# DOLOS: A Novel Architecture for Moving Target Defense

Giulio Pagnotta, Fabio De Gaspari, Dorjan Hitaj, Mauro Andreolini, *Member, IEEE*, Michele Colajanni, and Luigi V. Mancini

***Abstract*— Moving Target Defense and Cyber Deception emerged in recent years as two key proactive cyber defense approaches, contrasting with the static nature of the traditional reactive cyber defense. The key insight behind these approaches is to impose an asymmetric disadvantage for the attacker by using deception and randomization techniques to create a dynamic attack surface. Moving Target Defense (MTD) typically relies on system randomization and diversification, while Cyber Deception is based on decoy nodes and fake systems to deceive attackers. However, current Moving Target Defense techniques are complex to manage and can introduce high overheads, while Cyber Deception nodes are easily recognized and avoided by adversaries. This paper presents DOLOS, a novel architecture that unifies Cyber Deception and Moving Target Defense approaches. DOLOS is motivated by the insight that deceptive techniques are much more powerful when integrated into production systems rather than deployed alongside them. DOLOS combines typical Moving Target Defense techniques, such as randomization, diversity, and redundancy, with cyber deception and seamlessly integrates them into production systems through multiple layers of isolation. We extensively evaluate DOLOS against a wide range of attackers, ranging from automated malware to professional penetration testers, and show that DOLOS is effective in slowing down attacks and protecting the integrity of production systems. We also provide valuable insights and considerations for the future development of MTD techniques based on our findings.**

***Index Terms*— Moving target defense, active defense, cyber deception, intrusion detection.**

## I. INTRODUCTION

TRADITIONAL cyber defense techniques are typically based on a detection-reaction paradigm where various mechanisms are deployed to identify early signs of intrusion. Upon detection, intruders are generally blocked, and relevant alerts are forwarded to the Computer Security Incident Response Team (CSIRT) for remediation. This *reactive* approach to cybersecurity has considerably evolved over the past decades, and the core concept of detection-reaction is embedded in the large majority of cyber defense tools available today. However, reactive approaches create a disadvantage for defenders because of the asymmetrical relationship between attackers and defenders. Indeed, while reactive cyber defenses are static and come from fixed procedures, they must be effective even against novel and unfamiliar adversaries using constantly changing attack methods and exploiting new vulnerabilities.

Moving Target Defense (MTD) [1] (also known as Active Defense) and Cyber Deception [2] emerged in recent years as proactive cyber defense techniques aimed at protecting computer systems from intrusion. The key concept behind these approaches is to impose an asymmetric disadvantage for the attacker by exploiting randomization and deception techniques, creating a dynamic attack surface that is hard for the attacker to identify and exploit [3], [4], [5], [6].

MTD-based techniques are typically based on randomization, diversification, and redundancy and aim at changing system configuration unpredictably to hamper reconnaissance and exploitation attempts [7], [8], [9], [10]. Cyber Deception techniques generally rely on mock-ups of real systems and services deployed on separate, fake nodes in the network with the goal of misleading attackers and attracting them away from sensitive targets [11], [12], [13].

However, these approaches suffer several limitations. MTD approaches, particularly randomization-based ones, can impose large overheads on the system they are deployed on, are complex to implement, and risk affecting legitimate services [3]. Cyber Deception techniques do not hinder legitimate services and pose no overhead on production systems, as they are typically deployed on separate nodes or machines. However, it has been shown that expert adversaries can easily recognize and bypass them to focus on the actual production systems [14], [15].

This paper presents DOLOS, a new architecture to unify Moving Target Defense and Cyber Deception. DOLOS is designed to seamlessly integrate into production systems to provide deception and MTD capabilities through the orchestration of fake services and randomization of the attack surface. Unlike typical MTD, DOLOS *does not randomize real services*:

Manuscript received 11 February 2023; revised 7 June 2023 and 13 July 2023; accepted 8 September 2023. Date of publication 25 September 2023; date of current version 9 October 2023. This work was supported in part by the Project SERICS under the National Recovery and Resilience Plan by the Minister of University and Research (NRRP MUR) Program funded by the European Union-NextGenerationEU under Grant PE00000014 and in part by the Gen4olive Project funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant 101000427. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chia-Mu Yu. *(Corresponding author: Fabio De Gaspari.)*

Giulio Pagnotta, Fabio De Gaspari, Dorjan Hitaj, and Luigi V. Mancini are with the Dipartimento di Informatica, Sapienza University of Rome, 00185 Roma, Italy (e-mail: pagnotta@di.uniroma1.it; degaspari@di.uniroma1.it; hitaj.d@di.uniroma1.it; mancini@di.uniroma1.it).

Mauro Andreolini is with the Dipartimento di Scienze Fisiche, Informatiche e Matematiche, University of Modena and Reggio Emilia, 41121 Modena, Italy (e-mail: mauro.andreolini@unimore.it).

Michele Colajanni is with the Dipartimento di Informatica - Scienza e Ingegneria, University of Bologna, 40126 Bologna, Italy (e-mail: michele.colajanni@unibo.it).

Digital Object Identifier 10.1109/TIFS.2023.3318964

only deceptive services and system properties are manipulated. This approach allows DOLOS to reduce overhead and management complexity compared to previous MTD techniques. On the other hand, DOLOS integrates directly in production systems through multiple layers of isolation, making any bypass much harder compared to typical Cyber Deception techniques. The concept behind DOLOS is that deceptive techniques are much more powerful when integrated into the production systems rather than deployed alongside them. We show that embedding DOLOS in production systems can lead attackers to completely *disregard* them, mistaking them for fake machines. We evaluate DOLOS against different types of attackers, ranging from automated malware to professional penetration testers, and unequivocally show its effectiveness in thwarting attacks.

In this paper, which builds and expands upon our previous work of [6], we make the following new contributions:

- We present DOLOS, a novel architecture to unify Cyber Deception and MTD. DOLOS integrates deception and randomization capabilities directly into production systems, providing the advantages of both MTD and Cyber Deception techniques, without the drawbacks.
- We provide what is, to the best of our knowledge, the most thorough experimental evaluation of an MTD approach to date. We study the effectiveness of DOLOS against a wide range of different attackers, from automated malware to professional pentesters, both in the real world and virtual networks.
- We show that DOLOS's approach is highly effective in slowing down attackers and protecting the integrity of production systems, even against expert human attackers.
- We make available the code to reproduce our results at https://github.com/pagiux/dolos

In particular, we show that (1) DOLOS can effectively trap automated malware for extended periods of time, hindering reconnaissance and making remediation easier; (2) average human attackers are unable to identify and avoid DOLOS services, and are confused by DOLOS MTD-deception tools; (3) expert human attackers consistently fail to compromise machines protected by DOLOS; (4) DOLOS significantly increases the time-to-compromise when deployed and greatly decreases attack success rate.

## II. THREAT MODEL

Figure 1 illustrates the threat model. We consider the setting of a computer network, such as a company network, that needs to be secured from external and internal attackers. A DOLOS Agent (D in the figure) is deployed on each computer in the network (called *production systems* from here on) and provides them with advanced deception and MTD capabilities. A single DOLOS Controller (C in the figure) manages all the Agents in the network; the communication between Agent and Controller is unicast. Each Agent is managed individually by the Controller and no synchronization among the Agents or between messages from the different Agents is required. The DOLOS Controller is considered trusted, and the communication between the Controller and the Agent is considered tamper-proof and secure, for instance by means of TLS. The DOLOS Agent *is not* considered trusted: it can be compromised by adversaries which can, for instance, use it to send forged logs and alerts to the controller. Similar to other MTD techniques, DOLOS is assumed to be part of a complete set of cyber defense tools to provide defense in depth. DOLOS is designed to slow down and thwart reconnaissance and lateral movement activities from attackers, making it more difficult to find and exploit vulnerable services and applications. DOLOS is not designed to patch critical vulnerabilities and misconfigurations of system applications that an attacker can exploit after obtaining access to the production system.

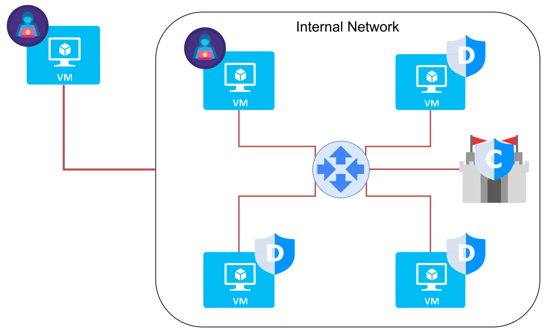

Fig. 1. DOLOS threat model. We consider both internal adversaries which reside inside the network that DOLOS is protecting, as well as external adversaries that can only reach machines on the network perimeter. The DOLOS controller is considered trusted.

We consider two types of adversaries: (1) external attackers and (2) internal attackers. External attackers are located outside the perimeter of the network and can only initiate connections toward production systems that are on the network perimeter, exposed to the Internet. External attackers are therefore limited in their interactions with production systems and can only query exposed services remotely in an attempt to find vulnerabilities and compromise them. Internal attackers reside on a production system within the network and have full control over it. Internal attackers can reach any production system within the network. Both internal and external attackers can scan the network to identify reachable production systems and are allowed to interact with them without limitation. We do not assume IDS or other traditional tools to prevent or limit the activity of the attacker in any way. We assume that legitimate users are well-behaved: they only interact with the production systems through standard applications provided to them with the specific configuration given by the manager of the network, and do not attempt to perform suspicious activities such as network scanning, enumeration or access files on the system that do not belong to them. Therefore, in our model legitimate users never interact with DOLOS Agent's services. Any interaction directed towards any service provided by the DOLOS Agent is considered suspicious and originated by a potential adversary.

## III. DOLOS

This section presents DOLOS, our novel Moving Target Defense architecture. The key idea behind DOLOS is that deception is a much more powerful tool when embedded directly in the systems it should protect rather than deployed alongside them. To this end, DOLOS is designed to seamlessly integrate into production systems to provide advanced deception and annoyance capabilities by orchestrating fake services and randomizing system resources. The goal of DOLOS is two-fold: on the one hand, protecting the production systems by disguising real services among multiple, real-looking fake ones; on the other, slowing down attackers by randomizing the attack surface through fake services and resources, thus allowing ample time for the CSIRT to detect and respond to the threat.

