Proactive Defense for Internet-of-Things: Moving Target Defense with Cyberdeception

MENGMENG GE, Deakin University, Australia
JIN-HEE CHO, Virginia Polytechnic Institute and State University, USA
DONGSEONG KIM, University of Queensland, Australia
GAURAV DIXIT, Virginia Polytechnic Institute and State University, USA
ING-RAY CHEN, Virginia Polytechnic Institute and State University, USA

Resource constrained Internet-of-Things (IoT) devices are highly likely to be compromised by attackers because strong security protections may not be suitable to be deployed. This requires an alternative approach to protect vulnerable components in IoT networks. In this paper, we proposed an integrated defense technique to achieve intrusion prevention by leveraging cyberdeception (i.e., a decoy system) and moving target defense (i.e., network topology shuffling). We evaluated the effectiveness and efficiency of our proposed technique analytically based on a graphical security model in a software defined networking (SDN)-based IoT network. We developed four strategies (i.e., fixed/random and adaptive/hybrid) to address "when" to perform network topology shuffling and three strategies (i.e., genetic algorithm/decoy attack path-based optimization/random) to address "how" to perform network topology shuffling on a decoy-populated IoT network, and analyze which strategy can best achieve a system goal, such as prolonging the system lifetime, maximizing deception effectiveness, maximizing service availability, or minimizing defense cost. We demonstrated that a software defined IoT network running our intrusion prevention technique at the optimal parameter setting prolongs system lifetime, increases attack complexity of compromising critical nodes, and maintains superior service availability compared with a counterpart IoT network without running our intrusion prevention technique. Further, when given a single goal or a multi-objective goal (e.g., maximizing the system lifetime and service availability while minimizing the defense cost) as input, the best combination of "when" and "how" strategies is identified for executing our proposed technique under which the specified goal can be best achieved.

CCS Concepts: • **Security and privacy** → **Network security**; **Security protocols**.

Additional Key Words and Phrases: Internet-of-Things, moving target defense, graphical security models, software defined networking

ACM Reference Format:
Mengmeng Ge, Jin-Hee Cho, Dongseong Kim, Gaurav Dixit, and Ing-Ray Chen. 2020. Proactive Defense for Internet-of-Things: Moving Target Defense with Cyberdeception. *ACM Trans. Internet Technol.* 1, 1 (December 2020), 27 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Internet-of-Things (IoT) has received significant attention due to their enormous advantages. Advances in IoT technologies can be easily leveraged to maximize effective service provisions

Authors' addresses: Mengmeng Ge, Deakin University, 75 Pigdons Road, Geelong, Victoria, 3216, Australia; Jin-Hee Cho, Virginia Polytechnic Institute and State University, 7054 Haycock Road, Falls Church, VA, 22043, USA; Dongseong Kim, University of Queensland, St Lucia, Brisbane, Queensland, 4072, Australia; Gaurav Dixit, Virginia Polytechnic Institute and State University, 7054 Haycock Road, Falls Church, VA, 22043, USA; Ing-Ray Chen, Virginia Polytechnic Institute and State University, 7054 Haycock Road, Falls Church, VA, 22043, USA.

1533-5399/2020/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

to users. However, due to the high heterogeneity and resource constraints of composed entities in a large-scale network, we face the following challenges [Roman et al. 2013]: (1) distributed technologies for communications, data filtering, processing, and dissemination with various forms of data (e.g., text, voice, haptics, image, video) in a large-scale IoT network with heterogeneous entities (i.e., devices, humans); (2) severely restricted resources in battery, computation, communication (e.g., bandwidth), and storage, causing significant challenges in resource allocation and data processing capabilities; (3) highly adversarial environments with compromised, deceptive entities and data, which may result in detrimental impacts on the capabilities of critical mission-related decision making; and (4) highly dynamic interactions between individual entities, data, and environmental factors (e.g., network topology or resource availability), where each factor itself is also highly dynamic in time/space. Due to these characteristics of IoT environments, highly secure, lightweight defense mechanisms are in need to protect and defend the system (or network) against potential attacks. As a solution to protect and defend a system against inside attacks, many intrusion detection systems (IDSs) have been developed to identify and react to the attacks. However, the core idea of IDSs is reactive in nature and even though it detects intrusions which have already been in the system. Hence, this reactive mechanism normally would be late and ineffective reacting to actions by agile and smart attackers. To overcome the inherent limitation of IDSs due to this reactive nature, intrusion prevention systems (IPSs) have been developed to thwart potential attackers and/or mitigate the impact of the intrusions before they penetrate the system [Cho and Ben-Asher 2018]. In this work, we are interested in developing an integrated intrusion prevention mechanism based on cyberdeception (i.e., a decoy system) and moving target defense (MTD) and evaluating their effectiveness and efficiency by a graphical security model (GSM)-based evaluation framework in a Software Defined Networking (SDN)-based IoT network via simulation.

1.1 Research Goal & Contributions

We propose an integrated proactive defense system based on cyberdeception and MTD techniques as intrusion preventive mechanisms to minimize the impact of potential attackers trying to penetrate into IoT systems via multiple entries. We consider a SDN-based IoT system as a deployment network environment to support MTD in our work. The key merits of SDN technology are programmability and controllability, enabling the development of defense techniques integrating MTD with cyberdeception over a wide range of conditions. MTD can be applied in traditional networks with the support of hardware-based middleboxes. However, this potentially increases operational cost and overhead that affects normal functionality [Sengupta et al. 2020].

We made the following **key contributions** in this work:

- We are the first to propose an integrated proactive defense system by shuffling the topology of an IoT network consisting of both decoy and real nodes to create maximum hurdles and/or complexity to the attackers while minimizing the defense cost for executing MTD operations.
- We address the issues of “when” to perform network topology shuffling and “how” to perform network topology shuffling on a decoy-populated IoT network. We consider four “when-to-shuffle” strategies (i.e., fixed, random, adaptive, and hybrid) and three “how-to-shuffle” strategies (i.e., genetic algorithm, decoy attack path-based optimization, and random).
- We obtain security and performance measures, including the number of attack paths toward decoy targets, mean time to security failure (i.e., MTTSF or system lifetime), and defense cost, to analyze and identify which strategy can best achieve a system goal.
- We develop a graphical security model (GSM) to evaluate the proposed cyberdeception and MTD technique. The GSM offers design solutions to consider attack graphs (AGs) and/or attack trees (ATs) which can provide efficient methods to calculate the potential security (or vulnerability)

levels of attack paths. This allows us to analytically evaluate the effectiveness of the proposed cyberdeception and MTD integrated technique in a large IoT network.

A preliminary version of this work appeared in [Ge et al. 2020]. We have substantially extended [Ge et al. 2020] in algorithm design and evaluation, including: (1) development of a new "when-to-shuffle" strategy based on adaptive shuffling; (2) development of a new "how-to-shuffle" strategy based on decoy attack path-based optimization which is efficient in topology computation and able to scale to large IoT networks with thousands of nodes by setting threshold values for outgoing connections of nodes and the maximum path length; (3) usage of a new metric, packet delivery ratio, to measure the service availability in the presence of attacks; and (4) comparative performance analysis for 12 schemes resulting from a combination of four "when-to-shuffle" triggering strategies and three "how-to-shuffle" strategies.

1.2 Structure of This Paper

The rest of this paper is organized as follows. Section 2 provides a brief overview of the related work. Section 3 gives an overview of the system model. Section 4 describes the design of our proposed integrated proactive defense mechanism in detail. Section 5 shows evaluation results and analyzes the results observed. Section 6 discusses limitations and suggests future research directions. Section 7 summarizes key findings.

2 RELATED WORK

We briefly survey related work in three areas: (1) existing MTD and cyberdeception techniques for IoT; (2) security models and metrics; and (3) SDN technology for IoT. All the techniques and models can be applied to non-IoT networks (e.g., Cloud [Alavizadeh et al. 2020]). We focus on the current state-of-the-art research in IoT rather than other contexts because the proposed defense mechanism is designed for IoT networks to address unique challenges faced by IoT.

2.1 MTD and Defensive Deception Techniques for IoT

The concept of moving target defense (MTD) has been emerged to support the goal of proactive intrusion prevention. The basic idea behind MTD is to defend against attackers by continuously changing attack surface (e.g., system/network configurations) so as to increase attack complexity/cost and also invalidate the system intelligence collected by the attackers [Cho et al. 2020; Hong and Kim 2015]. MTD has been discussed with three main classes: shuffling, diversity, and redundancy. *Shuffling-based MTD* aims to confuse attackers by changing network/system configurations such as network addresses (e.g., IP addresses, MAC addresses, or port numbers), software migration, or network topology configuration. *Diversity-based MTD* increases attack complexity by using various types of system components (e.g., software, operating systems) which provide same functionalities. Lastly, *Redundancy-based MTD* provides security protections by dynamically using multiple replicas of system components in a network for the purpose of maintaining high system reliability [Cho et al. 2020].

Several existing MTD techniques have been developed to provide security protection for resource-constrained IoT environments. Ge et al. [2017] investigated address space layout randomization (ASLR) and evaluated its performance using the proposed Hierarchical Attack Representation Model (HARM). Several lightweight MTD techniques are also proposed by randomly choosing different types of cryptographic primitives [Plaga et al. 2018] or both cryptosystems and firmwares [Casola et al. 2013] for wireless sensor networks. Sherburne et al. [2014] proposed a dynamically changing IPv6 address assignment approach over the IoT devices using Low-Powered Wireless Personal Area Networks (LPWPANs) protocol to defend against various network attacks. Zeitz et al. [2017]

extended the work in [Sherburne et al. 2014] by presenting a design based on address rotation to obscure the communications among IoT devices. However, both do not have any experimental validation of the design. Mahmood and Shila [2016] developed an MTD security framework based on context-aware code partitioning and code diversification for IoT devices to obfuscate the attackers. Zeitz et al. [2018] developed micro MTD IPv6 as a solution to provide privacy and defense services to resource constrained devices by limiting the time available to an attacker performing reconnaissance attacks. Kouachi et al. [2018] proposed the anonymization of packet flow for IoT devices via micro One Time Address that changes the structure of IPv4 packets; however, the change of IP header requires re-configuring all the routers. Nizzi et al. [2019] provided a lightweight solution to change the addresses of IoT devices through network-wide address shuffling. Kahla et al. [2018] proposed live migrations as an MTD technique to move applications in a self-configuring fog architecture to provide security and trust; however, the overhead of migrations was not considered in their work. Wang et al. [2019] proposed a game theoretic zero-determinant approach for MTD in IoT to minimize extra operations required by Markov gaming while dominating the game based on a Zero-Determinant (ZD) strategy. Almohaimeed et al. [2019] proposed a model that prevents attackers from discovering device addresses in IoT networks by transmitting data via a dedicated MTD channel. Lin et al. [2019] proposed the migration of virtual security functions upon changes in traffic states to protect SDN enabled smart grid from resource exhaustion attacks. Hamada et al. [2018] proposed an honeypot-like MTD management framework which dynamically projects cell phones as fake and real gateways and sensors and creates real and fake sub-nets to deceive attackers. Similarly, Vuppala et al. [2019] proposed an MTD mechanism against side channel attacks by calculating the interval required for encryption re-keying after collecting a minimum number of trace leakages so as to reduce computation overhead.

Defensive deception techniques provide proactive defense services by adding an extra layer of defense on top of traditional security solutions (e.g., IDSs, firewalls, or endpoint anti-virus software) [Miyazaki et al. 2014]. La et al. [2016] introduced a game theoretic method to model the interaction between an attacker who can deceive a defender with suspicious or seemingly normal traffic and a defender in honeypot-enabled IoT networks. Anirudh et al. [2017] used honeypots for online servers to mitigate Distributed Denial of Service (DDoS) attacks launched from IoT devices. Dowling et al. [2017] created a ZigBee honeypot to capture attacks and used it to identify the DDoS attacks and bot malware. However, none of the works cited above analyzed the impact of deception techniques on system-level security and considered the tradeoff between defense cost vs. system-level security for an IoT system which allows distributed decoy deployment to achieve adequate coverage and provide cost-effective defense service [Pingree 2016]. Cho and Ben-Asher [2018] investigated an integrated defense system to identify what components of each defense mechanism can provide the best solution for 'defense in breadth' considering both enhanced security and defense cost. However, their work is based on model-based analysis without empirical verification. [Liu et al. 2020] implemented an SDN-based architecture to enable cyberdeception for legacy IP-based IoT devices where SDN-enabled honeypots can share attack information with SDN controllers to block attack traffics through updating flow rules on the switches.

