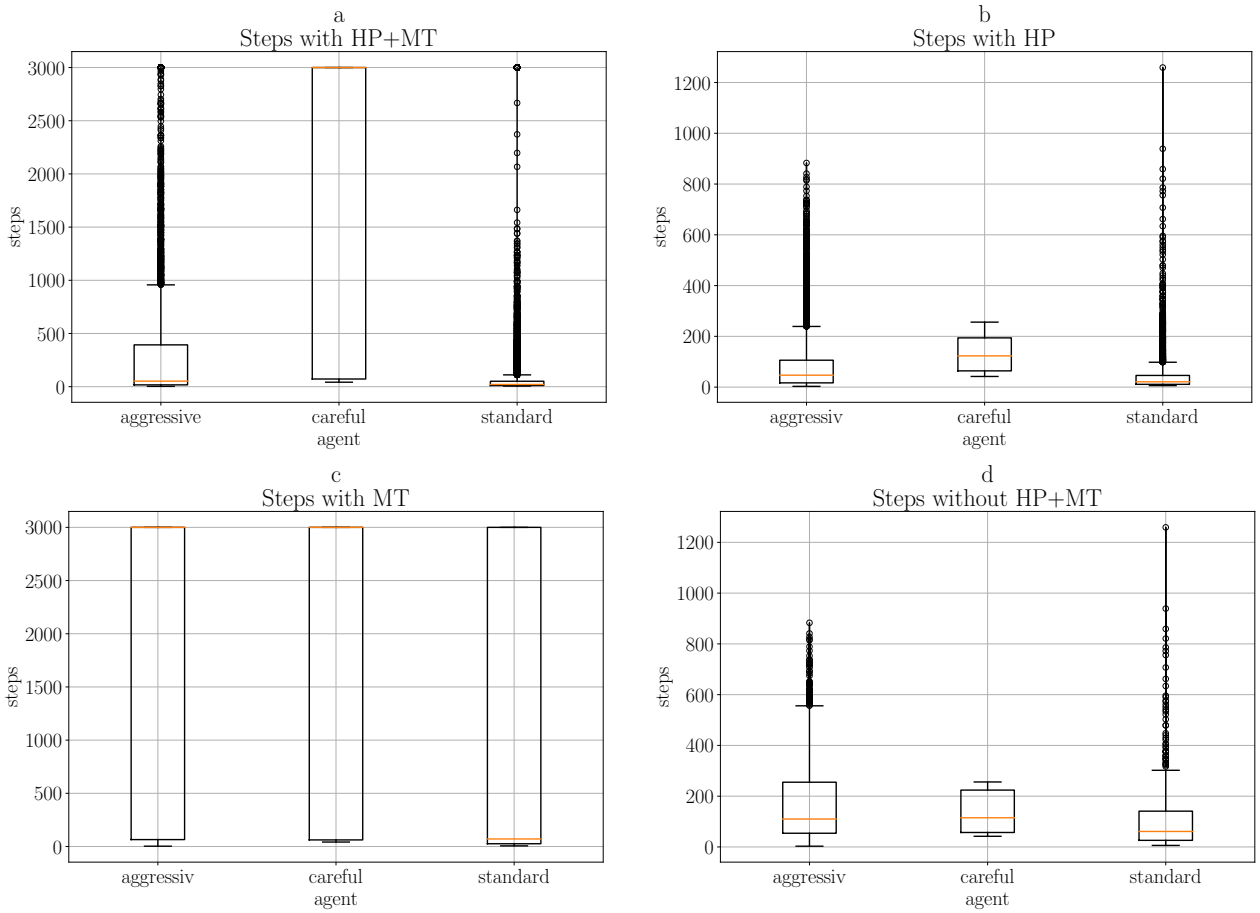


Figure 3: Comparison of steps needed for the three agents, a) MTD and Honeypots, b) Honeypots only, c) MTD only, d) no deception measures.

for the short mutation interval of 25 and at 10% for the longer mutation interval of 100 time steps.