Unlike traditional MTD tools, which directly modify the real services and the configuration of production systems through randomization and diversification techniques [3], DOLOS only alters fake services and system resources that are added by a DOLOS Agent running on the production systems. The real services running on the production system are never randomized or reconfigured, therefore avoiding the management complexities and overhead introduced by traditional MTD techniques. DOLOS can alter fake services and system resources without limitation since they have no impact on the activity of legitimate users, and no additional mechanisms are required to guarantee the consistency and availability of these services. Furthermore, given the direct integration of the fake services and resources into the production system, DOLOS avoids the limitations of traditional cyber deception techniques whereby adversaries can easily identify deceptive systems such as honeypots and avoid them [14], [15]. Since DOLOS fake services are deployed on the same production systems as real services, an attacker will need to thoroughly interact with all of them to assess whether they are real or fake in order to identify potential vulnerabilities, as demonstrated in our experimental evaluation. Finally, DOLOS is designed to transparently integrate with and be deployed alongside other cyber defense tools to provide defense in depth to the production systems. Since DOLOS does not affect real services, it can also be deployed alongside other traditional MTD techniques that employ service randomization and diversification without any conflict.

Overall, the benefits of integrating DOLOS into production are the following:

*Real-Time Protection:* DOLOS's MTD and deception modules transparently provide all production systems in the network with real-time, always-on defense capabilities. As highlighted Section IV-C, DOLOS modules are effective in slowing down and confusing attackers, as well as increasing the amount of interaction with the target system required for adversaries. These properties make it considerably easier to detect intrusions and provide ample time and opportunity for incident and response teams to intervene.

*Operational Continuity:* DOLOS integration ensures minimal disruption to production system operations. As shown in Section IV-I, the production system services remain fully functional and are not affected in any way by DOLOS, which operates only on fake, deceptive services and resources.

*Reduced Management Complexity:* DOLOS modules deploy and interact only with fake services and resources. This reduces the complexity and overhead associated with managing multiple MTD solutions that continuously shuffle and randomize real services. Indeed, as we show in Section IV-I and discuss in Section VI, traditional MTD tools require complex ad-hoc solutions to ensure that real services are not disrupted, increasing the burden on network management and rendering network troubleshooting more difficult.

*Adaptive Defense:* the architecture of DOLOS is designed to allow dynamic deployment and reconfiguration of the modules running in the production systems through the DOLOS Agent (see Section III-A.1). This provides DOLOS with the capability to dynamically adapt deployed modules based on specific attack patterns or actions performed by the adversary against DOLOS modules, making the system highly adaptable to evolving threats. An initial implementation of dynamic adaptation is currently implemented in DOLOS MTD Modules, as discussed in Section IV-H. However, a completely autonomous control logic to detect attack patterns and abstract adversarial goals into concrete actions is left for future work.

*Cost-Effectiveness:* the integration of DOLOS into the production system is more cost-effective compared to the deployment and management of multiple, separate honeydevices in the network. Organizations can leverage existing infrastructure and tools for the management of DOLOS, maximizing their investments while enhancing security capabilities.

The remainder of this section presents the architecture of DOLOS in Section III-A, the MTD tools (called *MTD modules* from here on) implemented in DOLOS in Section III-B, and an overview of the sandboxing and hardening techniques in Section III-C.

## A. Architecture

The DOLOS system consists of two main components: the DOLOS Agent and the DOLOS Controller. The DOLOS Agent is the architectural component deployed on production systems and effectively implements deceptive capabilities. The DOLOS Controller provides command and control functionality for the management of the Agent. In a real-world deployment, an Agent is installed in each production system that needs protection, with a single Controller managing all the agents in the network. In what follows, we describe the DOLOS Agent and Controller and their functionality in detail.

*1) DOLOS Agent:* The DOLOS Agent is implemented as a multi-layered architecture based on a clear separation between different component modules. Figure 2 presents an overview of the architecture. The DOLOS Core component of the Agent is a software module written in C that integrates a Python interpreter to facilitate the prototyping and development of MTD tools. The C component of the core efficiently implements a set of advanced functionalities typically required by MTD modules and exposes an interface to access them. This interface is exposed through the integrated Python interpreter that is extended to interact with the C components of the Core seamlessly. The extension of the interpreter was carefully designed to accommodate the concurrent nature of the DOLOS Core. In particular, the explicit management of the internal reference count of Python objects to periodically clean up memory and the fine-grained handling of the Global Interpreter Lock (GIL) posed non-trivial technical challenges.

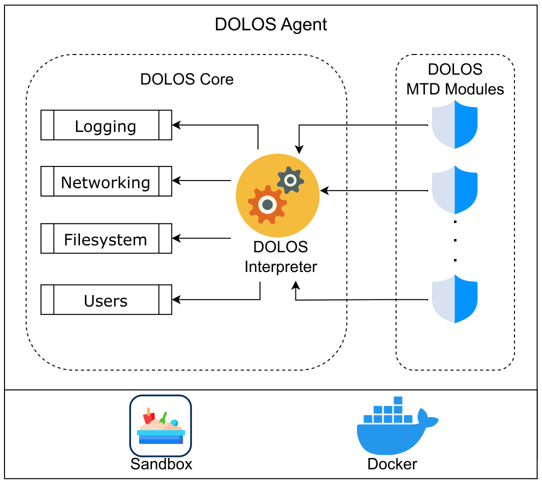

Fig. 2. Architecture of the DOLOS Agent. MTD tools are implemented as modules, executed by an interpreter in the DOLOS Core. The Agent is isolated from the real production system through multiple layers of sandboxing.

The C component of the DOLOS Core implements standardized functions for logging, networking, filesystem, and user-handling-related operations. In particular, the logging component exposes an interface to standardize the generation, storage, and management of logs across all the MTD modules, as well as the functionalities to send the event logs to the controller. The networking component provides methods to seamlessly send and receive network connections concurrently, relying on an implementation of a thread pool within the Core so that MTD modules can be developed without consideration for these complexities. The filesystem component provides a system-agnostic abstraction layer that makes the integration of MTD modules that operate on files easier. It provides interfaces for file creation, integrity checking, and deployment of monitors to detect file and directory modifications. Finally, the user-handling component exposes common methods for the creation and handling of user profiles, such as the setup of user directory trees and of various profile properties.

We highlight that the DOLOS Core component can be easily extended to integrate additional functionalities without any particular limitation, except for the boundaries imposed by the isolation of the DOLOS Agent and the underlying production system. Furthermore, as a result of the integration of the Python interpreter, a Python interface to these functionalities can be easily provided to MTD module developers.

MTD modules are implemented as external components and are executed by the DOLOS interpreter. This modular design of the Agent allows for easy extensibility and avoids duplication of functionality between different MTD modules. The Agent is separated from the underlying production system by multiple layers of isolation through container-based techniques and sandboxing. The sandboxing layer is carefully designed to limit the Agent's access only to the necessary functionality of the underlying system and to reduce the overhead for the production system. As our experimental analysis in Section IV-I shows, our design allows DOLOS to have negligible impact on system performance. We provide more details on the isolation techniques used in Section III-C and a detailed quantitative analysis of the performance overhead in Section IV-I.

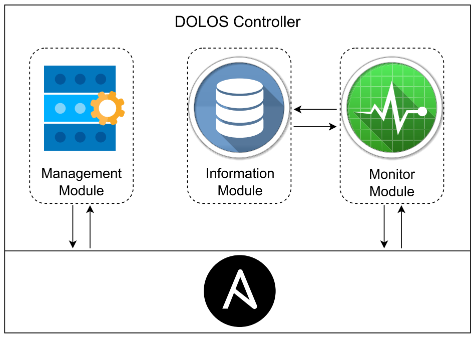

Fig. 3. Architecture of the DOLOS Controller.

The main advantages of the DOLOS Agent over previous container-based solutions are its modularity, extensibility, and efficiency. Typically, a basic container is used to run multiple independent MTD tools simultaneously. The container-based approach was used mainly for ease of deployment and to provide some level of isolation from the underlying system [6]. On the other hand, the DOLOS Agent implements standard functions in the Core component and exposes them to the MTD modules through a unified interface. This modular approach reduces overall resource utilization, as different modules do not need to instantiate the same resource multiple times. The centralization of common functions in the Core component also allows for fine-grained control over the interactions between the Agent and the underlying system, which occur only through the Core libraries. Finally, new MTD modules can be integrated into the Agent in a much easier manner than in typical container-based approaches.

*2) DOLOS Controller:* The architecture of the DOLOS Controller is presented in Figure 3. The Controller is the key component of DOLOS that provides management, monitoring, and control functions. It comprises three main modules: Management, Information, and Monitor. The Management module offers a remote management interface for the Controller and, indirectly, for the Agents. All control actions such as installing Agents, deploying new MTD modules, or restarting an Agent are executed by this component and are carried out through an interface layer implemented through the Ansible orchestrator [16]. The Information module maintains status updates, processed logs from MTD modules, and alerts generated by the Agent. The storage of logs generated by MTD modules is currently implemented in the Structured Threat Information Expression (STIX) format [17], thus allowing easier processing of security events and implementation of automated decision-making. Finally, the Monitor module is a correlation engine that regularly processes data in the Information module and generates alerts on potential threats detected. In the present version, response and remediation actions are not automated by the Controller and must be undertaken by the Incident Response Team. We plan to introduce autonomous learning capabilities in the Controller in future work, as detailed in Section V.

### B. DOLOS *MTD Modules*

This section describes the current capabilities of DOLOS and the MTD modules currently implemented by the

Agent. We subdivide DOLOS MTD modules into three categories: networking-related, filesystem-related, and user account-related.

*1) Networking Modules:* The Networking modules provide a set of essential functionalities to spoof well-known services and fake open ports. The main goal of these modules is to make it hard for an adversary to find real services and to slow down the reconnaissance phase of the attack significantly. Currently, DOLOS implements four networking MTD modules:

- Portspoof, a module that fakes the presence of arbitrary services behind a set of selected open ports. It emulates valid services by dynamically generating signatures to respond to service probes.
- Honeyports, a module that attaches and listens on a set of predefined ports, completing any incoming connection requests. Any IP address that initiates and completes a connection with Honeyports is added to a blocklist.
- Invisiport, a module that implements deceptive blocklisting of incoming connections. It listens for probes on a set of fake, open ports and adds the initiator to an internal blocklist when a connection is detected. All connections from blocklisted initiators are refused on all ports, except for a second set of fake Invisiport services that will still be reachable.
- Endlessh, a module that implements an SSH tarpit to slow down bruteforce attacks. At each connection request, before sending its SSH identification string, it slowly sends a series of banners keeping the bruteforcing script locked up for a long time.
- Artillery, a module that implements SSH bruteforce monitoring. It analyzes login attempts on any SSH shells it is configured to listen on, including potentially fake DOLOS shells, and reports suspicious chains of failed login attempts.