All the works cited above focused on either MTD or cyberdeception. All MTD-based approaches applied to IoT did not consider network topology shuffling which can effectively interrupt the attack actions using compromised IoT devices as stepping stones. Furthermore, there is no current work on developing an integrated defense system equipped with both MTD and defensive deception techniques. With the deployment of decoys, the network shuffling-based MTD can not only confuse the attacker with changing connections among IoT devices but also provide a false view of the network and divert the attacker from actual IoT devices. This can effectively increase the attack effort and cost while decreasing the chances of real IoT devices to be compromised. Therefore, relative to the works cited above, we propose an integrated proactive defense based on cyberdeception and MTD techniques as intrusion preventive mechanisms that can effectively and efficiently mitigate the adverse effect of attackers before the attackers penetrate a target IoT system.

2.2 Security Models and Metrics

Graphical security models, including attack graphs (AGs) [Sheyner et al. 2002] and attack trees (ATs) [Saini et al. 2008], have been widely employed for security analysis in various types of networks. Several factors can contribute to the development and adoption of graphical models for security analysis: (1) with the advancement of Internet technologies, computer systems do not operate in isolation but interact with each other, resulting in increased attack surface; (2) quantitatively formulating attack behavior is critical for in-depth understanding of an attacker's goal, motivation, and tactics; and (3) identifying relevant system aspects to thwart attacks is critical for determining an effective security budget. Therefore, graphical security models can offer intuitive and systematic directions for assessing security vulnerabilities of systems and applying potential defense mechanisms [Hong et al. 2017]. In specific, an attack graph (AG) shows all possible sequences of the attack actions that eventually reach the target based on the vulnerability information and connectivity of the computer systems. As the network size increases, the size of an AG can grow exponentially, thus limiting its applicability. An attack tree (AT) is a tree with nodes representing the attacks and the root representing the goal of attacks. It systematically presents potential attacks in the network. However, AT is also not scalable with the growth of network size.

In order to address the scalablity issue, a two-layer Hierarchical Attack Representation Model (HARM) was introduced in [Hong and Kim 2015] by combining various graphical security models onto different layers. In a two-layer HARM, the upper layer captures the network reachability information and the lower layer represents the vulnerability information of each node in the network. The layers of the HARM can be constructed independently of each other. This decreases the computational complexity of calculating and evaluating the HARM compared with that of the existing single-layered graphical security models. [Ge et al. 2017; Hong and Kim 2015] investigated the effectiveness of defense mechanisms based on HARM. [Ge et al. 2017] developed a framework to automate security analysis of an IoT system by which HARM is used to assess the effectiveness of both device-level and network-level defense mechanisms based on various performance metrics such as attack cost and attack impact. [Hong and Kim 2015] evaluated MTD techniques in a virtualized system based on HARM using a risk metric. However, three different MTD techniques, including shuffling, diversity and redundancy, were separately evaluated without considering an integrated defense system. Relative to the works cited above, we also leverage HARM as our graphical security model since it scales with large IoT systems. Unlike the cited works above, we develop a HARM model specifically for security analysis of our proposed integrated defense system using both cyberdeception and MTD techniques.

In the literature, a risk-based security model has also been used to assess the effectiveness of defense mechanisms [Abie and Balasingham 2012; Rullo et al. 2017; Savola et al. 2012]. Abie and Balasingham [2012] proposed a risk-based security framework for IoT environments in the eHealth domain to measure expected risk and/or potential benefits by taking a game theoretic approach and context-aware techniques. Savola et al. [2012] proposed an adaptive security management scheme considering security metrics to deal with the challenges in eHealth IoT environments. However, only high-level ideas about the metrics were described without taking into account key characteristics of IoT environments that would require lightweight solutions. Rullo et al. [2017] proposed a method to come up with the optimal security resource allocation plan for an IoT network consisting of mobile nodes using a risk metric estimated by reflecting an economic perspective. However, only device-level evaluations were considered without showing system-level evaluations.

Relative to works cited above, we develop a scalable lightweight HARM model to evaluate the deployment of an integrated defense mechanism for an IoT environment by meeting both system security and performance requirements.

2.3 SDN Technology for IoT

Software defined networking (SDN) is a promising technology to flexibly manage complex networks. In the SDN-based architecture, the control logic is decoupled from the switches and routers and implemented in a logically centralized controller; the controller communicates with the data forwarding devices via the southbound application programming interface (API) and provides the programmability of network applications using the northbound API. OpenFlow (OF) is the most widely used southbound API which provides the specifications for the implementation of OF switches (including the OF ports, tables, channels, and protocols) [Foundation 2012]. Some SDN solutions are applied to IoT networks for data flow control among IoT devices [De Oliveira et al. 2015], data exchange reduction in wireless sensor networks [Galluccio et al. 2015], wireless access networks [Lei et al. 2014], mobile networks [Bernardos et al. 2014], smart urban sensing [Liu et al. 2015], and topology reconfiguration decision making in wireless sensor networks [Ge et al. 2018]. Unlike the above cited works, our work considers a general IoT network with the support of SDN functionality for network topology shuffling where an IoT network consists of both decoy nodes and real nodes.

3 SYSTEM MODEL

In this section, we discuss our system model, including (1) the network model in an IoT environment with the support of SDN technology; (2) the attack model describing the attacker's capabilities and attack goals considered in this work; and (3) the defense model addressing defense mechanisms deployed in the given network.

3.1 Network Model

In this work, we consider an IoT network (e.g., a smart hospital) which consists of servers and IoT nodes. IoT nodes collect data and periodically deliver them to servers via single or multiple hops for further processing. IoT nodes of different functionalities and servers are placed in different Virtual Local Area Networks (VLANs) in the given network. We assume SDN technology [De Oliveira et al. 2015; Galluccio et al. 2015; Gärtner 2003; Lei et al. 2014] is applied to the IoT network in order to effectively and efficiently manage and control nodes. We consider one SDN controller to be deployed in a remote server. The SDN controller communicates with SDN switches and manages flows between IoT nodes and servers which are connected to switches. Users from the Internet can request services from the servers and will not interact with IoT nodes directly. We will further detail the network scenario in our case study in Section 5.1.

3.2 Node Model

We characterize a node's attributes by four aspects: (1) whether a node is compromised or not (i.e., $n_i.c = 1$ for compromised; $n_i.c = 0$ otherwise); (2) whether a node is a real node or a decoy (i.e., $n_i.d = 1$ for a decoy; $n_i.d = 0$ for a real node); (3) whether a node is a critical node with confidential information that should not be leaked out to unauthorized entities (i.e., $n_i.r = 1$ for a critical node; $n_i.r = 0$ otherwise); and (4) a list of vulnerabilities that a node is vulnerable to (i.e., $n_i.v = \{v_1, \ldots, v_m\}$ where m is the total number of vulnerabilities). Hence, node i's attributes are represented by:

$$A_{n_i} = [n_i.c, n_i.d, n_i.r, n_i.v]. \tag{1}$$

3.3 Attack Model

In this work, we consider the following attacks that may lead to breaching system security goals:

- *Reconnaissance attacks*: Outside attackers are able to perform scanning attacks to identify vulnerable targets (e.g., a server) and then break into a system (or a network). The success of this attack demonstrates the successful identification and compromise of vulnerable targets by the outside attacker and leads to the loss of system integrity. This is related to triggering the system failure based on the security failure condition 1 (SFC1) in Section 3.5.
- *Data exfiltration attacks*: Inside, legitimate attackers are able to use credentials (e.g., login credentials or a legitimate key to access resources) obtained from a compromised node to leak confidential information to unauthorized, outside entities. The success of this attack results in the leakage of confidential information to unauthorized parties and leads to the loss of confidentiality. This is related to triggering the system failure based on the security failure condition 2 (SFC2) in Section 3.5.

We make the following **assumptions on attack behaviors and goals** to characterize attackers:

- An attacker is assumed to have limited knowledge on whether a given node is decoy (i.e., a fake node mimicking a real node) or not. The attacker's capability to detect the deception depends on the knowledge gap between the attacker and the real system state (i.e., how effectively the deployed decoy system mimics the real system in a sophisticated manner). We characterize the level of an attacker's intelligence in detecting a decoy node by the degree (or probability) at which the attacker interacts with the decoy node, as described in Section 3.4.
- An attacker's behavior is monitored after interacting with a decoy. If the attacker realizes the existence of a decoy, it terminates interactions with the decoy immediately and attempts to find a new target to break into the system.
- An attacker's ultimate goal is to compromise servers to leak confidential information to unauthorized entities outside the IoT network.
- An attacker is capable of identifying and compromising unpatched exploitable vulnerabilities or unknown vulnerabilities in a given IoT network.
- An attacker is highly unlikely to compromise servers directly as each server is assumed to have strong protection mechanisms. Therefore, the attacker can exploit vulnerable IoT nodes as entry points, move laterally within the network after the exploitation, and eventually compromise servers by identifying and exploiting unpatched or unknown vulnerabilities.
- The SDN controller is assumed to be well-protected where communications between the SDN controller and SDN switches are secure [Gärtner 2003].

3.4 Defense Model

We assume traditional defense mechanisms are in place in the IoT network, including a network-based IDS, firewalls, and anti-virus software on servers. The IDS is capable of monitoring the whole IoT network and creates alerts on detected intrusions for incident responses. This work focuses on two types of intrusion prevention mechanisms, namely, cyberdeception and MTD, to divert attackers from real IoT nodes and dynamically change the attack surface to increase attack complexity.

3.4.1 Decoy System as Defensive Deception. A defender (i.e., system) can defensively deceive attackers with the purpose of luring them into a decoy system and interacting with them to capture and analyze malicious behaviors and reveal intentions/strategies. The decoy system is deployed independently from the real system. Accordingly, we assume that normal, legitimate users are not aware of the existence of the decoy system while the defender will receive alerts caused by the malicious intrusions if an attacker breaks into the decoy system. We consider two types of decoys utilized throughout an IoT network in this work:

(1) **Emulation-based decoys**: This type of decoys allows defenders to create a variety of fake assets and to provide a large-scale coverage across the network.
(2) **Full OS-based decoys**: This type of decoys enables the replication of actual operating system and software running on production devices to increase the engagement possibility of the attacker.

Both emulation-based and full OS-based decoys can be autonomously created to fit within the environment without changing the existing infrastructure. To increase overall chances of exploiting decoys by attackers, a combination of diverse forms of decoys with various interactive capabilities can be created to resemble legitimate nodes. There exists an intelligence center performing the following tasks: (1) create, deploy, and update a distributed decoy system; (2) provide automated attack analysis, vulnerability assessment, and forensic reporting; and (3) integrate the decoy system with other prevention systems (e.g., security incident and event management platform, firewalls) to block attackers. The module for the decoy node deployment can be implemented and placed in a remote server. We create a design parameter, P_d, indicating the probability that an attacker interacts with an individual decoy node. To be specific, we consider P_d^{em} as the probability that an attacker interacts with an emulation-based decoy and P_d^{os} as the probability that an attacker interacts with a full OS-based decoy ($P_d^{em} \leq P_d^{os}$ as full-OS-based decoys are considered as having more sophisticated services with more cost).

3.4.2 Network Topology Shuffling-based MTD. We consider Network Topology Shuffling-based MTD (NTS-MTD) to change the topology of a given IoT network. NTS-MTD is to be triggered following the concept of event-based MTD in that the network topology changes upon the occurrence of an event. We assume that the SDN controller can control and change flows among nodes in an SDN-based IoT system. We combine cyberdeception and NTS-MTD by means of network topology shuffling to change the attack surface of the IoT network populated with both real and decoy nodes. The details of the proposed decoy system and the event-based NTS-MTD will be described in Section 4.