7 RELATED WORK

While much research has been conducted on designing specific types of honeypots, decoys or moving target defense strategies, and the implications on attack detection and slow-down have been discussed, there is no standard strategy for the evaluation of such techniques.

Thus, the evaluation of deception technologies has been attempted in different forms. Many approaches are based on the formalization of the attacker and the defender using game theory [19]. Garg et al. created a framework based on extensive games of imperfect information to determine strategies of the attacker and the honeynet system [10]. Quang Duy et al. modeled the attacker and defender as a Bayesian game of incomplete information, where both the attacker and the defender have deception capabilities, to determine the strategies of both defending on the attack frequency [17]. Hereby the attacker can obfuscate attacks and the defender can incorporate honeypots.

Other works rely on experiments with human subjects acting as attackers, such as Han et al., who evaluated web based decoys resulting in a detection rate of 67% of the 150 participants with a low rate of false positives [11] or Ferguson-Walter et al., who used a psychological approach to evaluate deception by measuring the cognitive load on human penetration testers while interacting with cyber deception [7]. Bensalem et al. considered deception as a defense against masquerade attacks, in which attackers pose as legitimate users, e.g. in identity theft [3]. Their method of deception consisted of monitored honeyfiles acting as a IDS. The evaluation was done by conducting scenario-based user studies, in which computer science students were tasked to play the role of a disgruntled employee, leveraging physical access to a co-workers workstation for illicit financial gain. A special focus of the study were the properties of the honeyfiles. Heckman conducted a cyber-wargame, in which deception was employed along with a certificate-based defense strategy [12]. In this setup, a team of attackers tried to gain access to a command and control system defended by two defending teams, each of which employed one of the mentioned defense strategies, with the certificate-based strategy being the first line of defense, and the deception strategy only being activated once

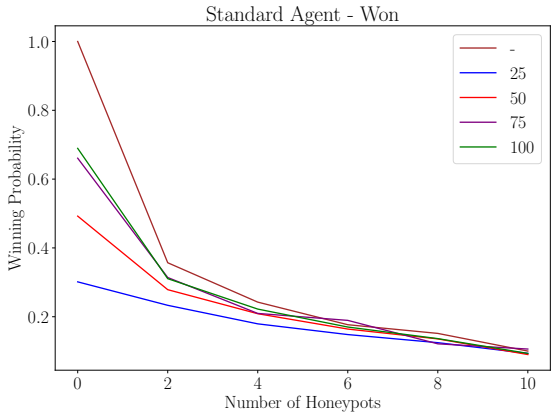
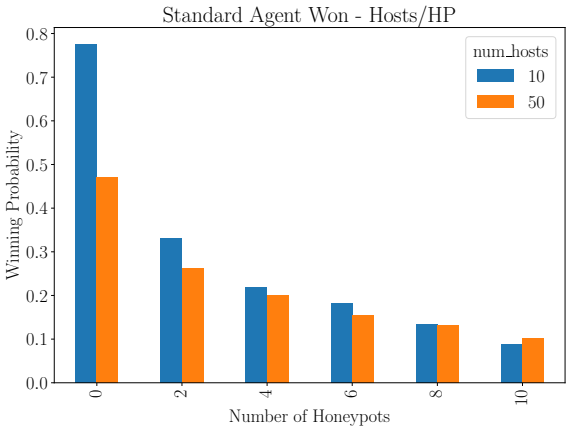
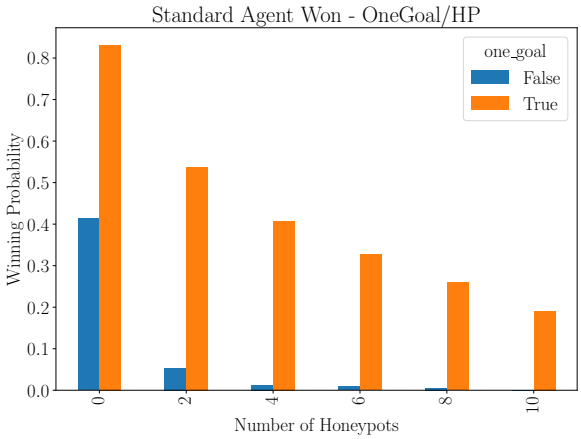


Figure 4: Comparison of the probabilities for the standard agent to win for an increasing number of honeypots. The colors represent different movement times



(a) Comparison of the probabilities for the standard agent to win for an increasing number of honeypots in networks of 10 and of 50 hosts.