*2) Filesystem Modules:* The Filesystem modules implement file-based deceptive functionalities. The main goal of these modules is to provide early detection of local access breaches and the ability to terminate access to compromised user accounts. Currently, DOLOS implements the following filesystem MTD modules:

- Honeyfiles, a module that monitors file and directory access in the production system. It attaches a listener to a set of specified files and directories and monitors whether unexpected actions are performed. It can also apply different countermeasures, such as terminating the offending process or locking out users.
- Cryptolocked, a module that deploys fake files in the production system called trip files. It attaches a listener to all deployed trip files and triggers an alert whenever access is detected.

*3) User Account Modules:* The User Account modules implement MTD functionality related to fake user accounts. The main goal of these modules is to provide the attacker with a believable environment to interact with in order to slow him down. Currently, DOLOS implements one user account MTD module:

- Honey Account, a module that deploys fake, realistic user accounts in the system. Each account is initialized with a semi-randomized directory structure and a set of files. Moreover, a high-interaction faux shell is attached to each honey account to provide realistic interactions with adversaries. The honey accounts are accessible through a regular SSH server.

### C. DOLOS *Hardening*

Introducing additional services and open ports in production systems, even when fake, inevitably increases the attack surface of the systems. Potentially, an attacker could compromise a DOLOS MTD module to obtain local access to the system and escalate privileges to fully compromise the machine. DOLOS employs a multi-layer sandbox architecture to limit this possibility and reduces the privileges of the Agent to those strictly necessary. The first layer of isolation used in the DOLOS Agent is based on container technology [18]. DOLOS processes are isolated inside a container from normal production system processes and cannot see nor interact with them in any manner. DOLOS Agents also have their own network stack, and interaction with the network interfaces of the production system is allowed only through typical network connections. This prevents the DOLOS Agent from sniffing traffic directed to the production system. Beyond the isolation provided by the container, DOLOS deploys additional tools to limit the Agent's access to the underlying system. Access to the production system kernel is filtered through an allowlist, authorizing execution only of specific system calls. The interaction between the Agent and the production system is further limited through mandatory access control (MAC) [19]. DOLOS uses strict MAC configuration policies that prevent the Agent from accessing any object it does not strictly need. Finally, DOLOS also uses fine-grained capabilities configuration to limit container permissions only to those effectively required by the system, which is mainly the use of raw sockets [20]. To use the DOLOS Agent as an entry point in the system, an attacker would have to: (1) compromise a DOLOS MTD module; (2) escalate privileges within the Agent's container; (3) escape the container through a vulnerability; (4) achieve all this while being limited by DOLOS's system call allowlist, MAC and capability limitation system.

Finally, an attacker that can compromise a DOLOS Agent but is unable to escape the isolation to the underlying system can only use the Agent to communicate with the Controller. However, this is inconsequential, as the Agent does not have the ability to send any commands or carry out any actions for the attacker. The attacker can either prevent local alerts and log forwarding to the Controller or send fake logs/alerts. Preventing log forwarding after compromising the Agent is ineffective because the attacker has already interacted with the MTD modules to gain access, and all relevant logs and alerts have already been sent to the Controller. Meanwhile, sending forged logs or alerts would serve no real purpose, as it may only risk triggering an alert from the Controller to the CSIRT. These alerts can be further validated by correlating data from the DOLOS Agent with those from other traditional defense systems in the network.

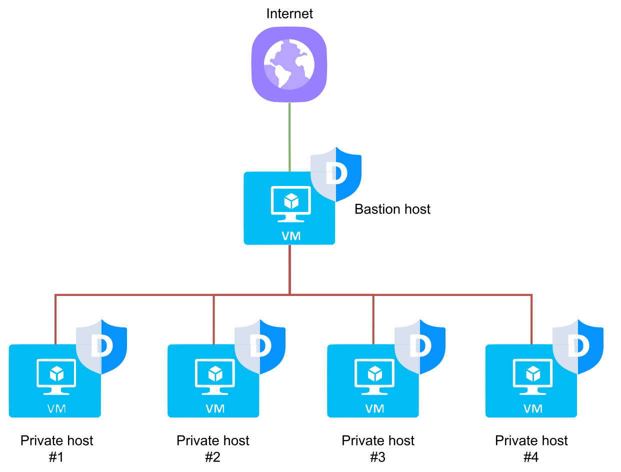

Fig. 4. Experimental setup used in Setting (I) of the Automated Adversary evaluation.

## IV. Evaluation

This section evaluates DOLOS across a wide range of conditions and against four different types of adversaries: automated malware, average human attackers, expert human attackers, and professional human attackers. Finally, we study the overhead introduced by the DOLOS framework on the production systems.

### A. Automated Adversary: Malware

This section evaluates the ability of DOLOS to slow down automated attacks by malware. We design two different settings to analyze malware interactions with our MTD tools: (I) focusing on interactions at the filesystem level, representing the case of malware running on a local machine in the network, for instance, after a successful phishing campaign; and (II) focusing on interactions at the network level, representing the case of an external attacker trying to get a foothold in the network. With these experiments, we aim to answer the following research questions: (a) are file system MTD tools practical to slow down automated malware? (b) are networking MTD tools effective in thwarting and detecting malware lateral movement? (c) can networking MTD tools slow down reconnaissance and local access attempts by malware?

*1) Experimental Setup - Setting (I):* Figure 4 illustrates the experimental setup. We deploy five Debian VMs on the open Internet through the Google Cloud Platform Compute Engine (GCP). One of these machines acts as a bastion host, providing connectivity to the subnetwork and preventing direct connections from the Internet to the remaining four private hosts. On the bastion host, we deploy three separate docker containers:

- DOLOS container: runs the Cryptolocked and Honeyfiles modules to deploy trip-files throughout the filesystem and to monitor file events in the */home, /data* and */tmp* directories of the SSH users and Apache container.
- Apache container: runs an HTTP web server exposed on port 80 comprised of a single CGI webpage. The page allows users to interact with the underlying systems and send commands to a bash shell vulnerable to the Shellshock vulnerability [21].
- Ubuntu container: runs an SSH service on the default port 22, exposing two accounts with extremely weak username-password pairs.

On the four private hosts, we deploy DOLOS with Honeyports that expose fake services on well-known open ports to detect any potential malware traffic generated from the bastion host. This configuration of DOLOS allows us to evaluate both the effectiveness of file-based deception against automated malware and the usefulness of network-level MTD against lateral movements by automated adversaries.

*2) Results - Setting (I):* We have deployed our virtual network on the open Internet for a period of 30 days, over which we analyzed all interactions with the containers in the bastion host and private hosts. Over this period, the bastion host was compromised within a few hours of deployment a total of 14 times, with a maximum deployment-to-compromise time of 27 hours. After each compromise, the bastion host was manually terminated and redeployed. All security breaches happened through the exploitation of the weak credentials on the SSH service, while no successful attacks were detected on the HTTP Web server.

Once local access to the bastion host was obtained, none of the malware interacted with any trip file or performed any operation in the directories monitored by DOLOS. Furthermore, no outbound connection to any of the private hosts was detected, and in general, no outbound traffic to the internal network was generated. In all breaches, the automated malware installed cryptominers in the bastion host, which explains why no interaction with the local file system nor private hosts was detected by DOLOS. Since the GCP forbids crypto mining, we shut down the bastion host and restored it soon after each security breach. This can also explain the lack of interaction registered with the malware. It is possible that, given more time, the malware would have attempted to spread toward other hosts in the network to increase the mining capacity.

*3) Experimental Setup - Setting (II):* The experimental setup for setting (II) consists of a single VM where we deploy DOLOS equipped with two tools: Endlessh and Honeyports. Endlessh is configured to run on the standard port 22 in “start” mode, while we configure Honeyports to simulate several open services on well-known ports: FTP on 21, DNS on 53, HTTP server on 80, IMAP on 143 and MySQL on 3306. The VM is deployed on the GCP and directly accessible on the Internet.

*4) Results - Setting (II):* We have deployed the VM for a total of 430 hours, distributed over a period of one month. In total, five instances of the VM were deployed. Over this period, Honeyports detected a large number of incoming connections (port scans) evenly distributed among the different deployed services. More rarely, complete Nmap fingerprinting connections were detected as well — also evenly distributed. Interestingly, the most significant portion of incoming connections did not target the Honeyports services, but concentrated on the ssh port: 1758 connections were detected by Endlessh over the 430 hours, with duration ranging from a few seconds to over 9 hours. This indicates a strong preference by the malware to target interactive services that can provide immediate access to the system, possibly because credential bruteforcing is an easily-automated process. Figure 5 shows the CDF of

TABLE I
FILESYSTEM-LEVEL ACTIVITY DETECTED THROUGH LOG ANALYSIS AFTER COMPROMISE OF BASTION HOST

| Botnet | Type | Prevalence | Filesystem Activities |
|---|---|---|---|
| Unidentified #1 | crypto miner masquerading as dhcp daemon | 7/14 VMs | • creation of ~/.dhpcd executable with XMRig cryptominer. • creation of ~/.ssh/authorized_keys with fixed key. • creation of ~/.core file containing bytecode. • crontab launching dhpcd every 60 seconds. |
| Muhstik | botnet - cryptominer | 2/14 VMs | • wget downloads of multiple files to home, /tmp. • renaming of several files and download of shell scripts. |
| Outlaw | botnet - cryptominer | 2/14 VMs | • creation of ~/.configrc and subdirectories “a” and “b”. • unpacking of XMRig cryptominer in “a”. |
| Unidentified #2 | cryptominer | 1/14 VMs | • creation of multiple files in /tmp directory. • all created files are deleted shortly after. |
| Unidentified #3 | cryptominer | 1/14 VMs | • wget download of script file in /home. • script file deleted shortly after. |

the inbound ssh connection duration registered during the evaluation. As we can see, most malware connections last for less than $100s$, with only a few long-lasting connections that skew the average. Most likely, the short-duration connections result from scanning/enumeration probes that give up after a short time, while longer-lasting ones correspond to real connections to attempt credentials bruteforcing.