3.5 Security Failure Conditions

A system fails when either of following two conditions is satisfied:

- **Security Failure Condition 1 (SFC1)**: This system failure is closely related to the attacker's successful reconnaissance attacks and accordingly their successful compromise of system components. We define this system failure based on the concept of Byzantine Failure [Gärtner 2003]. That is, when more than one third of legitimate nodes are compromised, the system fails due to the loss of system integrity.
- **Security Failure Condition 2 (SFC2)**: This system failure occurs when confidential information is leaked out to unauthorized entities by inside attackers (or compromised nodes), which perform data exfiltration attacks. Th system fails due to the loss of data confidentiality.

4 PROPOSED PROACTIVE DEFENSE MECHANISMS

In this section, we describe our proposed NTS-MTD technique in five main aspects: (1) deployment of decoy nodes; (2) when to perform network topology shuffling with decoy nodes; (3) how to perform topology network shuffling with decoy nodes; (4) performance metrics to measure security, performance, and service availability of proposed proactive defense mechanisms; and (5) graphical security model for security analysis.

4.1 Deployment of Decoy Nodes

In this section, we describe the initial deployment of decoy nodes in an IoT network. Both server and IoT nodes are deployed in the IoT network. As the network is divided into different virtual local area networks (VLANs), we place IoT decoy nodes into each VLAN based on the deployment of real nodes in the corresponding VLAN. At least one decoy server needs to be deployed to interact with the attacker and reveal the attacker's intent. Note that we can deploy more decoys if the VLAN has a large number of real nodes with different types. When adding decoy nodes, we connect real IoT nodes with decoy nodes by directing fake traffic to decoys (e.g., through deploying a script on real IoT nodes by the intelligence center to generate traffic or placing a fake credential on real IoT nodes that diverts the attacker to decoys) in order to lure attackers into the decoy system. The SDN controller controls flows from real IoT nodes to decoy nodes or from decoy nodes to decoy nodes, and also from real IoT nodes to real IoT nodes through updating flow tables in SDN switches. There will be no flows from decoy nodes to real nodes as decoy nodes are used to divert attackers from the real system; once the attacker is lured into the decoy system, it will be diverted to other decoys within the decoy system by SDN switches based on the flow table updated by the SDN controller while the behavior will be monitored by the intelligence center explained in Section 3.4.1; if the attacker detects a decoy node, it will terminate the interaction with the decoy node and look for a new target to break in. In this work, we consider changing connections from real nodes to both real and decoy nodes to increase the complexity of connection changes. The reason is that an intelligent attacker may observe the pattern of traffic flows among nodes through statistical analysis once inside the network and infer the trap (i.e., diversion to the decoy system) if only connections from real nodes to decoys are shuffled.

Updated flows (either addition or removal) may affect normal flows from IoT nodes to servers for service delivery. In practice, IoT nodes will consume more energy to deliver more flows and may delay the time to send normal packets toward the server. We use packet delivery ratio as a metric for measuring service availability, as discussed in Section 4.4.

Initially we create decoy nodes with added connections to some randomly chosen real nodes based on the deployment of real nodes in each VLAN. The randomly generated network topology will be used as the initial topology and then fed into the shuffling optimization algorithm to identify an optimal network topology.

4.2 When to Perform Network Topology Shuffling with Decoy Nodes

We can use a fixed time interval to execute NTS-MTD. Apparently the fixed time interval is the most important parameter of this strategy because if the interval is too short, the defense cost will be high although the system lifetime may be prolonged because frequent topology shuffling can mislead the attacker to decoy paths and nodes and thus keep real nodes from the attacker. On the other hand if the fixed time interval is too long, it will adversely shorten the system lifetime because of infrequent network shuffling. We call this strategy the fixed time interval strategy or just "fixed" for short. A variation of this "fixed" strategy is to have the time interval follow a distribution with the mean being the same as the fixed time interval used by the fixed strategy. This will add some stochastic nature to the time interval which is treated as a random variable. We will call this strategy "random" for short.

Alternatively, we can execute NTS-MTD when a condition is detected true, for example, based on the system security vulnerability level detected by the system. We will call this strategy "adapative" for short. To be specific, the system security vulnerability level at time t, denoted by $SSV(t)$, is measured by two dimensions: (1) how many legitimate, inside nodes are compromised until time t, which is associated with `SFC1`; and (2) how many neighboring nodes of a critical node (i.e., $n_i.r = 1$)

within k hops from the critical node i are compromised until time t, which is related to SFC2. Of course we do not know which node is actually compromised unless the IDS has detected it. However, given the list of vulnerabilities that a node is vulnerable to, as discussed in Section 3.2 and the compromise rate for each vulnerability which is documented in several sources such as [NIST 2005] we can estimate the probability that a node is compromised at time t. Note that when the system meets either SFC1 or SFC2, the system fails, leading to $SSV(t) = 1$. Otherwise, $SSV(t)$ is computed by:

$$SSV(t) = w_1 \frac{CN(t)}{N} + w_2 \frac{CN_{ck}(t)}{N_{ck}(t)} \tag{2}$$

Here w_1 and w_2 are weights to consider SFC1 and SFC2, respectively, where $w_1 + w_2 = 1$. N is the total number of real nodes which is known at deployment time and $CN(t)$ is the number of compromised, real nodes at time t which may be estimated from the compromise rate of each vulnerability that a node is vulnerable to. (See more about this in Section 5.1.) $N_{ck}(t)$ is the total number of real nodes within k hops from given critical nodes at time t which may be obtained from the topology shuffled at time t while $CN_{ck}(t)$ is the total number of compromised, real nodes within k hops from critical nodes which again can be estimated from the compromise rate of each vulnerability that a node is vulnerable to. Since there may be multiple critical nodes which have confidential information that should not be leaked to outside unauthorized parties, we estimate $CN_{ck}(t)$ by:

$$CN_{ck}(t) = \sum_{i \in L_k(t)} n_i.c(t) \tag{3}$$

where $L_k(t)$ is the number of real nodes that belong to neighbors of any critical nodes within k hops from them at time t and $n_i.c(t)$ refers to whether node i is compromised ($n_i.c(t) = 1$) or not ($n_i.c(t) = 0$) at time t. The cardinality of $L_k(t)$ (i.e., $|L_k(t)|$) yields $N_{ck}(t)$. Note that as the network topology keeps changing due to the execution of NTS-MTD, both $N_{ck}(t)$ and $CN_{ck}(t)$ are functions of time to reflect their dynamic changes. If $L_k(t)$ includes any critical nodes being compromised, the system meets SFC2 and fails. That is, $SSV(t) = 1$ and no further detection of system security level is needed.

Lastly we can have a hybrid strategy that will degenerate to the adaptive strategy when the triggering time as determined by the adaptive strategy is smaller than the fixed time interval used by the fixed strategy and will degenerate to the fixed strategy otherwise. The four "when-to-shuffle" strategies will be more formally defined and labeled later in Section 5.2 when we perform evaluation.

4.3 How to Shuffle Network Topology with Decoy Nodes

We develop three strategies to address how to perform network shuffling when it is time to execute NTS-MTD. The basic idea of our design is to maximize the chance of the attacker exploiting decoy targets, thus effectively deterring or preventing its security attacks to real nodes. In order to reach a target node, an attacker could exploit a node as an entry point and use it as the stepping stone to compromise other nodes and further compromise the target. It may be able to find multiple attack paths via one or multiple entry points. An attack path describes a sequence of nodes that an attacker could compromise to reach the target node. We consider a set of attack paths AP for an attacker to reach all targets from all possible entry points. Each attack path ap is a sequence of nodes along the path. We use AP_r to represent a set of attack paths with real nodes as targets and AP_d to denote a set of attack paths with decoy nodes as targets. AP_r only contains real nodes while AP_d contains both real and decoy nodes. To be specific, if an attacker finds a real node as the entry point and compromises other real nodes until reaching a real target node, this is counted as an attack path in AP_r; however, it could be diverted to a decoy node. Once the attacker is lured into the decoy system, it will be diverted to other decoy nodes within the decoy system. If the attacker reaches a decoy target node, this is counted as an attack path in AP_d; however, if the attacker figures out the decoy node and terminates its interaction, it is not counted as an attack path because the attacker does not reach the decoy target node. Besides, decoy nodes could be updated or cleared once it is detected compromised by the intelligence center in which case the attacker will not recognize the same decoy node during subsequent attacks.

To maximize the chance of the attacker being misled to decoy targets, we develop the following two "how-to-shuffle" strategies:

- **GA-based optimization:** We design three metrics to be optimized in the algorithm: (1) The number of attack paths toward the decoy targets (N_{DT}^{AP}); (2) Mean Time To Security Failure (MTTSF); and (3) Defense cost (C_D). Computations of these metrics are described in Section 4.4.
- **Decoy path-based optimization:** Due to the high computational complexity of GA, we design a simple heuristic algorithm to provide a close-to optimal solution in topology shuffling. The algorithm takes a path-based optimization approach in two ways: (i) Shuffle edges (connections) from real IoT nodes to decoy nodes to randomize decoy connections; and (ii) Shuffle edges (connections) among real IoT nodes to maximize the number of attack paths toward decoy targets. Pseudocode and implementation can be found in GitHub [Ge 2020].

The third strategy is a baseline strategy that generates a network topology based on a connection probability of a real/decoy node being connected to another decoy node. We call this strategy "random" meaning that the connection probability is a random variable in the range of [0, 1] which determines if a connection from a real/decoy node to another decoy node should be created in the resulting topology. The three "how-to-shuffle" strategies will be more formally defined and labeled later in Section 5.2 when we perform evaluation.

4.4 Metrics

We use the following metrics to measure security, performance, and service availability of the proposed proactive defense mechanisms:

- **Number of attack paths toward decoy targets** (N_{DN}^{AP}): This metric indicates the level of deception that diverts an attacker from the real system. N_{DN}^{AP} is calculated by $|AP_d|$ to sum attack paths toward the decoy targets.
- **Mean Time To Compromise (MTTC)**: This metric refers to the total amount of time that an attacker takes to compromise a series of nodes within the network until the system reaches a certain security vulnerability level SSV. MTTC is estimated by:

$$MTTC = \sum_{i \in S} S_i \int_{t=0}^{\infty} P_i(t)dt \tag{4}$$

  where S refers to a set of all system states and S_i is 1 when in state i the system does not reach the given SSV level and is 0 otherwise. $P_i(t)$ is the probability of the system being in state i at time t.
- **Mean Time To Security Failure (MTTSF)**: This metric measures the system lifetime indicating how long the system prolongs until the system reaches either SFC1 or SFC2 (described in Section 3.5). That is, MTTSF measures the system lifetime without occurring any security failure. MTTSF is measured by:

$$MTTSF = \sum_{i \in S} (1 - SF_i) \int_{t=0}^{\infty} P_i(t)dt \tag{5}$$

  where S is a set of all system states and SF_i returns 1 when system state i reaches either SFC1 or SFC2; 0 otherwise. $P_i(t)$ indicates the probability of the system being in state i at time t.

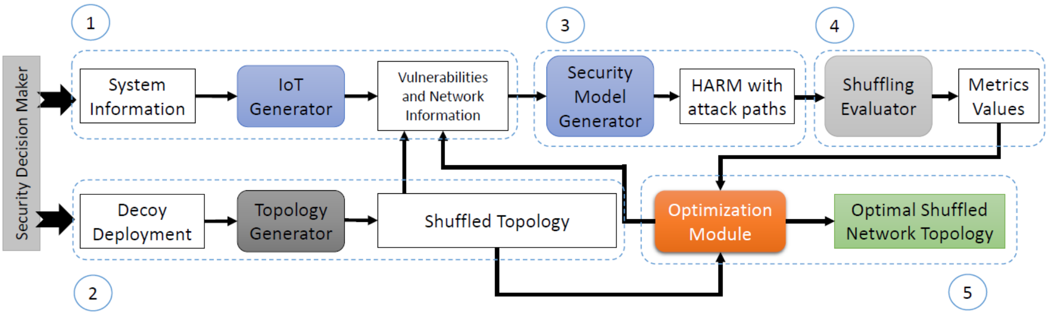


Fig. 1. Workflow for the security analysis.