(b) Comparison between the probabilities of the standard agent to compromise one host and to compromise all hosts for an increasing number of honeypots, respectively.

Figure 5: Comparison between the winning probabilities of the standard agent for different configurations.

the attackers breached the system, triggering a switch of the target system to a backup instance. The results of this experiment showed that while certificate-based defense could not deny the attacking team access to the target system, the deception-based defense succeeded in leading the attackers to waste their time and resources on a decoy system. Shabtai et al. studied insider data misuse by adding automatically generated honeytokens to collections of sensitive data, such as loan applications [25]. In an experiment with student participants it was studied how well honeytokens pass for authentic data, and how well adversarial behavior can be detected by them. Interestingly, while honeytokens could indeed be shown to serve as efficient sensors for malicious activity, the knowledge of honeytokens being deployed did not influence the participants' behavior, although the authors do conclude that this might be due to a lack of real-life consequences in the experimental setting.

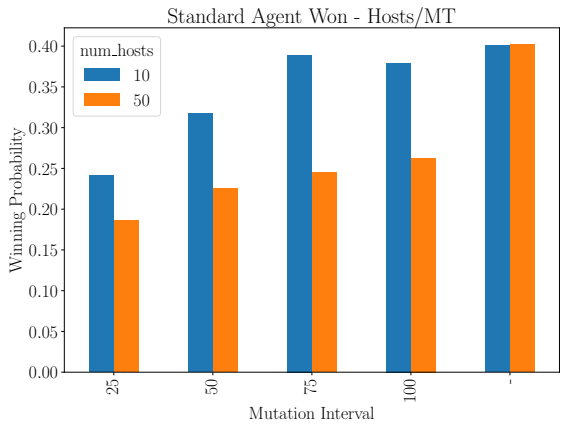
Brewer et al. studied the defense of websites against automated bot attack by implementing a deception service that actively modifies a website's HTML code to include decoy links, which allow for the detection of bot activity. This service was evaluated by deploying three instances of websites protected by the service and then respectively attacking each website with a different type of bot, modeled after such bots as commonly found in the wild. The websites were protected, i.e. not publicly available, thus human volunteers were invited to visit the websites and generate non-malicious traffic, as to allow a distinction between malicious bot activity and legitimate user activity [4].

In a simulation based approach, Ferguson-Walter et al. used the network attack simulator CyberBattleSim, developed by Microsoft, to evaluate the effectiveness of deception against a reinforcement learning (Rl) based penetration testing agent [29]. Hereby the focus lies on incorporating decoys in the simulation instead of honeypots, and measuring the effect of the number of the applied deceptive elements on the attack.

While the simulation environment used in this work is originally also intended for RL based agents, it was chosen to define the attackers with strict strategies to have more comparable and explainable results. The evaluation and recommendation of deception strategies is still unsolved. A simulation methodology combining honeypots and moving target defense, as described in this paper, has not been presented before to the author best knowledge.

8 CONCLUSION

In our work, a simulation based methodology was presented for evaluating the required configuration of honeypots and MTD on a network to achieve certain intrusion detection rates for defined attacker agents, depending on different network parameters that need to be provided, such as network size and an assumption about the amount of sensitive hosts, and the attacker type. Three different attacker types have been defined and considered and can be chosen for the simulation. The simulation, which was implemented



(a) Comparison of the probabilities for the standard agent to win for an increasing IP changing interval in networks of 10 and of 50 hosts.

(b) Comparison between the probabilities of the standard agent to compromise one host and to compromise all hosts for an increasing IP changing interval.