## B. Lessons Learned: Automated Adversary

*1) Lessons Learned - Setting (I):* While it is unfortunate that only cryptominers interacted with our systems in the 30 days deployment window, the results of this experiment provide some interesting insights. Concerning research question (a), we can say that DOLOS's file-level MTD modules are not helpful against cryptominers. This is unsurprising, as the only goal of this type of malware is to obtain as much computational power as possible to mine cryptocurrency. Indeed, all cryptominers in our experiments only downloaded scripts in the home directory, which is not considered suspicious activity. The answer to research question (b) is also negative, as cryptominers do not appear to attempt any lateral movement once they obtain access to a machine. This result is somewhat surprising, as one would expect malware to try to infect as many devices as possible to further increase the available computational power. However, as we stated before, this result could be influenced by the short window of time the malware had to propagate to new machines before the bastion host was reset. Finally, our analysis of the filesystem-level activity performed by the cryptominers provides interesting hints on what new active defense tools can be developed against this type of malware. Table I details the activity carried out by the different cryptominers obtained through log analysis. Most of the activity carried out by the malware relates to downloading malicious scripts or unpacking malware in the */home* and */tmp* directories. Furthermore, at least one botnet exploits known services and cronjobs to ensure that the malware is running at all times. These interactions are challenging to deal with since introducing filesystem-level tools that detect the creation/modification of files in the */home* and */tmp* directories could hinder regular use of the system. However, almost all the malware detected also downloads additional scripts/bytecode beside the cryptominer to perform further actions. An interesting direction to deal with this kind of malware is to analyze the behavior these scripts enable and target these new actions with specific new deception MTB modules. Due to the terms of use of the GCP, we could not analyze the secondary behavior of these malware, as the bastion VM was terminated as soon as we detected crypto activity. The detailed analysis of the secondary behaviors of crypto miners and possible MTD countermeasures are left as future work.

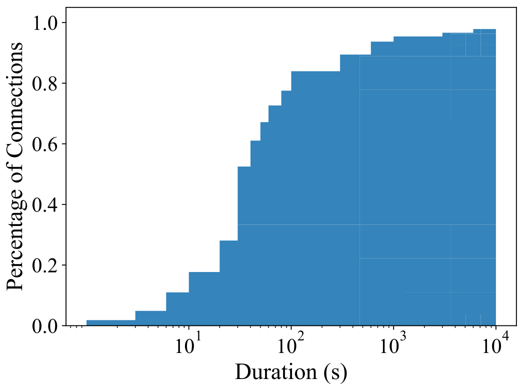

Fig. 5. Cumulative Distribution Function (CDF) of Endlessh connections duration for the Setting (II) of the Automated Adversary evaluation.

*2) Lessons Learned - Setting (II):* The results of this experiment show that DOLOS's networking MTD modules are highly effective in hindering malware during the reconnaissance phase, forcing it to waste time and preventing it from being able to quickly probe all available services. Furthermore, the large amount of network noise generated makes it much easier to detect automated exploit attempts. The results of the experiments also suggest that malware often exploits easily automatable vulnerabilities, such as weak credentials. By using fake interactive services, such as Endlessh, to trap the malware in endless connection loops, real services can be protected from attacks. Therefore, our experimental results suggest a positive answer to the research question (c).

## C. Average Human Attacker

This section evaluates the effectiveness of DOLOS against average human attackers. For these experiments, we enrolled a team of students from the final year of our computer science degree and asked them to attack a VM protected by DOLOS. The attackers aim to obtain local access to the VM and perform privilege escalation to steal sensitive documents. Similarly to our previous experiment, we design this evaluation to analyze the effectiveness of different DOLOS MTD modules in deceiving and hindering knowledgeable adversaries. This experiment aims at providing a qualitative evaluation of DOLOS and answering three research questions: (d) how effective are DOLOS's networking MTD modules against human adversaries? (e) Can networking MTD tools effectively slow down and confuse human attackers, providing time for incident response teams to act? (f) Are filesystem MTD tools practical against privilege escalation and data stealing?

*1) Experimental Setup:* The experimental setup consists in one VM where we deploy real, vulnerable services as well as DOLOS services. Attackers can interact with the VM remotely through the network. The detailed configuration of the VM is as follows:

*a) Real services:*

- Apache HTTP server on port 80. The server has a single webpage using CGI to handle user requests. The page shows a website under construction that internally calls a bash script to retrieve connection-related information to display. The bash version has been downgraded to be vulnerable to the Shellshock attack. The webserver runs under the default "www-data" user.
- SSH service on port 2000. This service is vulnerable to bruteforce attacks on two different user accounts using well-known weak passwords: "test" with password "test"; "user" with password "password123". These users do not have root privileges.

*b)* DOLOS *services:*

- Portspoof. Provides several fake services to slow down reconnaissance and make it harder to identify real services.
- Invisiport. Setup to trigger upon interaction with ports 53, 443, 445, corresponding to DNS service, HTTPS server, and Microsoft-ds (SMB - Server Message Block protocol).
- Endlessh. Provides fake ssh tarpits on ports 22, 2001, and 2222.
- Honeyfile. Configured to log interactions with a set of directories that could be of interest to an attacker.
- Honeyfile. Configured to kill the PID and lock the user that accesses a set of directories hidden in the /home and /data directories.
- Cryptolocked. Configured to kill the PID and lock the user interacting with any trip file deployed in the machine's filesystem.

The pentesting activity was carried out in four separate sessions for a total of approximately 12 hours. The participants were not told about the presence of DOLOS on the VM.

*2) Results:* During the initial phase of the attack, the pentesters carried out a port scan on the VM, detecting multiple open services. Initially, the pentesters focused mainly on port 445, the Server Message Block protocol, looking for misconfigurations that could potentially allow access to files or command execution. The pentesters also probed the service to enumerate potential system users. In reality, port 445 is a spoofed service provided by the Invisiport module, which automatically blocklisted the IP address of the pentesters, making all services unreachable except for a small set of fake services provided by the module itself. After further interaction, the pentesters noticed the inconsistencies caused by Invisiport and reset their session to restart the vulnerability assessment from scratch. In this second session, the pentesters focused mainly on the fake Samba and HTTPS services, primarily looking for ways to enumerate users and well-known web application vulnerabilities without success. During the third session, the pentesters switched focus to interactive services, particularly SSH. All four SSH services, one real and three fake SSH tarpits, saw considerable volumes of connections in this session. The fake SSH tarpit on port 22 absorbed the most significant number of connections, shifting the attention of the pentesters away from the real SSH service. Over the session, seven separate connections were directed to port 22, with an average duration of 30 seconds and a maximum duration of 115 seconds. Most of these were short-lived bruteforce attempts on the tarpit, which resulted in considerable wasted time. A small number of connections were also directed to the real SSH service and the other two SSH tarpits. However, no successful bruteforce attempt was registered.

In the fourth and final session of the test, we gave the pentesters local access to the machine by directing them to the real SSH service and suggesting bruteforcing with a list of well-known usernames/passwords. After obtaining local access, the pentesters began analyzing the filesystem looking for potentially valuable files and ways to escalate privileges. In their activity, the pentesters triggered some trip files, which resulted in Cryptolocked terminating the connection. The pentesters also used different scripts and tools to aid their search for potential vulnerabilities and valuable files, which resulted in numerous alerts from the deployed filesystem-based active defense tools. Finally, the pentesters identified a vulnerability in a Vim container, which allowed them to spawn a root shell and escalate privileges.

## *D. Lessons Learned: Average Human Attacker*

From the logs gathered during the four pentesting sessions and a descriptive report produced by the pentesters, we can confidently say that the answer to research questions (d) and (e) is affirmative: DOLOS networking MTD modules are highly effective at confusing and slowing down average human attackers. At the end of the third session, the testers reported that *"the conflicting information gathered by initial assessments made it hard to identify attack vectors and plan exploitation strategies"* and that *"active services that at times responded slowly and inconsistently complicated the pentesting process"*. Indeed, the Invisiport tool and SSH tarpits proved extremely valuable from the point of view of deception and annoyance, attracting many connections and absorbing a considerable amount of the pentester's time and attention.

Based on the logs and alerts generated from the filesystem-level active defense tools, the answer to research question (f) is also affirmative: DOLOS's filesystem MTD modules are a

TABLE II
VIRTUAL MACHINE CONFIGURATION FOR THE EXPERT HUMAN ATTACKER EVALUATION

| Layer | VM | Service - Port | Vulnerabilities | DOLOS Services - Port |
|---|---|---|---|---|
| Layer 1 | VM1.1 | • Bind DNS - 53<br>• HTTP server - 80<br>• Samba - 139, 445 | • SSH - 22 - remote user enumeration<br>• FTP - 2121 - weak credentials | • No DOLOS |
| | VM1.2 | • FTP - 21<br>• Bind DNS - 53<br>• Samba - 139, 445<br>• Apache Tomcat - 80 | • SSH - 22 - remote user enumeration<br>• HTTP server - 8080 - command injection | • No DOLOS |
| Layer 2 | VM2.1 | • SSH - 22<br>• Bind DNS - 53<br>• Rpcbind - 111<br>• MySQL - 3306 | • HTTP server - 80 - command injection<br>• Asterisk VoIP - 5038 - PBX extension enum. | • No DOLOS |
| | VM2.2 | • Bind DNS - 53 | • SSH - 22 - remote user enumeration<br>• HTTP server - 3241 - command injection | • Portspoof - multiple ports<br>• Endlessh - 2000, 2021<br>• Honeyfile |
| Layer 3 | VM3.1 | • Bind DNS - 53<br>• HTTP server - 80<br>• Samba - 139, 445 | • SSH - 2002 - remote user enumeration<br>• FTP - 2321 - weak credentials | • Portspoof - multiple ports<br>• Endlessh - 22<br>• Honeyfile |
| | VM3.2 | • FTP - 21<br>• Bind DNS - 53<br>• Samba - 139, 445<br>• Apache Tomcat - 8080 | • SSH - 2032 - remote user enumeration<br>• HTTP server - 3241 - command injection | • Portspoof - multiple ports<br>• Endlessh - 22, 2000<br>• Honeyfile |

valuable asset for the early detection of privilege escalation attempts and data stealing. Trip files proved exceptionally effective against scripts aimed at gathering intelligence on the filesystem and system configuration-based vulnerabilities. The scripts used by the pentesters to detect privilege escalation vulnerabilities triggered multiple trip files and generated several log alerts that, in an entire enterprise network, would have immediately warned the CSIRT of the attack.

Concluding, this evaluation shows that the coordination of multiple active defense tools by DOLOS provides valuable defense in depth against average attackers. Moreover, it can significantly increase the time required for an attack thus amplifying the response time available to cybersecurity teams.