- **Defense Cost** (C_D): This metric depicts the cost associated with shuffling operations. That is, we count the number of edges shuffled (i.e., from connected to disconnected or from disconnected to connected) by:
$$C_D = \int_{t=0}^{MTTSF} C_S(t) \tag{6}$$
where $C_S(t)$ refers to the number of shuffled edges at time t. Note that the same edge can be shuffled multiple times over time and each shuffling is counted as a separate MTD operation during the system uptime.
- **Packet Delivery Ratio (PDR)**: This metric measures service availability affected by topology shuffling. Because of topology shuffling, attackers tend to compromise nodes on attack paths. For each attack path in AP_r, a compromised node along the path may drop or manipulate packets travelling through it, thereby affecting service availability for service packets passing through the attack path. If packets are not dropped or manipulated by compromised nodes along the path (because the attacker may not want to get caught by the IDS) or if there is no compromised node along the attack path, then the path will be able to successfully deliver service packets. At each shuffling operation, we count the number of attack paths that can successfully do packet delivery and divide it by the total number of attack paths $|AP_r|$. When the system reaches either SFC1 or SFC2, we calculate the mean PDR over all shuffling operations. The focus of our work is to analyze the effect of attacks on service availability which can be disrupted due to packet loss caused by attacks. We assume that packet losses caused by collisions or errors will be handled by data link layer and network layer packet retransmission protocols and would not affect service availability represented by the packet delivery ratio metric.

4.5 Graphical Security Model for Security Analysis of NTS-MTD

We develop a graphical security model based on HARM to assess the security of an IoT network. Fig. 1 describes the workflow of our security analysis in five phases: network generation, topology generation, security model generation, shuffling mechanism evaluation, and shuffling optimization.

(1) **Phase 1**: The security decision maker provides the `IoT Generator` with the system information (i.e., an initial network topology and node vulnerability) to construct an IoT network.
(2) **Phase 2**: Given the network and initial deployment of decoys, the `Topology Generator` randomly generates a set of different topologies for GA-based shuffling and one topology for decoy path-based shuffling (i.e., add connections from real nodes to decoys/real nodes).
(3) **Phase 3**: The `Security Model Generator` takes the shuffled network as input and automatically generates a HARM model that captures all possible attack paths. We use a two-layer HARM as our graphical security model, with the upper layer capturing the node connectivity

information (i.e., nodes connected in the topological structure) and the lower layer denoting the vulnerability information of each node.

(4) **Phase 4**: The `Shuffling Evaluator` takes the HARM model as input along with evaluation metrics and computes results which are then fed into the `Optimization Module`.

(5) **Phase 5**: For GA-based shuffling, based on the initial set of shuffled topologies and associated evaluation results, the `Optimization Module` applies the multi-objective GA to compute the optimal topology for the IoT network. For decoy path-based shuffling, the `Optimization Module` takes the randomly shuffled topology from the `Topology Generator` and runs the heuristic algorithm to compute the close-to optimal topology.

5 NUMERICAL RESULTS & ANALYSIS

In this section, we first describe the simulation setup, introduction of 12 schemes, parameter table, implementation detail, and data collection process. Then we conduct a comparative performance analysis of 12 schemes of when and how to execute our proposed NTS-MTD technique.

5.1 Simulation Setup

We use an IoT network shown in Fig. 2 in our simulation and assume SDN is deployed to support connection changes. We consider a smart hospital scenario in the IoT context. Specifically, the network consists of four VLANs. There are two Internet of Medical Things (i.e., MRI and CT Scan) in VLAN1 (e.g., medical examination rooms), a smart thermostat, a smart meter, and a smart camera in VLAN2 (e.g., medical care units), a smart TV and a laptop in VLAN3 (e.g., staff office) and a server located in VLAN4 (e.g., server room). At the initial deployment, VLAN4 is connected with other three VLANs as IoT devices need to deliver information to the server for further processing. VLAN2 is also connected to VLAN3 for applications running on the laptop to control smart sensors as well as receive videos from the smart camera.

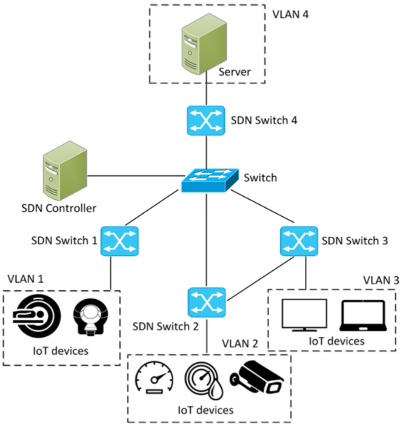


Fig. 2. A software-defined IoT network.

Table 1. Real node and vulnerability information.

Real Node	VLAN	CVE ID	CVSS Exploitability	Compromise Rate
MRI	VLAN1	CVE-2018-8308	6.8	0.006
CT Scan	VLAN1	CVE-2018-8308	6.8	0.006
Smart Thermostat	VLAN2	CVE-2018-11315	6.5	0.006
Smart Meter	VLAN2	CVE-2017-9944	10.0	0.042
Smart Camera	VLAN2	CVE-2018-10660	10.0	0.042
Smart TV	VLAN3	CVE-2018-4094	8.6	0.012
Laptop	VLAN3	CVE-2018-8345	4.9	0.004
Server	VLAN4	CVE-2018-8273	10.0	0.042

We collect software vulnerabilities from Common Vulnerabilities and Exposures (CVE)/National Vulnerability Database (NVD) [NIST 2005]. We assume each real node has one vulnerability that could be exploited by the attacker to gain a root privilege. More vulnerabilities could be chosen for nodes in the future work. This research work

Table 2. Decoy Node and Vulnerability Information.

Decoy Node	VLAN	CVE ID	CVSS Exploitability	Compromise Rate
CT Scan	VLAN1	CVE-2018-8308	6.8	0.006
		CVE-2018-8136	8.6	0.012
Smart Camera	VLAN2	CVE-2018-6294	10.0	0.042
		CVE-2018-6295	10.0	0.042
		CVE-2018-6297	10.0	0.042
Smart TV	VLAN3	CVE-2018-4094	8.6	0.012
		CVE-2018-4095	8.6	0.012
Server	VLAN4	CVE-2016-1930	10.0	0.042
		CVE-2016-1935	8.6	0.012
		CVE-2016-1962	10.0	0.042

focuses on proposing and evaluating the integrated proactive defense mechanism, rather than demonstrating capabilities of the graphical security model to analyze the security posture of the IoT network with multiple vulnerabilities. The vulnerability information of real nodes (i.e., CVE ID) is presented in Table 1. We also assume the compromise rate of each vulnerability. The compromise rate represents the frequency that an attacker could successfully exploit the vulnerability to gain root privilege per time unit (i.e., hour). We estimate the mean vulnerability exploitation time according to the exploitability metric of the base score from the Common Vulnerability Scoring System (CVSS) and calculate the compromise rate using the inverse of the mean vulnerability exploitation time. Specifically, we estimate the compromise rate as once per day (i.e., $1/24 = 0.042$) if the exploitability value is 10.0, twice per week (i.e., $1/84 = 0.012$) if the value of is around 9.0, once per week (i.e., $1/168 = 0.006$) if the value is around 7.0, and once per 10 days (i.e., $1/240 = 0.004$) if the value is around 5.0. This value will be used to calculate Mean Time to Compromise (MTTC) and Mean Time to Security Failure (MTTSF) by the HARM model. Once a node is compromised it can perform packet dropping or manipulating attacks to affect service availability. In practice, however, a compromised node may not drop or manipulate a packet passing through it, so it won't get caught by the network IDS. In our simulation, we consider a packet drop probability P_a^d and a packet manipulation probability P_a^m by the attacker.

We put one decoy node in each VLAN in the initial deployment of the decoy system. In order to lure attackers, each decoy is assumed to be configured to have multiple vulnerabilities. An attacker could exploit any vulnerability to gain the root permission of the node. The vulnerability information of decoys is listed in Table 2. We use emulated decoys for the CT scan, smart camera, smart TV, and full-OS based server.

We ran all simulations on a HPC cluster with Intel(R) Xeon(R) Gold 5120 CPU @ 2.20GHz (28 physical cores) and 256GB RAM. We use Ubuntu 18.04.4 LTS and PyCharm with Python 3.7.3. Simulation code (i.e., implementation of the system model and proposed mechanisms) can be found in GitHub [Ge 2020]).

5.2 Twelve Schemes to Execute NTS-MTD based on When and How Strategies

We investigate two aspects of NTS-MTD: (i) when-to-shuffle a network topology (in an interval or in an adaptive manner); and (ii) how to select a network topology (by a GA-based optimization, a decoy path-based optimization, or random shuffling).

Four strategies regarding **when-to-shuffle a network topology** are:

- **Fixed Shuffling (FS)**: This strategy is to execute NTS-MTD in a fixed time interval, γ_1, to shuffle the network topology.

- **Random Shuffling (RS)**: This strategy is to execute NTS-MTD in a random interval based on exponential distribution with mean λ.
- **Adaptive Shuffling (AS)**: This strategy is to execute NTS-MTD in an *adaptive manner* based on $SSV(t)$ with two given thresholds: (1) β to check the decrease of the SSV during a checking interval Δ; and (2) ρ to check the current system security vulnerability, $SSV(t)$, as described in Section 4.2. NTS-MTD is executed when the condition, $(SSV(t) - SSV(t-\Delta) > \beta) \wedge (SSV(t) > \rho)$, is true. This condition is checked whenever the system detects a compromised real node, thus reflecting the nature of an event-driven adaptive MTD.
- **Hybrid Shuffling (HS)**: This strategy is a mixture of AS and FS. Since AS triggers the execution of NTS-MTD until the event condition is detected, it may delay the execution of NTS-MTD unnecessarily especially in the beginning because security vulnerability does not necessarily increase rapidly in the beginning. To remedy this, we introduce an upper bound time limit (i.e., the maximum delay) for NTS-MTD execution. Specifically, the time interval to execute NTS-MTD is set to $\min[Int(AS), \gamma_2]$ where $Int(AS)$ returns a time interval when AS is used and γ_2 is the fixed time interval for the maximum delay when FS is used.

Three strategies regarding **how to select a network topology** are:

- **Random Network Topology (RNT)**: This strategy is a baseline strategy that selects a network topology based on a rewiring probability P_r of a node being connected with another node. Here P_r is critical in determining the overall network density in a given network.
- **GA-based Network Topology (GANT)**: This strategy selects a network topology that maximizes objective functions used in the GA, as discussed in Section 4.2.
- **Decoy Path-optimized Network Topology (DPNT)**: This strategy selects a network topology that maximizes the number of decoy paths for each real IoT node, as discussed in Section 4.2.

Since we have four "when" strategies and three "how" strategies for NTS-MTD execution, there are 12 schemes resulting from the combination of one "when" strategy and one "how" strategy, viz., FS-RNT (i.e., execution of NTS-MTD based on Fixed Shuffling (FS) and Random Network Topology (RNT)), RS-RNT, AS-RNT, HS-RNT, FS-GANT, RS-GANT, AS-GANT, HS-GANT, FS-DPNT, RS-DPNT, AS-DPNT, and HS-DPNT.

5.3 Parameter Table, Implementation Detail, and Data Collection Process

Table 3 summarizes the model parameters, their meanings, and default values used in our simulation runs. We used equal weights for w_1/w_2 and $w_N/w_M/w_C$, respectively, where the values can be adjusted based on the importance of a given component. Low intelligent attacker ($P_d^{em} = 0.9$ and $P_d^{os} = 1.0$) with medium attack severity ($P_a^d = P_a^m = 0.5$) is used in the baseline scenario. We analyzed the impact of attacker's intelligence and severity in Section 5.4.2 and 5.4.3. We set $k = 1$ due to the current topology of the example IoT network where a larger value can be selected for a larger network. Regarding parameters in GANT, we chose maximum generation $N_g = 100$ with a fairly high crossover rate to produce new offspring while choosing a low mutation rate to prevent the convergence to a local optimal (i.e., a high mutation rate may turn GA into random search, which needs to be avoided). We set $P_r = 0.5$ in RNT to avoid a minimal change of the network topology by a low value and high shuffling cost by a high value. In AS/HS, we set $\beta = 0.01$ to trigger the shuffling. We set $\rho = 0.1$ to represent a low tolerance of attacks. We conducted sensitivity analysis of varying ρ in Section 5.5.2. We set shuffling time intervals γ_1 and λ as 24 hrs in FS (shuffling once per day) and RS, respectively, in the baseline scenario to avoid high cost introduced by frequent shuffling and high delay of shuffling. We set maximum delay γ_2 as 120 hrs (shuffling once per 5 days) in HS because a low value may turn HS into FS while a high value may turn HS into AS. We conducted sensitivity analysis of varying γ_2 in Section 5.5.1.