Figure 6: Comparison between the winning probabilities of the standard agent for different configurations.

based on Python and the NASim library, has shown to provide valuable insight on the impact of honeypots and MTD in various attack scenarios. The results can be utilized to determine the optimal trade-off between the number of honeypots or size of mutation intervals in a network and the resulting intrusion detection and prevention capabilities and on the other hand the increased resources required to configure, deploy and maintain the honeypot and MTD implementations. Also a lower number of honeypots saves IPv4 address space which might be reserved or used for other purposes. Although the simulation is based on abstraction and many assumptions have to be taken, the result is of value if these assumptions can be sufficiently justified for the real world scenarios of practitioners. Additionally, researchers may use the proposed simulation methodology or implementation to evaluate their deception methods and tools in dynamic environments. The results of the case studies provided in Section 6 of this paper are only valid for the specific simulation parameters and can not be seen as general metrics, since the parameters in the simulation, such as limiting an attack to 3000 steps, may not accurately describe any real world scenario. They should be rather interpreted in relation within the simulation setting to measure the impact of the attacker types, network size, mutation intervals and honeypots.

ACKNOWLEDGMENTS

This research was supported by the German Federal Ministry of Education and Research (BMBF) through the Open6GHub project (Grant 16KISK003K).

REFERENCES

[1] Ehab Al-Shaer, Qi Duan, and Jafar Haadi Jafarian. 2013. Random Host Mutation for Moving Target Defense. In *Security and Privacy in Communication Networks (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering)*, Angelos D. Keromytis and Roberto Di Pietro (Eds.). Springer, Berlin, Heidelberg, 310–327. https://doi.org/10.1007/978-3-642-36883-7_19

[2] Callum Baillie, Maxwell Standen, Jonathon Schwartz, Michael Docking, David Bowman, and Junae Kim. 2020. CybORG: An Autonomous Cyber Operations Research Gym. https://doi.org/10.48550/ARXIV.2002.10667

[3] Malek Ben Salem and Salvatore Stolfo. 2011. Decoy Document Deployment for Effective Masquerade Attack Detection. (2011), 35–54. https://doi.org/10.7916/D86W9MFJ

[4] Douglas Brewer, Kang Li, Laksmish Ramaswamy, and Calton Pu. 2010. A Link Obfuscation Service to Detect Webbots. In *2010 IEEE International Conference on Services Computing*. IEEE, Miami, FL, USA, 433–440. https://doi.org/10.1109/SCC.2010.89

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540 http://arxiv.org/abs/1606.01540

[6] Augusto Paes de Barros. 2003. DLP and honeytokens. http://blog.securitybalance.com/2007/08/dlp-and-honeytokens.html. Accessed: 2022-07-27.

[7] Kimberly J. Ferguson-Walter, Maxine M. Major, Chelsea K. Johnson, and Daniel H. Muhleman. 2021. Examining the Efficacy of Decoy-based and Psychological Cyber Deception. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 1127–1144. https://www.usenix.org/conference/usenixsecurity21/presentation/ferguson-walter

[8] Daniel Fraunholz, Simon Duque Anton, Christoph Lipps, Daniel Reti, Daniel Krohmer, Frederic Pohl, Matthias Tammen, and Hans Dieter Schotten. 2018. Demystifying Deception Technology:A Survey. https://doi.org/10.48550/ARXIV.1804.06196

[9] Daniel Fraunholz, Daniel Krohmer, Simon Duque Anton, and Hans Dieter Schotten. 2018. Catch me if you can: Dynamic concealment of network entities. In *Proceedings of the 5th ACM Workshop on Moving Target Defense*. 31–39.

[10] Nandan Garg and Daniel Grosu. 2007. Deception in honeynets: A game-theoretic analysis. In *2007 IEEE SMC information assurance and security workshop*. IEEE, 107–113.

[11] Xiao Han, Nizar Kheir, and Davide Balzarotti. 2017. Evaluation of deception-based web attacks detection. In *Proceedings of the 2017 Workshop on Moving Target Defense*. 65–73.