## *E. Expert Human Attacker*

This section studies the effectiveness of DOLOS against expert human attackers. We enrolled 31 final-year students with a solid background in systems security from our Master's degree program in Cybersecurity, and designed a multi-layer evaluation of our system. The goal of the attackers is to complete all layers of the experiment by compromising at least one VM in each layer and obtaining root privileges. The attackers had knowledge of the details of DOLOS, but were not explicitly told of its presence on the VMs. We designed this evaluation to assess the effectiveness of DOLOS as a deterrent and measure how much it can slow down attackers. To this end, layers one and three of the experiment are designed to be a one-to-one comparison between an attack on machines without DOLOS and the same machines with DOLOS. Layer two assesses the ability of DOLOS to discourage attacks on systems by comparing a DOLOS-protected VM to the same VM without DOLOS-protection. This evaluation aims at answering two research questions: (g) is DOLOS an effective deterrent? How much less likely to be compromised is a system running it? And (h) can DOLOS provide additional

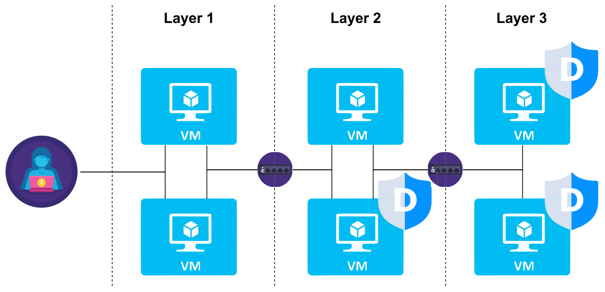

Fig. 6. Experimental setup for the Expert Human Attacker evaluation.

time for incident response teams to react compared to systems without it? If so, how much?

*1) Experimental Setup:* The experimental setup used is illustrated in Figure 6. We use a three-layer design for this experiment where the attacker must compromise a VM in one layer before accessing the next one. Each layer comprises two VMs. To unequivocally identify the VMs, we use the notation *VMx.y*, where x identifies which layer the VM belongs to, and y identifies the VM number in that layer. For example, the second VM of the third layer would be VM3.2. The configuration of the VMs in each layer is presented in Table II. In the first layer, no machine uses DOLOS, and both have reasonably straightforward vulnerabilities. In layer two, VM2.2 is protected by DOLOS and has an easy-to-exploit command injection vulnerability on the webserver, similar to VM1.2. VM2.1, on the other hand, is not protected by DOLOS but has a more complex exploitation path which requires successful PBX extension enumeration by exploiting the Asterisk VoIP service and then using one of these extensions to exploit a command injection vulnerability on a FreePBX web page [22]. Finally, the third layer is configured with the same vulnerabilities as layer 1, but both machines are protected by DOLOS.

TABLE III
EXPERT HUMAN ATTACKER EVALUATION RESULTS. SHOWS THE PERCENTAGE OF PENTESTERS WHO SUCCESSFULLY COMPROMISED EACH VM, THE AVERAGE TIME TO COMPROMISE, AND THE OVERALL TRAFFIC GENERATED

| Layer | VM | Compromise % | Time to Compromise | Traffic (GB) |
|---|---|---|---|---|
| Layer 1 | VM1.1 | 5.34% | 2h 1m 21s | 7.43 |
| | VM1.2 | 84.98% | 1h 9m 36s | 13.79 |
| Layer 2 | VM2.1 | 61% | 14h 37m 20s | 52.06 |
| | VM2.2 (DOLOS) | 0% | N.A. | 26.82 |
| Layer 3 | VM3.1 (DOLOS) | 0% | N.A. | 23.60 |
| | VM3.2 (DOLOS) | 6% | 48h 25m 42s | 31.29 |

*2) Results:* The results of our evaluation are presented in Table III. Overall, the pentesting activity lasted for over 57 hours over a period of one month and generated more than 150GB of traffic. More than 90% of the attackers successfully compromised layer 1, with a significant preference for VM1.2 (84.98% compromise rate). This VM was vulnerable to a reasonably direct command injection vulnerability, which could be found using the OWASP ZAP tool [23]. VM1.1 saw considerably less interaction, almost half as much as VM1.2, and generally was ignored in favor of the other VM. From the penetration testing reports, no particular difficulties were reported on the first layer. This layer of the evaluation was designed to establish a baseline level of proficiency of the attackers and a baseline performance to compare DOLOS's effectiveness.

The second layer was successfully compromised by a considerably lower number of attackers (61%), and *all of them* achieved it by exploiting VM2.1, which did not run DOLOS. As discussed, the exploitation path required to compromise this VM was much more complex than VM2.2, which had the same vulnerable web application as VM1.2. The fact that 85% of the attackers compromised VM1.2, but *none of them* could exploit the same vulnerability in a VM with DOLOS (VM2.2) unequivocally shows its effectiveness as a deterrent. We also note that many attackers reported suspicion that VM2.2 was a honeypot rather than a real system. These reports highlight an important advantage of embedding deception into production systems. This second layer demonstrates that the knowledge of the possible presence of deceptive tools in a system does not adversely impact the effectiveness of the deception. Indeed, it becomes an advantage for the defenders, as the attacker is led to exclude the easier VM in favor of the harder one, due to the fear that it might be a honeypot designed to deceive him. This behavior is consistent with recent findings in the literature on the effects of cyber deception on the behavior of attackers, which finds that "The ambiguity effect suggests that ambiguity causes people to be unwilling to act" and that being informed about the possible presence of deception can increase the confusion in the attacker [11], lending further credibility to our experimental results.

The final layer of the evaluation consists of just DOLOS-protected systems. The VMs in the third layer were configured precisely like the VMs of layer 1, with only port numbers varying. From Table III, we see that only 6% of the attackers successfully compromised this layer, compared to 90% in layer 1. Moreover, the time-to-compromise for layer 3 was over 48 hours, compared to just $\sim$ 1.5 hours for layer 1: a 32-fold increase. The amount of generated traffic also dramatically increased between layer 1 and layer 3, more than doubling from 21.22GB in layer 1 to 54.89GB in layer 3. These results highlight the effectiveness of DOLOS in defending production systems and significantly slowing down attacks, giving CSIRT ample time to take some actions. Furthermore, the considerable increase in registered network traffic and the large number of alerts generated by DOLOS (on average, 63 per attacker) eases the detection of intruders considerably.

## F. Lessons Learned: Expert Human Attacker

From the results of this evaluation, we can draw several conclusions. We can claim that the answer to the research question (g) is affirmative. As shown by the layer 2 evaluation, DOLOS is an effective deterrent, and machines protected by our system are unlikely to be targets of attacks. None of the expert attackers successfully compromised VM2.2, regardless of the easier exploitation path. Instead, after thoroughly interacting with it and generating over 26.82GB of traffic, they decided to avoid it, considering it a potential honeypot.

The answer to the research question (h) is also affirmative. The direct comparison between layers 1 and 3 shows that DOLOS deception capabilities significantly increase system defenses and that, even when successful, expert human attackers require much more time and traffic to compromise the systems. The same VM configuration (VM1.2 - VM3.2) was much more challenging to exploit when DOLOS was deployed because it decreased the success rate by 93% and increased the time-to-compromise by over 30 times.

In conclusion, the evaluation results show that DOLOS is an effective defense tool against expert human attackers with considerable knowledge of system security.

## G. Professional Penetration Tester

This section outlines the opinion that a professional penetration tester reported on DOLOS. We provided the pentester with access to the architecture of DOLOS and the MTD modules implemented and asked him how he would carry out an attack on a production system armed with DOLOS.

The goal of this evaluation is to answer the following two research questions: (i) can DOLOS be effective in slowing down professional attackers? (j) Can DOLOS help CSIRT to expose stealthy attackers who try to compromise the systems without being detected?

We report the main conclusions of the professional pentester. He would begin the evaluation by obtaining information on the operating system run by the host, for instance through the *ping* command. Based on the time-to-live field in the response packet, it is possible to identify the operating system running on the machine, as DOLOS does not implement any MTD countermeasure against this type of interaction. Successively, the pentester would begin a system-wide scan to reveal open ports and an enumeration of running services with an automated tool. Since DOLOS networking MTD modules are configured to block and prevent banner grabbing and enumeration, the results from the scan would appear suspicious to a professional. The pentester would then move to a more curated, stealth approach, enumerating the running services manually using a standard fixed request string. The majority of currently implemented DOLOS networking modules provide a low-interaction environment designed to fool automated scanning tools. Through manual inspection, the professional pentester would slowly identify DOLOS fake services and converge on the set of real services offered by the production system. During this process, the pentester's IP address would be blocklisted multiple times by the automated networking modules of DOLOS, which would require a continuous IP change to pursue the attack. In the real world, this can be achieved by using multiple VMs from a cloud provider but would increase the time and cost required for exploitation. Eventually, the professional pentester would identify all DOLOS fake services and obtain access to the production system by exploiting the real vulnerable services.

## H. Professional Penetration Tester: Lessons Learned

As is often the case with cybersecurity defenses, a professional adversary dedicating enough time, resources and effort would be able to identify DOLOS MTD modules and eventually converge on the real production services. However, the goal of DOLOS is not to stop an attacker completely but rather to slow him down and expose his activity to CSIRT for remediation. To identify the real production services, a professional attacker would have to manually interact with all available services and filter out DOLOS's fake services one at a time. This activity requires a considerable amount of time, as it involves manual analysis of the responses from each deployed networking module. Moreover, interactions with several DOLOS networking modules would result in blocklisting of the attacker's IP address. This would further increase the time required for the attack, forcing the adversary to switch between multiple machines to continue the activity. Finally, the interaction with DOLOS networking modules would raise several alerts in the Controller which would relay them to the CSIRT. Therefore, we can conclude that the answers to research questions (i) and (j) are affirmative.

The professional pentester report provided us with important insights in how to improve DOLOS and MTD techniques further. An important point highlighted by the pentester is that IP blocklisting makes it trivial to identify fake services. Indeed, interactions with spoofed services are often used as a trigger for IP blocklisting by DOLOS. While this approach can be effective against automated tools and adversaries who rely on similar tools, this measure is not sufficient against a professional attacker, which can exploit this information to filter DOLOS services. Furthermore, the professional pentester could filter out MTD modules individually because DOLOS does not dynamically mutate the attack surface over time and the same port numbers stay assigned to the same fake services. The traditional MTD technique of periodically randomizing services would not work to address these issues. In fact, it would make it *easier* for an attacker to identify real services: since DOLOS would randomize only fake services, the remaining fixed, unchanging ones would be the real services. Allowing DOLOS also to randomize real services would incur in all the drawbacks of traditional MTD randomization techniques, as well as providing DOLOS with extended access to the real production system, which we strive to avoid.