Table 3. Design parameters, their meanings and default values.

Param.	Meaning	Value
w_1	A weight to consider the security vulnerability associated with SFC1	0.5
w_2	A weight to consider the security vulnerability associated with SFC2	0.5
P_d^{em}	Interaction probability of an attacker with an emulated decoy	0.9
P_d^{os}	Interaction probability of an attacker with a full-OS based decoy	1.0
P_a^d	Probability of a packet to be dropped	0.5
P_a^m	Probability of a packet to be manipulated	0.5
k	Number of hops to determine a node's ego network	1
N	Total number of network topologies with initial decoy deployment and randomly generated connections between real and decoy nodes used in GANT	100
w_N	A weight to consider in objective function used in GANT	1/3
w_M	A weight to consider in objective function used in GANT	1/3
w_C	A weight to consider in objective function used in GANT	1/3
N_g	Maximum number of the generation used in GANT	100
r_c	Crossover rate used in GANT	0.8
r_m	Mutation rate used in GANT	0.2
P_r	Probability of an edge being shuffled in RNT (i.e., add/remove an edge)	0.5
β	Threshold used to estimate the decrease of the system security vulnerability level during the time used in AS/HS	0.01
ρ	Threshold of tolerating system security vulnerability used in AS/HS	0.1
γ_1	Fixed shuffling time interval used in FS (hour)	24
γ_2	Fixed shuffling time interval (maximum delay) used in HS (hour)	120
λ	Mean value used for exponential distribution in RS (hour)	24

Our proposed NTS-MTD technique is implemented based on the workflow shown in Fig. 1. The `Optimization Module` implements the algorithms to execute the three "how" strategies, i.e., RNT, GANT, and DPNT, as discussed in Section 5.2.

We assume that there is an attacker exploiting node vulnerabilities. The vulnerability exploitation attack is implemented via the HARM model that computes potential attack paths. In each simulation run, the attacker will randomly choose an entry point from one attack path and compromise nodes along the attack path with behaviors defined in Section 3.3 until either SFC1 or SFC2 (see Section 3.5) is met. For each node to be compromised, we implemented the behavior of the attacker based on two steps: (1) check the privilege of the vulnerability; and (2) add the mean vulnerability exploitation time to MTTC if the required privilege is lower than what the attacker has (i.e., the exploitation of a vulnerability requires no authorization or user/administrator/root privilege). The attacker's intelligence, estimated by P_d^{em} and P_d^{OS} (see Section 3.4), is incorporated into the calculation of MTTC as well as MTTSF. We assume the system will clear decoy nodes once the intelligence center detects the attacker's interaction with the decoy target. Therefore, the attacker will not recognize the same decoy node in its subsequent action. Decoy nodes are also cleared at each shuffling. This is implemented by only marking the compromised real nodes as being compromised in the attack paths. By using FS/RS strategies, the network may be shuffled periodically or randomly right at the moment a node is under attack. We assume that the attacker is forced to quit the network due to lost connections and needs to find other ways to break into the network. This is implemented by checking the shuffling time interval with the mean vulnerability exploitation time of the node under compromise and forcing the attacker to randomly choose another entry point among the attack paths if the interval is met. In the subsequent attack after shuffling, the attacker could continue its previous attack action once it encounters the same real node next time (i.e., MTTC for the real node is accumulated throughout the MTTSF). By using the AS strategy, the network is shuffled due to changes to *SSV* being detected by the defender. The attacker is also forced to quit the network after each shuffling due to lost connections and needs to find ways to re-enter the network. Each newly shuffled network is modeled by a new HARM model for calculating potential attack paths. We encode each shuffling solution for the whole network as a binary valued vector with 1 representing the existence of an edge between two nodes and 0 representing no edge. We limit potential connections to be edges from real IoT nodes to either decoy nodes or real IoT nodes. Hence, to optimize the defense cost, we aim to maximize $C_T(t) - C_D(t)$ where $C_T(t)$ refers to the total defense cost (i.e., the total number of potential edge changes at time t) and $C_D(t)$ is the number of edges changed by executing NTS-MTD at time t (see Section 4.4).

For GANT, we aim to solve a multi-objective optimization (MOO) problem with three objectives to maximize N_{DN}^{AP} and MTTSF while minimizing C_D (or maximizing $C_T(t)-C_D(t)$). The optimization problem is to compute a set of Pareto optimal solutions (or Pareto frontier) [Cho et al. 2017b]. In order to choose one optimal solution among the Pareto frontier, we first normalize three metrics, denoted by $\widetilde{N_{DN}^{AP}}$, $\widetilde{MTTSF}$ and, $\widetilde{C_D}$, and then assign a weight to each metric based on scalarization-based MOO technique to transform the MOO problem to a single-objective optimization (SOO) problem [Cho et al. 2017a]. The normalized metric, $\tilde{X}$, is given by:

$$\tilde{X} = \frac{X}{X_{max}} \tag{7}$$

where X is the original metric value and X_{max} is the maximum metric value of the corresponding fitness function in the final population in the GA-based algorithm.

The objective function we aim to maximize is represented by:

$$\max \quad w_N \widetilde{N_{DN}^{AP}} + w_M \widetilde{MTTSF} + w_C \widetilde{C_D} \tag{8}$$

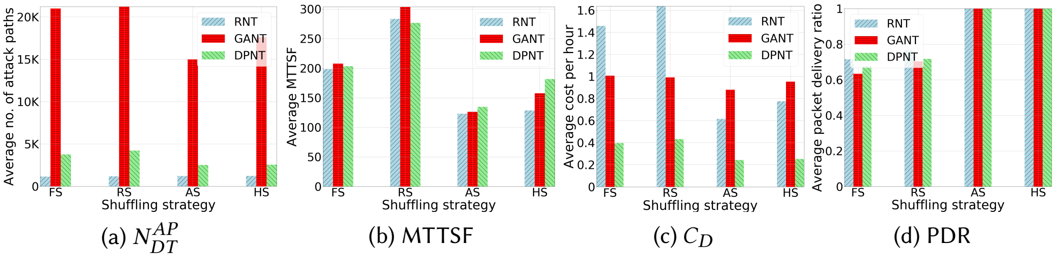
where w_N, w_M, and w_C are weights to the three metrics with $w_N + w_M + w_C = 1$. The optimal solution is the network topology with the maximum objective value.

In each simulation run, we collect data to calculate the mean time to security failure, MTTSF, the number of attack paths toward decoy targets, N_{DT}^{AP}, the defense cost per time unit, C_D, and the packet delivery ratio, PDR. We run the simulation 100 times using random seeds in each simulation. After 100 runs, we collect the means of MTTF, N_{DT}^{AP}, C_D, and PDR for performance analysis.

5.4 Comparative Performance Analysis

In this section, we conduct a comparative performance analysis of the 12 schemes discussed in Section 5.2. We follow the parameter table in Table 3. We vary the level of attackers' intelligence in detecting decoy nodes (i.e., P_d^{em} and P_d^{os}), attack severity (i.e., packet drop probability P_a^d and packet manipulation probability P_a^m), the number of decoys in each VLAN, and the number of real IoT nodes to analyze their effects on performance in terms of the mean time to security failure, MTTSF, the number of attack paths toward decoy targets, N_{DT}^{AP}, the defense cost per time unit, C_D, and the packet delivery ratio, PDR.

5.4.1 Comparison of Schemes under the Baseline Scenario. We first consider a baseline scenario in which there is only one decoy in each VLAN and the attacker intelligence is low characterized by its high interaction probabilities with decoys, i.e., P_d^{em}=0.9 for an emulated decoy and P_d^{os}=1.0 for a full-OS based decoy. Recall that a high interaction probability means that the attacker must interact with a decoy node intensively in order to detect it is a decoy.



(a) N_{DT}^{AP} (b) MTTSF (c) C_D (d) PDR

Fig. 3. Comparison of schemes under the baseline scenario.

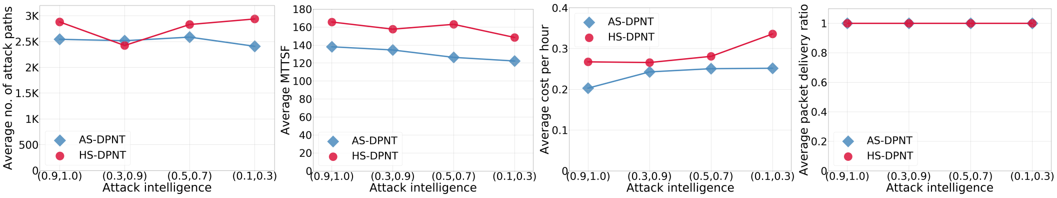
Fig. 3 compares the performance characteristics of the 12 schemes discussed in Section 5.2 for executing our proposed NTS-MTD technique under the default parameters presented in Table 3.

- Fig. 3a compares the number of attack paths toward decoy targets N_{DT}^{AP} (the higher the better) representing deception effectiveness. In the "when-to-shuffle" category, fixed/random shuffling (FS/RS) based schemes perform comparably among themselves. On the other hand, in the "how-to-shuffle" category, the genetic algorithm network topology (GANT) scheme performs the best in deception effectiveness, followed by the decoy path-optimized network topology (DPNT) scheme and the random network topology (RNT) scheme. This indicates how-to-shuffle the network has a major impact on deception effectiveness.
- Fig. 3b compares MTTSF (the higher the better) representing the system lifetime before the system experiences a failure. In the "when-to-shuffle" category, fixed/random shuffling (FS/RS) based schemes significantly outperform adaptive/hybrid shuffling (AS/HS) based schemes in MTTSF. One factor is system failure in AS/HS is determined by either SFC1 or SFC2 being triggered, or SSV exceeding the threshold. In the current setting, AS/HS uses a low SSV threshold (i.e., 0.1), which indicates a low tolerance on SFC1 and SFC2. This means system status could be considered as failure based on SSV threshold before either SFC1 or SFC2 is triggered. Another factor is that in FS/RS based schemes, a node may be under attacks while the network is shuffled because the fixed/random interval for topology shuffling could be much smaller than the MTTC of the node at which time topology shuffling is triggered by AS/HS. After each shuffling, the attacker is forced to quit the network due to lost connections and needs to re-enter the network by randomly choosing entry points to compromise. After re-entering, the attacker could continue its previous attack once it encounters the same real node next time or launch a new attack for a decoy node as decoys are cleared at each shuffling. This could effectively lead to an increase of MTTSF over time in order to meet either SFC1 or SFC2 security failure condition. We see that RS produces the highest MTTSF among all. In the "how-to-shuffle" category, GANT and DPNT perform comparably among themselves and both outperform RNT.
- Fig. 3c compares the defense cost C_D (the lower the better). Since the defense cost is inversely related to the number of attack paths toward decoy targets (i.e., deception effectiveness), we expect the trend for defense cost is just opposite to that in Fig. 3a for deception effectiveness. This is indeed the case. In the "when-to-shuffle" category, adaptive/hybrid shuffling (AS/HS) based schemes perform comparably among themselves and outperform fixed/random shuffling (FS/RS) based schemes, a trend that is opposite to that for deception effectiveness. In the "how-to-shuffle" category, DPNT performs the best in defense cost among all, followed by GANT and RNT. This is also a trend that is in line with that exhibited in Fig. 3a for deception effectiveness. DPNT has the lowest C_D among all due to less edge changes made during topology shuffling compared to GANT and RNT.
- Fig. 3d compares packet delivery ratio PDR (the higher the better) representing service availability. In the "when-to-shuffle" category, adaptive/hybrid shuffling (AS/HS) based schemes perform comparably among themselves and outperform fixed/random shuffling (FS/RS) based schemes. The reason is that AS/HS produces a smaller number of attack paths toward decoy targets than

FS/RS, so the attacker has a smaller chance to drop or manipulate packets passing through the attack paths. In the "how-to-shuffle" category, GANT, DPNT and RNT perform comparably among themselves.