[12] Kristin E. Heckman, Michael J. Walsh, Frank J. Stech, Todd A. O'Boyle, Stephen R. DiCato, and Audra F. Herber. 2013. Active Cyber Defense with Denial and Deception: A Cyber-Wargame Experiment. *Computers & Security* 37 (Sept. 2013), 72–77. https://doi.org/10.1016/j.cose.2013.03.015

[13] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. 2012. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the first workshop on Hot topics in software defined networks*. 127–132.

[14] Ari Juels and Ronald L Rivest. 2013. Honeywords: Making password-cracking detectable. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 145–160.

[15] Panos Kampanakis, Harry Perros, and Tsegereda Beyene. 2014. SDN-based solutions for moving target defense network protection. In *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 1–6.

[16] Li Kechao and Xiong Xinli. 2019. OpenHIP Random Host Hopping in Network Layer. In *International Conference on Education, Management and Information Technology (ICEMIT 2019)*.

[17] Quang Duy La, Tony Q. S. Quek, and Jemin Lee. 2016. A game theoretic model for enabling honeypots in IoT networks. In *2016 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2016.7510833

[18] Vishnu Navda, Aniruddha Bohra, Samrat Ganguly, and Dan Rubenstein. 2007. Using channel hopping to increase 802.11 resilience to jamming attacks. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*. IEEE, 2526–2530.

[19] Radek Píbil, Viliam Lisỳ, Christopher Kiekintveld, Branislav Bošanskỳ, and Michal Pěchouček. 2012. Game theoretic model of strategic honeypot selection in computer networks. In *International conference on decision and game theory for security*. Springer, 201–220.

[20] Niels Provos. 2002. Honeyd is a small daemon that creates virtual hosts on a network. http://www.honeyd.org/. Accessed: 2022-07-12.

[21] Lukas Rist. 2011. Web Application Honeypot. https://github.com/mushorg/glastopf. Accessed: 2022-07-12.

[22] Lukas Rist. 2018. Conpot is a low interactive server side Industrial Control Systems honeypot. http://conpot.org/. Accessed: 2022-07-12.

[23] Neil C Rowe, E John Custy, and Binh T Duong. 2007. Defending cyberspace with fake honeypots. *J. Comput.* 2, 2 (2007), 25–36.

[24] Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Dijiang Huang, and Subbarao Kambhampati. 2020. A Survey of Moving Target Defenses for Network Security. *IEEE Communications Surveys and Tutorials* 22, 3 (July 2020), 1909–1941. https://doi.org/10.1109/COMST.2020.2982955

[25] Asaf Shabtai, Maya Bercovitch, Lior Rokach, Ya'akov Kobi Gal, Yuval Elovici, and Erez Shmueli. 2016. Behavioral Study of Users When Interacting with Active Honeytokens. *ACM Transactions on Information and System Security* 18, 3 (Feb. 2016), 9. https://doi.org/10.1145/2854152

[26] Dilli P. Sharma, Jin-Hee Cho, Terrence J. Moore, Frederica F. Nelson, Hyuk Lim, and Dong Seong Kim. 2019. Random Host and Service Multiplexing for Moving Target Defense in Software-Defined Networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2019.8761496

[27] Dilli Prasad Sharma, Dong Seong Kim, Seunghyun Yoon, Hyuk Lim, Jin-Hee Cho, and Terrence J. Moore. 2018. FRVM: Flexible Random Virtual IP Multiplexing in Software-Defined Networks. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 579–587. https://doi.org/10.1109/TrustCom/BigDataSE.2018.00088

[28] Upi Tamminen. 2009. Kippo is a medium interaction SSH honeypot designed to log brute force attacks. https://github.com/desaster/kippo. Accessed: 2022-07-12.

[29] Erich Walter, Kimberly Ferguson-Walter, and Ahmad Ridley. 2021. Incorporating deception into cyberbattlesim for autonomous defense. *arXiv preprint arXiv:2108.13980* (2021).

[30] Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer. 2004. Honeyfiles: deceptive files for intrusion detection. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.* IEEE, 116–122.