*Countermeasures.* These limitations have been addressed and DOLOS improved by integrating a new dynamic deception approach as described below. We can exploit the fake services exposed by DOLOS to identify suspicious IP addresses and implement a *transparent filter* on the open ports associated with real production services. Any IP address that interacts with a DOLOS service is added to a suspicious list. Any connection coming from an IP in the suspicious list towards a real service is re-routed to a DOLOS-controlled port where a similar fake service is exposed. From the adversary's point of view, the real service port is exposing a fake service. However, any legitimate user who did not interact with DOLOS modules will be connected to the real production service without disruption. This dynamic deception approach can be further extended to introduce full, system-wide randomization of services without affecting any real service for legitimate users. After detection of suspicious activity from an IP address, DOLOS can use transparent filtering to make real services unreachable for the attacker and at the same time re-randomize all running DOLOS fake services. From the point of view of the adversary, this would effectively result in *all* the services and ports on the machine changing at the same time. However, it would not affect legitimate users, and they would still be able to access real services without interruption. These functionalities do not require implementing new modules and are already achievable with the current DOLOS architecture. The Management module of the DOLOS Controller already implements a command to re-initialize all MTD modules deployed by an Agent, which can be used to provide re-randomization of the fake services. The implementation of the port filtering can be achieved by installing a simple rule in any programmable router or IDS in the network. The DOLOS Controller instructs the relevant network device to change the destination port of any packet matching a specific triplet < *source IP, dest. IP, dest. port*>, where source IP is an address in the suspicious list, and destination IP and port identify the socket of the real services to protect.

Finally, we can further strengthen the resiliency of the suspicious list by implementing *enumeration fingerprinting* techniques in DOLOS [24], [25]. The professional pentester report highlights how advanced adversaries employ manual

TABLE IV
OVERHEAD OF DOLOS UNDER DIFFERENT CONDITIONS

| **Configuration** | **CPU Overhead** | **RAM Overhead** |
|---|---|---|
| DOLOS Core only | 0% | 3.1MB |
| All modules idle | 0% | 10.7MB |
| Under Port scan | 0.1% | 40.8MB |
| Under filesystem scan | 0.1% | 52.1MB |
| Under Port scan<br>Under filesystem scan | 0.2% | 52.1MB |

techniques for enumeration to be more stealthy and to analyze the responses from the services better. Typically, an adversary does this through the use of fixed query requests using tools such as netcat. DOLOS can exploit this by creating a fingerprint of all connection requests received by any of its networking modules. This fingerprint can be later used to identify further connection requests *even if they come from the same adversary using other IP addresses*. We leave the implementation of the transparent filter module based on enumeration fingerprinting as future work.

### I. Overhead

In the final section of our evaluation, we assess the impact of DOLOS on system performance from the point of view of CPU and memory usage overhead. The main findings in Table IV show that the impact of DOLOS on both CPU and RAM is negligible. When idle with only the core module loaded, the overhead in the system is essentially zero. Loading and deploying all DOLOS tools results in similar negligible performance impact, with only $\sim$ 11MB of RAM used by the system. We could detect any measurable CPU overhead only when the system was actively under attack. When under port scan with all modules deployed, we registered a 0.1% increase in CPU load and a $\sim$ 41MB overhead in system RAM usage. Similarly, with all modules deployed and a file-scanning script running in the system, we barely registered a 0.1% increase in CPU usage and a 52MB increase in system RAM usage. Finally, when under both port scan and filesystem scanning, the CPU overhead reached a brief peek of 0.2%, while the system RAM usage remained similar at +52MB. By all measures, we can say that DOLOS adds negligible overhead to the systems in which it is deployed.

## V. DISCUSSION

This section discusses potential pitfalls of DOLOS and future development directions toward a fully autonomous active defense agent.

### A. Increase in Attack Surface

The main potential pitfall of DOLOS is the increased attack surface it provides to adversaries. A general rule of cybersecurity is if you don't need something, then don't have it. This especially applies to open ports and exposed services. If a service is not necessary to achieve the function of a system, it should be removed, and its port should be closed to reduce the attack surface for potential adversaries. The DOLOS approach of embedding fake services in production systems flips this concept on its head. While it is undeniable that adding fake services can potentially introduce new attack vectors in a system, we believe that our evaluation of DOLOS shows the advantages outweigh the risks. The container-based architecture of DOLOS and the multi-layer security features described in Section III-C minimize the risk that compromise of DOLOS services leads to privilege escalation and breach of the container. Moreover, by setting up seccomp profiles, capabilities, and AppArmor in a controlled manner, the container's access to the underlying system is restricted, reducing the ability of an attacker to exploit DOLOS and compromise the production system.

### B. Towards an Autonomous Agent

Currently, the DOLOS Controller implements an interface through which security teams can view and update the configuration of DOLOS modules, as well as view logging information and alerts. A promising future direction is to introduce a new component in the Controller that can autonomously deploy DOLOS modules, as well as initialize their configuration based on the network environment and potential attackers. At the most basic level, such component could be modeled as an expert system with well-defined rules and conditions covering the most common use cases and system settings. A good starting point to design a system of this type would be the MITRE Engage framework [26], which defines a set of activities that can be performed in response to adversaries' actions.

A more advanced, more flexible form of automation would be the introduction of an intelligent component in the Controller, able to make decisions based on the assessment of the state of the deployment environment and the actions taken by attackers. An interesting approach in this direction is deep reinforcement learning, which has been recently applied to many complex problems from robotics [27] to cybersecurity [28]. Deep reinforcement learning is an interesting candidate for an intelligent controller since it excels at learning and solving complex high-dimensional problems. Controlling DOLOS's module deployment and configuration while considering the current state of the environment and adversarial interactions is a problem with a very large state space and numerous potential actions that can be applied. Deep reinforcement learning [29] uses an approximator, typically a deep neural network, to estimate the value function that is typically based on a table of state-action pairs in traditional reinforcement learning [30]. The ability of the neural network approximator to learn a complex function mapping arbitrary states to action/rewards appears to be a perfect fit for the control problem of active defense tools. Some initial research on applications of reinforcement learning to active defense and cybersecurity tools configuration exists [31], [32], [33]. However, these works are restricted to limited automation of specific tools, and no general solution to this problem exists to our knowledge. We leave the study of these research questions as future work.

## VI. Related Works

Researchers have proposed randomization and shuffling-based defense techniques for several decades, ranging from the first on n-version programming [34] to reconfigurable software and networks [35]. However, this approach to cybersecurity was formalized only recently under the umbrella of Moving Target Defense [1]. Since then, many approaches have been proposed in the literature at different levels of the system stack, with several review papers covering the topic [36], [37]. In this section, we analyze relevant related works and provide a comparison against Dolos.

### *A. Hardware-Level MTD Techniques*

In [38], the authors propose an MTD technique based on instruction-set randomization at the CPU level. They create process-specific randomized instruction sets based on a secret key to defend against code-injection attacks. The idea is that an attacker who does not know the randomization key will inject code that is invalid for the specific randomized CPU, failing the attack. The authors of [39] extend this approach to the whole software stack to prevent the execution of unauthorized binaries regardless of the attack vector. These approaches are orthogonal to our proposal, which does not aim to defend against this family of attacks.

### *B. Network-Level MTD Techniques*

Several works propose applications of different flavors of IP address randomization [7], [40], [41], [42]. Typically, this family of approaches periodically changes the IP address of production systems in a randomized manner to thwart adversarial reconnaissance. The authors of [42] exploit Software-Defined Networking (SDN) to introduce an IP mutation technique that is transparent to end hosts. They exploit the global view provided by the SDN controller to keep track of two different IP for each host in the network: the real IP and an ephemeral "virtual IP" that changes at regular intervals. In [7], the authors proposed an SDN-based IP multiplexing technique to randomize the addresses of end hosts in a similar fashion. Moreover, the authors of [43] propose a shuffling-based MTD technique that realies on SDN to randomize the network configuration of a host (mac/ip addresses, ports) based on the likelihood of it being in an adversary's attack path. While promising, these proposals showcase the complications introduced by traditional MTD techniques: in order to utilize the defense, a network needs to use specific technologies (SDN in this case) and implement complex IP translation methods that complicate the maintenance of the network. Dolos avoids these complexities by introducing randomization only to fake services and resources, which greatly simplifies maintainability. In any case, both these approaches can be deployed alongside Dolos transparently, as Dolos does not affect real services in any way. In [40], the authors propose a similar IP mutation technique based only on routing updates. The proposal uses similar virtual/real IP pairs to randomly mutate the address of end hosts. At the network edges, real IPs are translated to virtual, which are used to route packets inside the network. The consistency of the routing is achieved by regularly updating the routing tables within the network with the current virtual IP for each host. Similarly to other IP-randomization proposals, this approach introduces high complexity in network management, with end hosts constantly changing IP addresses. Moreover, it also introduces additional overhead required for the convergence of routing paths after virtual IP updates, which increases proportionally to the size of the network and the average number of branches at each router. Chowdhary et al [44] propose another approach to counter DDoS attacks based on SDN, Snort IDS, and Nash Folk Theorem to analyze the behavior of suspected nodes. The paper [45] introduces an MTD framework employing reinforcement learning to inspect network traffic and a network shuffling MTD technique to defend against detected threats. Similar to previous works, the proposal employs SDN to implement transparent IP randomization. These two approaches suffer the limitations mentioned above, namely the reliance on non-standard network technologies and increased network troubleshooting complexity. In [46] Mani et al. analyze the resilience of address randomization-based MTD techniques. They show that machine learning-based can be employed to detect address randomization techniques and, in some cases, even to predict future assigned addresses. The analysis of the limitations of address-randomization techniques further adds to the limitations of these approaches.

### *C. System-Level MTD Techniques*

Another thread of MTD research focuses on system diversification and redundancy. This family of works employs multiple, diverse implementations of system components designed to achieve the same end result in different manners. In [47], the authors propose a theoretical model to fully randomize the network stack of systems. From all possible combinations of the different layers of the stack, a subset of allowed combinations is defined. The proposed system randomizes the layers periodically by choosing from this set of allowed combinations. The authors of [48] propose a system that changes components of a running program in a randomized manner. The authors use multiple functionally-equivalent implementations of different program components and randomly chose variants at runtime. In [49], the authors combine shuffling and diversity techniques and introduce a multi-version web service architecture designed to maximize system dependability. The paper [50] introduces a redundancy-based MTD technique aimed at protecting webservers from command injection attacks. It uses multiple replicas of different software components of the webserver that are changed at runtime. All these works are orthogonal to our proposal and can be used in conjunction to Dolos to offer additional protection to production systems. However, it is worth noting that many of these techniques introduce considerable additional complexity in the management of systems, as well as sometimes prohibitive overhead [3]. On the other hand, by limiting the application of MTD techniques only to deceptive services and system properties Dolos provides strong security at low complexity and overhead cost.