Summarizing above, there is no winner that can achieve the goal of maximizing deception effectiveness (see Fig. 3a), MTTSF (see Fig. 3b), and service availability (see Fig. 3d) while minimizing defense cost (see Fig. 3c). However, we could identify DPNT as the best "how-to-shuffle" strategy that can maximize MTTSF and minimize defense cost, while maintaining comparable service availability. We explore optimal parameters of DPNT-based schemes in Section 5.5 and compare the IoT network with and without these DPNT-based schemes in Section 5.6. We also note that RS performs better than FS even the mean time interval for executing topology shuffling in RS is the same as the fixed time interval for executing topology shuffling in FS (see Table 3). We attribute this to the fact that the execution time interval in RS follows exponential distribution and this stochastic nature matches better with the stochastic nature of attack behavior.

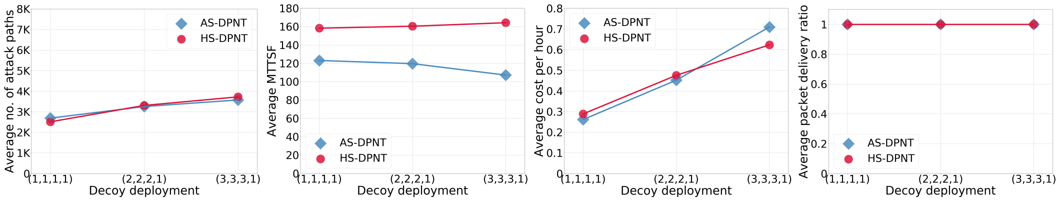
5.4.2 *Analysis on Impact of Attacker's Intelligence.* We use the baseline scenario and consider the attackers can exhibit different levels of intelligence. We consider three levels of attack intelligence represented by four pairs of interaction probabilities with decoys (P_d^{em} for an emulated decoy, P_d^{os} for a full-OS based decoy): low intelligence (0.9, 1.0), medium intelligence (0.3, 0.9), medium intelligence (0.5, 0.7), and high intelligence (0.1, 0.3). We consider two cases of medium intelligence: the case 1 is for a medium-intelligence attacker that can easily recognize an emulated decoy but can hardly recognize a full OS-based decoy and the case 2 is for a medium-intelligence attacker that can only modestly recognize an emulated decoy or a full OS-based decoy, respectively. For other design parameters, we follow their default values summarized in Table 3. Without loss of generality, we consider AS-DPNT and HS-DPNT to analyze the impact of attack intelligence.



(a) N_{DT}^{AP} (b) MTTSF (c) C_D (d) PDR

Fig. 4. Performance analysis on impact of an attacker's intelligence.

Fig. 4 shows how AS-DPNT and HS-DPNT perform in terms of the mean time to security failure, MTTSF, the number of attack paths toward decoy targets, N_{DT}^{AP}, the defense cost per time unit, C_D, and the packet delivery ratio, PDR. In Fig. 4a, with the decreasing attack intelligence, N_{DT}^{AP} fluctuates for each scheme as this metric is related to the shuffling algorithm (i.e., DPNT in this case study). In Fig. 4b, for each scheme, MTTSF reaches the highest when the attacker has low intelligence. This implies the potential attacker with higher intelligence in detecting decoys hurts the system lifetime as measured based on MTTSF. However, both AS-DPNT and HS-DPNT are resilient under high-intelligent attacks without much reduction of MTTSF compared with the case of low-intelligent attacks. In Fig. 4c, C_D has an increasing trend for both schemes when intelligence increases. In Fig. 4d, PDR remains at 1.0 for both schemes. One reason is that in adaptive/hybrid shuffling, critical nodes may not be compromised when the SSV threshold is small (e.g., $\rho = 0.1$). Even if some neighbor nodes are compromised, there are still some clean neighbor nodes to be able to deliver packets. Another reason is that in our simulation setting, $P_a^d = 0.5$ and $P_a^m = 0.5$ to avoid detection, so compromised nodes will only drop half of the packets passing through them.



(a) N_{DT}^{AP} (b) MTTSF (c) C_D (d) PDR

Fig. 6. Performance comparative analysis of the variants of DPNT schemes under different decoy deployment scenarios.

In summary, attack intelligence has a moderate degree of impact (10-20%) on MTTSF and C_D because high intelligent attackers are capable of detecting decoys early on. This allows them to have more interactions with real nodes early on, thereby leading to shorter lifetime and forcing the system to trigger costly shuffling operations to prevent security attacks. Attack intelligence, however, has little impact on N_{DT}^{AP} and PDR.

5.4.3 Analysis on Impact of Attack Severity on Service Availability. We use the baseline scenario, except considering attacks with different levels of severity that would affect service availability. We consider three levels of attack severity represented by three pairs of packet drop probability P_a^d and packet manipulation probability P_a^m: low severity (0.1, 0.1), medium severity (0.5, 0.5) and high severity (1.0, 1.0). For other design parameters, we follow their default values summarized in Table 3. We again apply DPNT-based schemes to analyze the impact of attack severity on packet delivery ratio (PDR) representing service availability.

Fig. 5 shows the effect attack severity on PDR for DPNT based schemes. We observe that PDR remains at 1.0 for AS-DPNT and HS-DPNT while steadily decreases for FS-DPNT and RS-DPNT as the attack severity increases. This demonstrates resilience of adaptive shuffling schemes (i.e., AS/HS) in response to increasing attack severity because critical nodes are well protected from security attacks by setting a low *SSV* threshold (e.g., $\rho = 0.1$).

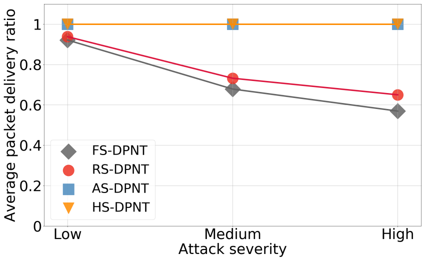


Fig. 5. Comparative performance analysis of the variants of DPNT schemes under the different attack severity.

5.4.4 Analysis on Impact of Decoy Node Population. We increase the number of decoy nodes in each VLAN to analyze the impact of decoy node population. The baseline scenario has (1, 1, 1, 1) decoy nodes for (CT scan, smart camera, smart TV, server). We consider two more scenarios: (2, 2, 2, 1) and (3, 3, 3, 1) where random connections among decoy IoT nodes are also added. For other design parameters, we follow their default values summarized in Table 3. We consider AS-DPNT and HS-DPNT to analyze the impact of decoy population.

Fig. 6 shows how AS-DPNT and HS-DPNT perform in terms of the mean time to security failure, MTTSF, the number of attack paths toward decoy targets, N_{DT}^{AP}, the defense cost per time unit, C_D, and the packet delivery ratio, PDR, as the decoy population changes. In Fig. 6a, N_{DT}^{AP} increases slightly with the increasing number of decoys within each scheme. The reason is that as the number of decoys increases, DPNT also increases the number of attack paths toward the decoy target. In Fig. 6b, MTTSF remains steady as the number of decoys increases. We attribute this to the design of DPNT algorithm which only focuses on maximizing the number of decoy paths. There may be many paths with a majority of real IoT nodes on the paths. Attackers could still be able to compromise a large portion of real IoT nodes thus leading to system failure because MTTSF is calculated based on compromised real nodes within the network. In Fig. 6c, C_D also increases as the number of decoys increases. This is due to the fact that a lot of edges need to be changed with additional decoy nodes. In Fig. 6d, PDR remains at 1.0 across all scenarios as PDR is related to service availability among real nodes, so it is little affected by decoys especially when SSV threshold is low (e.g., $\rho = 0.1$ in our test case). In summary, decoy node population impacts N_{DT}^{AP} and C_D while it does not largely improve MTTSF or PDR.

5.4.5 Analysis on Impact of Network Size. We increased the number of nodes in each VLAN to analyze the impact of network size. The baseline scenario has (2, 3, 2, 1) real IoT nodes for (MRI/CT scan, smart thermostat/meter/camera, smart TV/laptop, server) and (1, 1, 1, 1) decoy nodes for (CI scan, smart camera, smart TV, server). We consider two more scenarios: (4, 300, 200, 1) real nodes and (2, 50, 50, 1) decoy nodes vs. (4, 1200, 800, 1) real nodes and (2, 100, 100, 1) decoy nodes respectively. The number of medical devices in VLAN1 is kept in a low number as they are rarely deployed in a large scale due to their high price. Therefore, we have three scenarios with the number of real nodes as 8, 505, and 2005, respectively. We apply the HS-DPNT algorithm to analyze the impact of the network size with default parameters specified in Table 3. To reduce the computational complexity of attack paths, we constrained the out-degree of a real IoT node (i.e., the number of outgoing connections to other real nodes) and the maximum path length in the DPNT algorithm. We do not set any constraint for the baseline scenario while using 2 as the maximum outgoing connections and 5 as the maximum path length for the other two scenarios.

Table 4. Analysis on impact of network size.

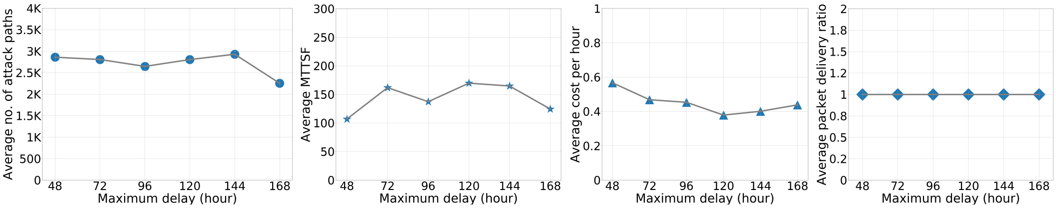
Metric	No. of nodes (real, decoy)		
	(8, 4)	(505, 103)	(2005, 203)
N_{DT}^{AP}	2586.9	2822521.4	88128332.3
MTTSF	145.0	2767.1	11297.9
C_D	0.28	84.2	602.7
PDR	1.0	0.97	0.97

Table 4 shows the effect of network size on performance in terms of the mean time to security failure, MTTSF, the number of attack paths toward decoy targets, N_{DT}^{AP}, the defense cost per time unit, C_D, and the packet delivery ratio, PDR. We see that N_{DT}^{AP} has a significant jump when the network size increases while MTTSF grows relatively steadily. As the number of real and decoy IoT nodes increases, the number of decoy paths with real IoT nodes acting as entry points and intermediate nodes increases and more decoy paths/edges are also created. C_D rises dramatically from the baseline scenario to the second scenario and then increases gradually in the third scenario. Lastly, the network size has little effect on Packets Delivery Ratio (PDR). The reason is that more real nodes introduce more paths towards the real target but more decoy nodes also introduce more paths towards the decoy target even if some real nodes along paths towards the real target can be compromised. In summary, network size has a high impact on N_{DT}^{AP}, MTTSF, and C_D, but little impact on PDR.

5.5 Sensitivity Analysis

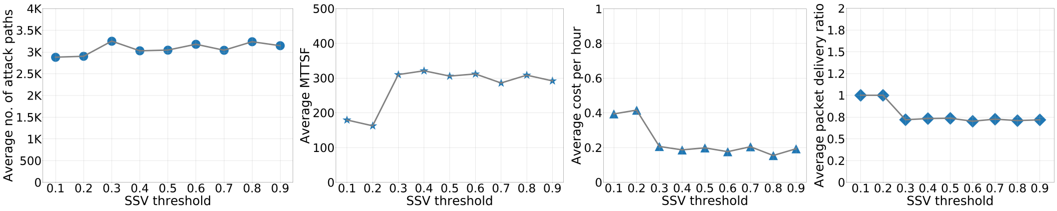
In this section, we examine the sensitivity of the performance results with respect to the maximum delay parameter (γ_2) and the security vulnerability level (SSV) threshold parameter (ρ) to identify the optimal parameter setting under which the system performance can be maximized. These two parameters are used in two "when-to-shuffle" strategies, namely, adaptive shuffling and hybrid shuffling (AS/HS). Without loss of generality, we consider HS-DPNT in the sensitivity analysis since earlier we have identified DPNT as the best "how-to-shuffle" strategy that can maximize MTTSF and minimize defense cost, while maintaining comparable service availability.