### D. Deception-Based Techniques

At a high level, deception-based techniques can be divided into two categories: (1) system-level deceptive techniques and (2) service-level deceptive techniques. System-level techniques provide complete, fake systems aimed at attracting the attacker's focus and slowing down his progress. Service-level techniques aim at altering the properties of existing services in deceptive ways to confuse adversaries. System-level techniques (1) are the most closely related to DOLOS, since both are designed to limit changes to the production services, while at the same time slowing down attackers. Service-level techniques (2) are inherently orthogonal to our proposal, as they require changes to the real production services running on the systems, which is directly against one of the key design principles of DOLOS. This section reviews both categories of deception based techniques.

*1) System-Level Deception Techniques:* The authors of [51] propose a decoy-based shuffling approach to protect IoT devices. They exploit SDN to randomly perform network topology shuffling, hiding real IoT devices among fake decoy devices. In [52] the authors propose an adaptive cyber deception approach to predict the most likely attack path an adversary will follow, and successively deploy decoy nodes along the predicted path. Similarly to other deception-based approaches, these proposals suffer from the limitation that an adversary can typically detect decoy nodes with limited effort, and simply avoid them. DOLOS avoids this drawback by embedding deceptive services directly into the production systems through the DOLOS Agent, forcing attackers to either sift through numerous fake services to pinpoint the real ones, or drop the system entirely and move to a different target. Recently Qin et al. [13] provide a thorough evaluation on hybrid defense strategies relying on honeypots, honeynets, honeyfarms, honeytokens to provide cyberthreat intelligence and protect against attacks. Specifically, Honeynets [53], [54] consist of a virtual replica of a real-world production system that lacks production activities and authorized services. This deployment is primarily used for data collection, data which are further used in the detection and mitigation pipeline. Honeyfarms is a technique similar to honeynet used to simplify the deception network management that allows large-scale honeypot deployments. Similar to the above, these techniques suffer from limitations that they can be easily detected by the adversary due to the lack of production activities. Honeytokens [55], [56] are another set of intrusion detection mechanisms that consist of the deployment of bait files. Honeytokens are similar to decoy files in working, which are already implemented in DOLOS.

*2) Service-Level Deception Techniques:* In [57], the authors propose a deception-based technique to defend systems against attacks exploiting known vulnerabilities. The core idea is that if a vulnerability is patched, an attacker exploiting it will immediately know because the attack fails. The authors propose that security patches should include a deceptive component such that whenever an exploit against the patched vulnerability is attempted, the attack is transparently redirected to an unpatched decoy system, making the attacker believe the exploit succeeded. This approach is orthogonal to DOLOS, which was designed with the specific goal of requiring no modification to the production systems, except for the deployment of the DOLOS Agent. Furthermore, this approach requires manual activity to securely redact sensitive information before decoy generation. Nonetheless, the approach proposed by the authors can be adopted in conjunction with DOLOS to increase the deceptive capabilities of the systems. Albanese et al [58] propose a graph-based approach to cyber deception aimed at altering the view an attacker has of a target system. The authors define a theoretical approach and algorithmic solution to maximize the distance between the attacker's view of a system and the real state of the system. The main limitation of this proposal is that it requires modification to the real production systems and to the network traffic generated, which goes against the design principle of DOLOS of not interfering with the behavior of real services in any way. In [59], the authors analyze the effectiveness of decoy file-based defenses against cryptographic ransomware. Similarly, Ganfure et al. [60] study a similar decoy file-based defense called RTrap. Both these approaches focus on how to generate reliable decoy files that are hard to avoid by modern decoy-aware ransomware. These approaches are essentially an advanced version of traditional decoy files. DOLOS already provides modules implementing decoy files, and RTrap's advanced functionalities can be easily incorporated into these modules. In [61] the authors propose a system-level approach to MTD to defend against Stuxnet-like malware, by including MTD in the control and sensing units of cyber-physical systems. The authors show the effectiveness of this approach through simulations of multiple attacks. This proposal is orthogonal to our approach, as DOLOS is not specifically designed to run on resource-constrained cyber-physical systems. We plan to study the applicability of DOLOS in these settings in future work.

### E. Autonomous Agents-Based Techniques

The family of works that are conceptually closest to DOLOS is on autonomous cyber defense agents [62], [63], [64], [65]. Works in this family propose the use of agents based on artificial intelligence to reduce reliance on human cybersecurity experts and define a high-level architecture for the agents. These agents should be capable of autonomously planning and executing complex cyber defense activities to slow down and defeat automated malware. This family of works proposes only a high-level approximation of the architecture of such agents, and no implementation or detailed instantiation of the concept is given. In [66] the authors propose an orchestration framework, SODA, for cyber deception. SODA analyzes real-world malware behavior in the form of WinAPIs and maps them to MITRE ATT&CK techniques. A set of deception-based techniques are then designed to deceive the specific threat detected in the systems. SODA relies on traditional detection agents to identify malware running on the system and builds a profile of the malware based on WinAPIs calls. A subset of the suggested deception techniques is then selected by the user and applied through API hooking to deceive the malware. SODA has several differences and limitations compared to

DOLOS. The first, important difference is that SODA follows a traditional *reactive* approach to cyber defense — it requires an agent to detect the malware running —. Reactive measures typically have poor effectiveness against quickly mutating threats, as adversaries can evolve their strategies to avoid static defenses. DOLOS employs proactive techniques that are always active and don't rely on detection to thwart attackers. Secondly, SODA can provision systems only with limited deception capabilities at the API level, while DOLOS can implement essentially any MTD/deception technique. Moreover, SODA implements deception capabilities as hooks in the underlying system, which requires tight integration with the production system. Third, SODA is limited in interacting only with malware that is *already* running on the production system, since it requires analysis of the WinAPIs calls performed by the malware. This limitation heavily restricts its applicability compared to DOLOS. Finally, SODA is designed against static malware employing well-known attack patterns, and is not designed to defend against human adversaries. In contrast, DOLOS is very effective in thwarting and slowing down even expert human attackers. In [67] the authors propose an evolutionary approach to select the optimal MTD strategy based on a Wright-Fisher process. The proposed approach is designed to learn and describe the evolution trajectories of the attacker's and defender's strategies. This proposal is orthogonal to DOLOS, as it focuses on the optimal orchestration of MTD tools. Nonetheless, this related work provides an interesting direction for future extensions of an intelligent DOLOS Controller.

## VII. CONCLUSION

This paper presented DOLOS, a novel architecture that unifies cyber deception and moving target defense approaches. DOLOS is motivated by the insight that deceptive techniques are much more powerful when integrated into the production systems rather than deployed alongside them. DOLOS combines typical MTD techniques such as randomization, diversity, and redundancy to deception, and brings all these functionalities directly into the production systems through the DOLOS Agent. We showed that DOLOS can seamlessly and securely integrate into production systems through multiple layers of isolation, making it much more effective compared to typical deception techniques. Through an extensive and thorough evaluation of DOLOS against several different types of attackers, ranging from automated malware to professional penetration testers, we conclude that DOLOS is highly effective in slowing down attackers and protecting the integrity of production systems.

Future work is needed to investigate the applicability of autonomous agents to the DOLOS control logic for MTD modules deployment and orchestration. Furthermore, to provide a more complete context for autonomous decision-making, the Controller should correlate data coming from multiple Agents in real time. These improvements require a thorough comparison between synchronous and asynchronous communication paradigms with the controller, and an analysis of the advantages and limitations of each approach.

## REFERENCES

[1] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*, vol. 54. Berlin, Germany: Springer, 2011.

[2] S. Jajodia, V. Subrahmanian, V. Swarup, and C. Wang, *Cyber Deception*, vol. 6. Berlin, Germany: Springer, 2016.

[3] J.-H. Cho et al., "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 1, pp. 709–745, 1st Quart., 2020.

[4] C. Lei, H.-Q. Zhang, J.-L. Tan, Y.-C. Zhang, and X.-H. Liu, "Moving target defense techniques: A survey," *Secur. Commun. Netw.*, vol. 2018, pp. 1–25, Jul. 2018.

[5] G.-L. Cai, B.-S. Wang, W. Hu, and T.-Z. Wang, "Moving target defense: State of the art and characteristics," *Frontiers Inf. Technol. Electron. Eng.*, vol. 17, no. 11, pp. 1122–1153, Nov. 2016.

[6] F. De Gaspari, S. Jajodia, L. V. Mancini, and A. Panico, "AHEAD: A new architecture for active defense," in *Proc. ACM Workshop Automated Decis. Making Act. Cyber Defense*, Oct. 2016, pp. 11–16.

[7] D. P. Sharma, D. S. Kim, S. Yoon, H. Lim, J.-H. Cho, and T. J. Moore, "FRVM: Flexible random virtual IP multiplexing in software-defined networks," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 579–587.

[8] T. E. Carroll, M. Crouse, E. W. Fulp, and K. S. Berenhaut, "Analysis of network address shuffling as a moving target defense," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 701–706.

[9] M. Taguinod, A. Doupé, Z. Zhao, and G.-J. Ahn, "Toward a moving target defense for web applications," in *Proc. IEEE Int. Conf. Inf. Reuse Integr.*, Aug. 2015, pp. 510–517.

[10] Y. Huang and A. K. Ghosh, "Introducing diversity and uncertainty to create moving attack surfaces for web services," in *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. New York, NY, USA: Springer, 2011, pp. 131–151.

[11] K. Ferguson-Walter, M. Major, C. K. Johnson, and D. H. Muhleman, "Examining the efficacy of decoy-based and psychological cyber deception," in *Proc. USENIX Secur. Symp.*, 2021, pp. 1127–1144.

[12] A. Mairh, D. Barik, K. Verma, and D. Jena, "Honeypot in network security: A survey," in *Proc. Int. Conf. Commun., Comput. Secur.*, 2011, pp. 600–605.

[13] X. Qin, F. Jiang, M. Cen, and R. Doss, "Hybrid cyber defense strategies using honey-X: A survey," *Comput. Netw.*, vol. 230, Apr. 2023, Art. no. 109776.

[14] S. Mukkamala, K. Yendrapalli, R. Basnet, M. K. Shankarapani, and A. H. Sung, "Detection of virtual environments and low interaction honeypots," in *Proc. IEEE SMC Inf. Assurance Secur. Workshop*, Jun. 2007, pp. 92–98.

[15] T. Holz and F. Raynal, "Detecting honeypots and other suspicious environments," in *Proc. 6th Annu. IEEE Syst., Man Cybern. (SMC) Inf. Assurance Workshop*, Jun. 2005, pp. 29–36.

[16] *Ansible Orchestrator*. Accessed: Jan. 2023. [Online]. Available: https://www.ansible.com

[17] *Structured Threat Information Expression (STIX) Format*. Accessed: Jan. 2023. [Online]. Available: https://oasis-open.github.io/cti-documentation/stix/intro.html

[18] *Docker*. Accessed: Jan. 2023. [Online]. Available: https://www.docker.com/

[19] *Apparmor*. Accessed: Jan. 2023. [Online]. Available: https://apparmor.net/

[20] *Raw Sockets*. Accessed: Jan. 2023. [Online]. Available: https://man7.org/linux/man-pages/man7/raw.7.html

[21] NIST. (2014). *CVE-2014–7169*. Accessed: Mar. 2022. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2014-7169

[22] *Freepbx*. Accessed: Jan. 2023. [Online]. Available: https://www.freepbx.org/

[23] *Owasp Zed Attack Proxy*. Accessed: Jan. 2023. [Online]. Available: https://www.zaproxy.org/

[24] T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *IEEE Trans. Dependable Secure Comput.*, vol. 2, no. 2, pp. 93–108, Apr./Jun. 2005.

[25] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Proc. IEEE Symp. Secur. Privacy*, May 2013, pp. 541–555.

[26] *Mitre Engage*. Accessed: Jan. 2023. [Online]. Available: https://engage.mitre.org/

[27] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Feb. 2019, pp. 590–595.

[28] T. T. Nguyen and V. J. Reddi, "Deep reinforcement learning for cyber security," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 3779–3795, Aug. 2023.

[29] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, pp. 279–292, May 1992.

[31] T. Eghtesad, Y. Vorobeychik, and A. Laszka, "Adversarial deep reinforcement learning based adaptive moving target defense," in *Proc. Int. Conf. Decis. Game Theory Secur.*, 2020, pp. 58–79.

[32] R. Klima, K. Tuyls, and F. Oliehoek, "Markov security games: Learning in spatial security problems," in *Proc. NIPS*, 2016, pp. 1–8.

[33] X. He, H. Dai, P. Ning, and R. Dutta, "Dynamic IDS configuration in the presence of intruder type uncertainty," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[34] L. Chen and A. Avizienis, "N-version programming: A fault-tolerance approach to reliability of software operation," in *Proc. 8th IEEE Int. Symp. Fault-Tolerant Comput.*, vol. 1, 1978, pp. 3–9.

[35] K. Compton and S. Hauck, "Reconfigurable computing: A survey of systems and software," *ACM Comput. Surv.*, vol. 34, no. 2, pp. 171–210, Jun. 2002.

[36] J. Tan et al., "A survey: When moving target defense meets game theory," *Comput. Sci. Rev.*, vol. 48, May 2023, Art. no. 100544.

[37] R. E. Navas, F. Cuppens, N. Boulahia Cuppens, L. Toutain, and G. Z. Papadopoulos, "MTD, where art thou? A systematic review of moving target defense techniques for IoT," *IEEE Internet Things J.*, vol. 8, no. 10, pp. 7818–7832, May 2021.

[38] G. S. Kc, A. D. Keromytis, and V. Prevelakis, "Countering code-injection attacks with instruction-set randomization," in *Proc. 10th ACM Conf. Comput. Commun. Secur.*, Oct. 2003, pp. 272–280.

[39] G. Portokalidis and A. D. Keromytis, "Global ISR: Toward a comprehensive defense against unauthorized code execution," in *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. New York, NY, USA: Springer, 2011, pp. 49–76.

[40] E. Al-Shaer, Q. Duan, and J. H. Jafarian, "Random host mutation for moving target defense," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, Padua, Italy, 2013, pp. 310–327.

[41] S. Antonatos, P. Akritidis, E. P. Markatos, and K. G. Anagnostakis, "Defending against hitlist worms using network address space randomization," in *Proc. ACM Workshop Rapid Malcode*, Nov. 2005, pp. 30–40.

[42] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "Openflow random host mutation: Transparent moving target defense using software defined networking," in *Proc. 1st Workshop Hot Topics Softw. Defined Netw.*, Aug. 2012, pp. 127–132.

[43] S. Yoon, J.-H. Cho, D. S. Kim, T. J. Moore, F. Free-Nelson, and H. Lim, "Attack graph-based moving target defense in software-defined networks," *IEEE Trans. Netw. Service Manag.*, vol. 17, no. 3, pp. 1653–1668, Sep. 2020.

[44] A. Chowdhary, S. Pisharody, A. Alshamrani, and D. Huang, "Dynamic game based security framework in SDN-enabled cloud networking environments," in *Proc. ACM Int. Workshop Secur. Softw. Defined Netw. Netw. Function Virtualization*, Mar. 2017, pp. 53–58.

[45] S. Kim et al., "DIVERGENCE: Deep reinforcement learning-based adaptive traffic inspection and moving target defense countermeasure framework," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 4, pp. 4834–4846, Dec. 2022.

[46] G. Mani et al., "Machine learning based resilience testing of an address randomization cyber defense," *IEEE Trans. Dependable Secure Comput.*, early access, Jan. 11, 2023, doi: 10.1109/TDSC.2023.3234561.

[47] Q. Zhu and T. Basar, "Game-theoretic approach to feedback-driven multi-stage moving target defense," in *Proc. Int. Conf. Decis. Game Theory Secur.*, Fort Worth, TX, USA, 2013, pp. 246–263.

[48] M. Azab, R. Hassan, and M. Eltoweissy, "ChameleonSoft: A moving target defense system," in *Proc. 7th Int. Conf. Collaborative Computing: Netw., Appl. Worksharing (CollaborateCom)*, Oct. 2011, pp. 241–250.

[49] A. Gorbenko, V. Kharchenko, and A. Romanovsky, "Using inherent service redundancy and diversity to ensure web services dependability," in *Methods, Models and Tools for Fault Tolerance*. Berlin, Germany: Springer, 2009, pp. 324–341.

[50] E. Yuan, S. Malek, B. Schmerl, D. Garlan, and J. Gennari, "Architecture-based self-protecting software systems," in *Proc. 9th Int. ACM Sigsoft Conf. Quality Softw. Architectures*, Jun. 2013, pp. 33–42.

[51] M. Ge, J.-H. Cho, D. Kim, G. Dixit, and I.-R. Chen, "Proactive defense for Internet-of-Things: Moving target defense with cyberdeception," *ACM Trans. Internet Technol.*, vol. 22, no. 1, pp. 1–31, Feb. 2022.

[52] M. A. R. A. Amin, S. Shetty, L. Njilla, D. K. Tosh, and C. Kamhoua, "Hidden Markov model and cyber deception for the prevention of adversarial lateral movement," *IEEE Access*, vol. 9, pp. 49662–49682, 2021.

[53] Z. Heng-ru and G. Jie, "Research and design of network attack and defense platform based on virtual honeynet," in *Proc. Int. Conf. Comput. Inf. Sci.*, Dec. 2010, pp. 507–510.

[54] W. Fan, Z. Du, and D. Fernández, "Taxonomy of honeynet solutions," in *Proc. SAI Intell. Syst. Conf. (IntelliSys)*, Nov. 2015, pp. 1002–1009.

[55] J. Yuill, M. Zappe, D. Denning, and F. Feer, "Honeyfiles: Deceptive files for intrusion detection," in *Proc. 5th Annu. IEEE SMC Inf. Assurance Workshop*, Jun. 2004, pp. 116–122.

[56] B. M. Bowen, S. Hershkop, A. D. Keromytis, and S. J. Stolfo, "Baiting inside attackers using decoy documents," in *Security and Privacy in Communication Networks*, Y. Chen, T. D. Dimitriou, and J. Zhou, Eds. Berlin, Germany: Springer, 2009, pp. 51–70.

[57] F. Araujo, K. W. Hamlen, S. Biedermann, and S. Katzenbeisser, "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 942–953.

[58] M. Albanese, E. Battista, S. Jajodia, and V. Casola, "Manipulating the attacker's view of a system's attack surface," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 472–480.

[59] Z. A. Genc, G. Lenzini, and D. Sgandurra, "On deception-based protection against cryptographic ransomware," in *Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2019, pp. 219–239.

[60] G. O. Ganfure, C.-F. Wu, Y.-H. Chang, and W.-K. Shih, "RTrap: Trapping and containing ransomware with machine learning," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1433–1448, 2023.

[61] J. Tian, R. Tan, X. Guan, Z. Xu, and T. Liu, "Moving target defense approach to detecting Stuxnet-like attacks," *IEEE Trans. Smart Grid*, vol. 11, no. 1, pp. 291–300, Jan. 2020.

[62] A. Kott and P. Theron, "Doers, not watchers: Intelligent autonomous agents are a path to cyber resilience," *IEEE Secur. Privacy*, vol. 18, no. 3, pp. 62–66, May 2020.

[63] F. De Gaspari, S. Jajodia, L. V. Mancini, and G. Pagnotta, *Towards Intelligent Cyber Deception Systems*. Berlin, Germany: Springer, 2019.

[64] A. Kott et al., "Autonomous intelligent cyber-defense agent (AICA) reference architecture. Release 2.0," 2019, *arXiv:1803.10664*.

[65] P. Theron et al., "Reference architecture of an autonomous agent for cyber defense of complex military systems," in *Adaptive Autonomous Secure Cyber Systems*. Cham, Switzerland: Springer, 2020, pp. 1–21.

[66] M. S. I. Sajid et al., "SODA: A system for cyber deception orchestration and automation," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2021, pp. 675–689.

[67] J. Tan, H. Jin, H. Hu, R. Hu, H. Zhang, and H. Zhang, "WF-MTD: Evolutionary decision method for moving target defense based on Wright-Fisher process," *IEEE Trans. Dependable Secure Comput.*, early access, Dec. 27, 2022, doi: 10.1109/TDSC.2022.3232537.

Open Access funding provided by 'Università degli Studi di Roma "La Sapienza" 2' within the CRUI CARE Agreement