5.5.1 Sensitivity Analysis of Maximum Delay. We use the baseline scenario in Section 5.4.1, except that we vary the maximum delay parameter, γ_2, when performing hybrid shuffling. The reason we use a maximum delay is to avoid the situation in which an incremental increase of SSV does not



(a) N_{DT}^{AP} (b) MTTSF (c) C_D (d) PDR

Fig. 7. Effect of the maximum delay (γ_2) on the performance of HS-DPNT (identified as the best scheme) under the baseline scenario.



(a) N_{DT}^{AP} (b) MTTSF (c) C_D (d) PDR

Fig. 8. Effect of the system security vulnerability (SSV) threshold ρ on the performance of HS-DPNT (identified as the best scheme) under the baseline scenario.

reach the threshold, thus delaying the execution of DPNT. We consider the following values for γ_2 in the sensitivity analysis: 48, 72, 96, 120, 144, 168 (hours). These values are related to the scenario and could change due to different scenarios. For other design parameters, we follow their default values summarized in Table 3.

Fig. 7 shows the sensitivity of the performance results in terms of N_{DT}^{AP}, MTTSF, C_D, and PDR with respect to the maximum delay parameter (γ_2) in hybrid shuffling (HS). Intuitively, a shorter delay may cause the network to be shuffled more often which makes HS similar to fixed/random shuffling (FS/RS) while a longer delay may delay shuffling thus degenerating HS to adaptive shuffling (AS). In Fig 7a, N_{DT}^{AP} fluctuates as the metric is related to how-to-shuffle instead of when-to-shuffle. In Fig. 7b, MTTSF fluctuates slightly from 48 to 120 with a local optimal value of 162 hours at 72, reaches the peak of 170 hours at 120 and then drops to 124 hours at 168. In Fig. 7c, C_D has the opposite trend of MTTSF (the higher the frequency of shuffling and longer MTTSF, the lower the cost per hour). In Fig. 7d, PDR remains at 1.0. The reason is the same as stated in Section 5.4.2. Summarizing above, if the goal is to maximize MTTSF, setting the maximum delay at 120 could be considered as optimal for HS-DPNT.

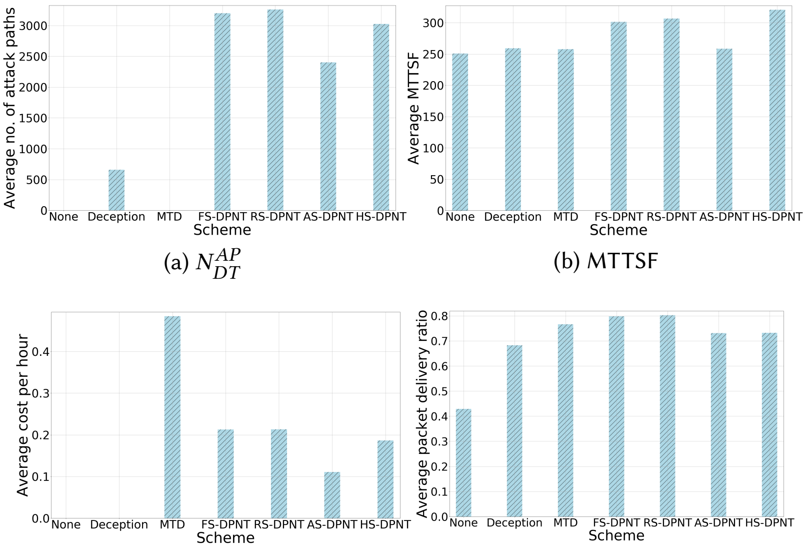
5.5.2 *Sensitivity Analysis of the System Security Vulnerability (SSV) Threshold.*

We use the baseline scenario in Section 5.4.1, except that we vary the SSV threshold values, ρ, in the range of [0.1, 0.9] with 0.1 as the increment. For other design parameters, we follow their default values summarized in Table 3. We again apply HS-DPNT in our sensitivity analysis.

Fig. 8 shows the sensitivity of the performance results in terms of N_{DT}^{AP}, MTTSF, C_D, and PDR with respect to the SSV threshold parameter ρ. In Fig. 8a, N_{DT}^{AP} fluctuates as N_{DT}^{AP} is related to how-to-shuffle instead of when-to-shuffle. In Fig. 8b, MTTSF jumps from 163 at 0.2 to 310 at 0.3, slightly increases to 321 as the peak when ρ is 0.4, and then varies between 286 and 312 when ρ increases. In Fig. 8c, C_D decreases to 0.21 when ρ increases to 0.3 and then stays stable with increasing ρ. In Fig. 8d, PDR drops to 0.72 when ρ is 0.3 and stays stable afterwards. The reason is the same as stated in Section 5.4.2. Summarizing above, if the goal is to maximize MTTSF, setting the SSV threshold parameter at 0.4 could be considered as optimal for HS-DPNT.

5.6 Performance Comparison of IoT Networks with vs. without NTS-MTD Running and/or Decoy Deployment

In this section, we compare the performance of IoT networks with vs. without our proposed network topology shuffling-based MTD (NTS-MTD) technique running and/or decoy deployment. That is, the baseline IoT network has no decoy nodes deployed and no proposed network topology shuffling-based MTD (NTS-MTD) technique running for intrusion prevention, referred to as no defense. We also consider two single defense schemes deployed on the baseline IoT network: (1) only decoy nodes deployed and connections from real nodes to decoy nodes being randomly added upon the decoy deployment, referred to as only deception; (2) only NTS-MTD technique applied via random shuffling algorithm with a fix interval, referred to as only MTD. We use the same baseline scenario as before with the same attack model applied. We collected performance data for computing N_{DT}^{AP}, MTTSF, C_D, and PDR based on 100 times of simulation runs. We computed the optimal fix interval at 48 hours for only MTD scheme and compare the baseline, only deception, and only MTD schemes with DPNT based schemes running at optimal settings identified in the sensitivity analysis study of Section 5.5 (i.e., FS-DPNT with the optimal fixed interval at 72 hours, RS-DPNT with the optimal mean interval at 72 hours, AS-DPNT with the optimal SSV threshold at 0.3, and HS-DPNT with the optimal SSV threshold at 0.4).



(a) N_{DT}^{AP}

(b) MTTSF

(c) C_D

(d) PDR

Fig. 9. Performance Comparison of IoT Networks with vs. without NTS-MTD Running.

Fig. 9 shows the performance comparison results in N_{DT}^{AP}, MTTSF, C_D, and PDR over all DPNT based schemes considered in this work. The baseline IoT system is labeled with “No defense” and the other two single defense schemes are labeled with “Only deception” and “Only MTD” in Fig. 9. We observe that all DPNT-based schemes significantly outperform the counterpart baseline IoT system regarding N_{DT}^{AP}, MTTSF, and PDR. In particular, HS-DPNT has the highest increase (28%) in MTTSF while FS-DPNT has the highest increase (59%) in PDR. All DPNT-based schemes incur much higher N_{DT}^{AP} compared with only deception scheme and maintain much lower cost compared with only MTD scheme. All DPNT-based schemes incur higher MTTSF compared with single defense schemes while FS-DPNT and RS-DPNT have highest PDR. These results demonstrate that an IoT network running our intrusion prevention technique at the optimal parameter setting prolongs system lifetime, increases attack complexity of compromising critical nodes (so that the system lifetime is prolonged), and maintains good service availability compared with a counterpart baseline IoT network without running our intrusion prevention technique and single defense schemes.

6 DISCUSSION & FUTURE WORK

In this section, we discuss the reproduciblility of the simulation results, limitations of the proposed integrated mechanism, possible evasion techniques by attackers, and present the future work.

Reproducibility of results: Simulation results are reproducible using the simulation code on a specified platform. There will be a variation of the values within a reasonable range by: (1) probability of random shuffling in RNT (i.e., P_r); (2) random generation of an initial population in GANT; (3) random generation of connections from real IoT nodes to decoy nodes in DPNT; (4) random selection of entry points by the attacker at each shuffling and each simulation run; and (5) interaction probabilities with decoy nodes (i.e., P_d^{em} and P_d^{os}).

Scalability: We imposed two constraints on the DPNT algorithm (i.e., the maximum outgoing connections and path length) to limit the complexity of an attack path calculation in the HARM and conducted simulations on networks with a maximum of two thousand nodes. The growing network size (to tens of thousands of nodes or more) will introduce higher computational overhead and memory consumption. This can be reduced by dividing the network into sub-networks and adopting distributed MTD operations discussed in Section 7.

Decoy deployment: We applied one full OS-based server decoy and emulated IoT decoys in the example IoT network. A full OS-based decoy may require additional hardware to replicate the production service while an emulation-based decoy runs in the virtual machine. Therefore, in reality, the deployment of full OS-based decoys can be restricted by the budget and actual network configurations. We have completed a preliminary research on the optimal deployment of patch and decoys by considering decoy cost in [Ge et al. 2018]. This can be integrated with the proposed mechanism to explore the optimal topology via shuffling under environments with various decoys and cost constraints or their combinations.

Applicability of SDN: We discussed different SDN solutions applied to IoT networks in Section 2.3. However, SDN is still an emerging technology in network management for IoT and not applicable in all application domains of IoT (e.g., smart home) due to the additional cost of SDN controller and switches to replace traditional network devices.

Implication on results: We use SDN technology to simulate the example IoT network to which we apply the proposed defense mechanisms. The network shuffling-based MTD can be integrated with the functionality of open source SDN controller as an application. We consider the testbed development with MTD and deception integration as future work discussed in Section 7. The simulation results demonstrated the optimal combination among "when-to-shuffle" and "how-to-shuffle" strategies and implied a promising security improvement achieved through MTD and deception.

Evasion techniques by attackers: We designed and implemented adaptive/hybrid shuffling (AS/HS) strategies under the condition that each shuffling is triggered by detecting compromised real nodes by the IDS. Any IDS evasion techniques by intelligent attackers (e.g., APT attacks) can lead to evasion of triggering AS/HS. Therefore, hybrid shuffling is recommended to enable shuffling upon a maximum delay even if the IDS evasion is successful.

As our future work, we plan to explore the following research areas: (1) setup of a cloud-based testbed with virtual devices to simulate IoT behaviors and virtual decoys where an SDN-based environment is considered by leveraging an open source SDN controller to support packet control and virtual switches to perform packet forwarding actions and the network shuffling-based MTD as an application is integrated with the SDN controller; (2) development of distributed MTD operations with decentralized SDN controllers through dividing an IoT network into multiple sub-networks which can be controlled by different SDN controllers with the aim of providing lightweight shuffling-based MTD solutions; (3) investigation of machine/deep learning-based approaches to compute an optimal network topology in network shuffling-based MTD (e.g., graph neural networks to model complex relationships and learn information structured as graphs [Rusek et al. 2019]); and (4) incorporation of machine/deep learning-based network topology generation technology with the graphical security model (GSM) to determine the optimal network topology by reconstructing GSM and developing new security metrics for solution optimization (e.g., the average number of decoy nodes on an attack path).

7 CONCLUSIONS

In this paper, we proposed an integrated proactive defense mechanism by utilizing cyberdeception and network topology shuffling and completed a comprehensive analysis via simulation. We considered a smart hospital scenario within the IoT context. The proposed approach could be applied to any IoT environment. From this study, we obtained the following **key findings**:

- In the "when-to-shuffle" category, adaptive/hybrid shuffling (AS/HS) based schemes outperform fixed/random shuffling (FS/RS) based schemes in defense cost. On the contrary, FS/RS based schemes outperform AS/HS based schemes in the average number of attack paths toward decoy targets (i.e., deception effectiveness). Choices of fixed/mean interval used by FS/RS and SSV threshold used by AS/HS have significant impact on MTTSF and service availability and need to be properly determined. The analysis performed in this paper can help the system designer determine the best interval by FS/RS and best SSV threshold by AS/HS to maximize MTTSF.
- In the "how-to-shuffle" category, decoy path-optimized network topology (DPNT) based schemes perform comparably with genetic algorithm network topology (GANT) based schemes in MTTSF (i.e., system lifetime) and packet delivery ratio. On the other hand, DPNT incurs less defense cost than GANT since GANT tends to create more attack paths toward decoy targets (i.e., deception effectiveness). Both DPNT and GANT based schemes outperform random network topology (RNT) shuffling schemes in MTTSF and the number of attack paths toward decoy targets (deception efficiency). Consequently, if MTTSF is the goal, DPNT should be chosen over GANT because it incurs less defense cost while achieving comparable MTTSF.
- If maximizing MTTSF is the most important goal, while maximizing deception effectiveness and service availability and minimizing defense cost are sub-goals, HS-DPNT with an optimal SSV threshold (with HS as the "when-to-shuffle" strategy and DPNT as the "how-to-shuffle" strategy) emerges as the best scheme among the 12 schemes investigated for executing our proposed NTS-MTD technique because it can maximize MTTSF (even the number of attack paths toward decoy targets generated by DPNT is low) and minimize defense cost, while maintaining comparable service availability.
- Among the 12 schemes investigated for executing our proposed NTS-MTD technique, AS-DPNT/HS-DPNT (with AS/HS as the "when-to-shuffle" strategy and DPNT as the "how-to-shuffle" strategy) can achieve high MTTSF and deception effectiveness, while maintaining low defense cost and high service availability. Further, AS-DPNT/HS-DPNT are resilient against attackers with increasing intelligence capability of detecting decoy nodes. There exist an optimal setting for the system security vulnerability level threshold parameter and the maximum delay parameter for maximizing MTTSF. The analysis performed in this paper can help the system designer identify the best parameter setting under which MTTSF may be maximized.

REFERENCES

H. Abie and I. Balasingham. 2012. Risk-based Adaptive Security for Smart IoT in eHealth. In *Proceedings of the 7th International Conference on Body Area Networks (BodyNets '12).* ICST, 269–275.

H. Alavizadeh, D. S. Kim, and J. Jang-Jaccard. 2020. Model-based evaluation of combinations of Shuffle and Diversity MTD techniques on the cloud. *Future Generation Computer Systems* 111 (2020), 507–522.

A. Almohaimeed, S. Gampa, and G. Singh. 2019. Privacy-Preserving IoT Devices. In *Proceedings of the 2019 IEEE Long Island Systems, Applications and Technology Conference (LISAT '19).* 1–5.

M. Anirudh, S. A. Thileeban, and D. J. Nallathambi. 2017. Use of honeypots for mitigating DoS attacks targeted on IoT networks. In *Proceedings of the 2017 International Conference on Computer, Communication and Signal Processing (ICCCSP '17).* IEEE, 1–4.

C. J. Bernardos, A. de la Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zuniga. 2014. An Architecture for Software Defined Wireless Networking. *IEEE Wireless Communications* 21, 3 (2014), 52–61.

V. Casola, A. D. Benedictis, and M. Albanese. 2013. A Multi-Layer Moving Target Defense Approach for Protecting Resource-Constrained Distributed Devices. In *IRI.*

J. Cho, D. P. Sharma, H. Alavizadeh, S. Yoon, N. Ben-Asher, T. J. Moore, D. S. Kim, H. Lim, and F. F. Nelson. 2020. Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense. *IEEE Communications Surveys Tutorials* (2020), 1–1.

J. H. Cho and N. Ben-Asher. 2018. Cyber defense in breadth: Modeling and analysis of integrated defense systems. *The Journal of Defense Modeling and Simulation* 15, 2 (2018), 147–160.

J. H. Cho, Y. Wang, I. R. Chen, K. S. Chan, and A. Swami. 2017a. A Survey on Modeling and Optimizing Multi-Objective Systems. *IEEE Communications Surveys Tutorials* 19, 3 (2017), 1867–1901.

J. H. Cho, Y. Wang, R. Chen, K. S. Chan, and A. Swami. 2017b. A survey on modeling and optimizing multi-objective systems. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1867–1901.

B. T. De Oliveira, L. B. Gabriel, and C. B. Margi. 2015. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. *IEEE Latin America Transactions* 13, 11 (2015), 3690–3696.

S. Dowling, M. Schukat, and H. Melvin. 2017. A ZigBee Honeypot to assess IoT cyberattack behaviour. In *Proceedings of the 2017 28th Irish Signals and Systems Conference (ISSC '17).* IEEE, 1–6.

Open Network Foundation. 2012. *OpenFlow Switch Specification (Version 1.3.0).* Technical Report.

L. Galluccio, S. Milardo, G. Morabito, and S. Palazzo. 2015. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for WIreless SEnsor networks. In *Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM '15).* 513–521.

F. C. Gärtner. 2003. *Byzantine failures and security: Arbitrary is not (always) random.* Technical Report.

M. Ge. 2020. IoT_IntegratedDefence. Retrieved 2020-08-10 from https://github.com/mmge88/IoT_IntegratedDefence

M. Ge, J. Cho, C. A. Kamhoua, and D. S. Kim. 2018. Optimal Deployments of Defense Mechanisms for the Internet of Things. In *2018 International Workshop on Secure Internet of Things (SIoT).* IEEE, 8–17.

M. Ge, J. H. Cho, B. Ishfaq, and D. S. Kim. 2020. *Modeling and Design of Secure Internet of Things.* Wiley, Chapter Modeling and Analysis of Proactive Defense Mechanisms for Internet-of-Things. IEEE Press.

M. Ge, J. B. Hong, W. Guttmann, and D. S. Kim. 2017. A framework for automating security analysis of the internet of things. *Journal of Network and Computer Applications* 83 (2017), 12–27.

M. Ge, J. B. Hong, S. E. Yusuf, and D. S. Kim. 2018. Proactive defense mechanisms for the software-defined Internet of Things with non-patchable vulnerabilities. *Future Generation Computer Systems* 78 (2018), 568–582.

A. O. Hamada, M. Azab, and A. Mokhtar. 2018. Honeypot-like Moving-target Defense for secure IoT Operation. In *Proceedings of the 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON '18).* 971–977.

J. B. Hong and D. S. Kim. 2015. Assessing the Effectiveness of Moving Target Defenses using Security Models. *IEEE Transactions on Dependable and Secure Computing* 13, 2 (2015), 163–177.

Jin B. Hong, Dong Seong Kim, Chun-Jen Chung, and Dijiang Huang. 2017. A survey on the usability and practical applications of Graphical Security Models. *Computer Science Review* 26 (2017), 1–16.

M. Kahla, M. Azab, and A. Mansour. 2018. Secure, Resilient, and Self-Configuring Fog Architecture for Untrustworthy IoT Environments. In *Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE '18).* 49–54.

A. I. Kouachi, S. Sahraoui, and A. Bachir. 2018. Per Packet Flow Anonymization in 6LoWPAN IoT Networks. In *Proceedings of the 2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM '18).* 1–7.

Q. D. La, T. Q. S. Quek, J. Lee, S. Jin, and H. Zhu. 2016. Deceptive Attack and Defense Game in Honeypot-Enabled Networks for the Internet of Things. *IEEE Internet of Things Journal* 3, 6 (2016), 1025–1035.

T. Lei, Z. Lu, X. Wen, X. Zhao, and L. Wang. 2014. SWAN: An SDN Based Campus WLAN framework. In *Proceedings of the 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace*

Electronic Systems (VITAE '14). 1–5.

G. Lin, M. Dong, K. Ota, J. Li, W. Yang, and J. Wu. 2019. Security Function Virtualization Based Moving Target Defense of SDN-Enabled Smart Grid. In *Proceedings of the 2019 IEEE International Conference on Communications (ICC '19)*. 1–6.

J. Liu, Y. Li, M. Chen, W. Dong, and D. Jin. 2015. Software-Defined Internet of Things for Smart Urban Sensing. *IEEE Communications Magazine* 53, 9 (2015), 55–63.

Y. Liu, G. Grigoryan, C. A. Kamhoua, and L. L. Njilla. 2020. *Modeling and Design of Secure Internet of Things*. Wiley, Chapter Leverage SDN for CyberâĂŘSecurity Deception in Internet of Things. IEEE Press.

K. Mahmood and D. M. Shila. 2016. Moving target defense for Internet of Things using context aware code partitioning and code diversification. In *Proceedings of the 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT '16)*. IEEE, 329–330.

T. Miyazaki, S. Yamaguchi, K. Kobayashi, J. Kitamichi, Song Guo, T. Tsukahara, and T. Hayashi. 2014. A Software Defined Wireless Sensor Network. In *Proceedings of the IEEE 2014 International Conference on Computing, Networking and Communications (ICNC '14)*. 847–852.

NIST. 2005. National Vulnerability Database (NVD). Retrieved 2019-08-20 from https://nvd.nist.gov/

F. Nizzi, T. Pecorella, F. Esposito, L. Pierucci, and R. Fantacci. 2019. IoT Security via Address Shuffling: The Easy Way. *IEEE Internet of Things Journal* 6, 2 (2019), 3764–3774.

L. Pingree. 2016. Emerging Technology Analysis: Deception Techniques and Technologies Create Security Technology Business Opportunities. Retrieved 2017-10-31 from https://www.gartner.com/doc/reprints?id=1-2LSQOX3&ct=150824&st=sb&aliId=87768

S. Plaga, N. Wiedermann, M. Niedermaier, A. Giehl, and T. Newe. 2018. Future Proofing IoT Embedded Platforms for Cryptographic Primitives Support. In *Proceedings of the 12th International Conference on Sensing Technology (ICST '18)*. 52–57.

R. Roman, J. Zhou, and J. Lopez. 2013. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks* 57, 10 (2013), 2266–2279.

A. Rullo, E. Serra, E. Bertino, and J. Lobo. 2017. Shortfall-Based Optimal Security Provisioning for Internet of Things. In *Proceedings of 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS '17)*. IEEE, 2585–2586.

K. Rusek, J. Suárez-Varela, A. Mestres, P. Barlet-Ros, and A. Cabellos-Aparicio. 2019. Unveiling the Potential of Graph Neural Networks for Network Modeling and Optimization in SDN. In *Proceedings of the 2019 ACM Symposium on SDN Research (SOSR '19)*. Association for Computing Machinery, 140–151. https://doi.org/10.1145/3314148.3314357

V. Saini, Q. Duan, and V. Paruchuri. 2008. Threat Modeling using Attack Trees. *Journal of Computer Science in Colleges* 23, 4 (2008), 124–131.

R. M. Savola, H. Abie, and M. Sihvonen. 2012. Towards Metrics-driven Adaptive Security Management in e-Health IoT Applications. In *Proceedings of the 7th International Conference on Body Area Networks (BodyNets '12)*. ICST, 276–281.

S. Sengupta, A. Chowdhary, A. Sabur, A. Alshamrani, Huang D., and S. Kambhampati. 2020. A Survey of Moving Target Defenses for Network Security. arXiv:1905.00964.

M. Sherburne, R. Marchany, and J. Tront. 2014. Implementing moving target ipv6 defense to secure 6lowpan in the internet of things and smart grid. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR '14)*. ACM, 37–40.

O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. 2002. Automated Generation and Analysis of Attack Graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (SP '02)*. IEEE Computer Society, 273–284.

S. Vuppala, A. E. Mady, and A. Kuenzi. 2019. Rekeying-based Moving Target Defence Mechanism for Side-Channel Attacks. In *Proceedings of the 2019 Global IoT Summit (GIoTS '19)*. 1–5.

S. Wang, H. Shi, Q. Hu, B. Lin, and X. Cheng. 2019. Moving Target Defense for Internet of Things Based on the Zero-Determinant Theory. *IEEE Internet of Things Journal* (2019), 1–1.

K. Zeitz, M. Cantrell, R. Marchany, and J. Tront. 2017. Designing a micro-moving target ipv6 defense for the internet of things. In *Proceedings of the 2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI '17)*. IEEE, 179–184.

K. Zeitz, M. Cantrell, R. Marchany, and J. Tront. 2018. Changing the Game: A Micro Moving Target IPv6 Defense for the Internet of Things. *IEEE Wireless Communications Letters* 7, 4 (2018), 578–581.