EasyChair Preprint

№ 3786

IANVS: A Moving Target Defense Framework for a Resilient Internet of Things

Renzo Navas, Håkon Sandaker, Frédéric Cuppens,
Nora Cuppens-Boulahia, Laurent Toutain and
Georgios Papadopoulos

July 7, 2020

IANVS: A Moving Target Defense Framework for a Resilient Internet of Things

Renzo E. Navas*, Håkon Sandaker§, Frédéric Cuppens*, Nora Cuppens*,
Laurent Toutain† and Georgios Z. Papadopoulos†

*Lab-STICC, UMR CNRS 6285, F-35700. †IRISA, UMR CNRS 6074, F-35700.
*†IMT Atlantique, Rennes, France. Email: {firstname.lastname}@imt-atlantique.fr
§University of Oslo, Oslo, Norway. Email: hasa@student.matnat.uio.no

***Abstract*—The Internet of Things (IoT) is more and more present in fundamental aspects of our societies and personal life. Billions of objects now have access to the Internet. This networking capability allows for new beneficial services and applications. However, it is also the entry-point for a wide variety of cyber-attacks that target these devices. The security measures present in real IoT systems lag behind those of the standard Internet. Security is sometimes completely absent. Moving Target Defense (MTD) is a 10-year-old cyber-defense paradigm. It proposes to randomize components of a system. Reasonably, an attacker will have a higher cost attacking an MTD-version of a system compared with a static-version of it. Even if MTD has been successfully applied to standard systems, its deployment for IoT is still lacking. In this paper, we propose a generic MTD framework suitable for IoT systems: IANVS (pronounced *Janus*). Our framework has a modular design. Its components can be adapted according to the specific constraints and requirements of a particular IoT system. We use it to instantiate two concrete MTD strategies. One that targets the UDP port numbers (port-hopping), and another a CoAP resource URI. We implement our proposal on real hardware using Pycom LoPy4 nodes. We expose the nodes to a remote Denial-of-Service attack and evaluate the effectiveness of the IANVS-based port-hopping MTD proposal.**

***Index Terms*—IoT, Security, Moving Target Defense, MTD, framework, design, stream-cipher, ChaCha20, port-hopping, attack, reconnaissance, LoPy4, hping3, CoAP**

I. Introduction

The Internet of Things (IoT) is composed of billions of connected devices that interact with the physical world. It lies at the frontier of the digital and physical realms and has applications in many aspects of modern societies like agriculture, industrial automation (Industry 4.0), cities, and houses. As of 2020, there are as many IoT devices as human beings (around 8 billion), but the annual growth for IoT is $\sim 15\%$ while for the population is 1% [1]. In five years, IoT devices will double the human population. The increase will not only be quantitative, but qualitative: the relevance of the IoT in our lives and societies will keep growing. Unfortunately, cyberattacks that target the IoT [2] will also increase: security and privacy issues are still the norm in IoT systems [3].

The term Internet of *Broken* Things reflects this reality. To change this, research effort in IoT security has risen in recent years [4]. Moving Target Defense (MTD) [5] is one promising security research field that can further help in this effort.

978-1-7281-8086-1/20/$31.00 ©2020 IEEE

MTD is a cyber-defense paradigm. It tries to disrupt the information-asymmetry between system and attackers. Most systems are static (e.g., well-known IP addresses, hardware). An attacker facing a static-system has potentially infinite time to explore it and find weaknesses (*reconnaissance*), prepare, and launch an attack. MTD proposes to perpetually change -*move*- components of the system. This will impose constraints on the attacker in the domain of Time.

The use of MTD for IoT systems has already been explored, but there is still a large gap between the MTD development on classical systems as compared to the IoT [6].

With the aim of reducing this gap, we propose IANVS: a generic framework to help in the design and implementation of MTD techniques adapted for the IoT. We also propose two IANVS-based concrete MTD techniques: one that targets UDP port numbers (port-hopping), and the other the Constrained Application Protocol (CoAP) `/.well-known/core` resource URI. We implement and share IANVS in micro-python code, and evaluate the port-hopping proposal empirically using a LoPy4 IoT platform facing a remote attacker.

The paper is organized as follows. Section II presents background, related work, and motivation for our work. Section III defines IANVS. Section IV defines two IANVS-based MTD techniques. Section V implements and evaluates one of the techniques in an IoT platform. Section VI offers some discussion and future work. Finally, Section VII concludes.

II. Background, Related Work and Motivation

A. Background: MTD Fundamentals

Cai et al. [7] divide MTD literature into three fields: theory, evaluation, and strategy. The first two deal with mathematical theory, system-attacker models and interactions, and metrics. The MTD strategy field is about concrete MTD techniques that can be implemented in real systems. In this work, we focus on MTD strategies. We use the terms MTD *strategy* and *technique* interchangeably. An MTD strategy needs to define three fundamental design questions: WHAT, HOW, and WHEN to *move*.

WHAT to move determines the component(s) of the system to which the technique will be applied, i.e., the **Moving Parameter(s) (MP)**. Based on the system layer of the MP, a common categorization for MTD techniques is the following:

1) *Network*. Network component, e.g., protocols, addresses.
2) *Platform*. Platform component, e.g., hardware, OS.
3) *Runtime Environment*. Changes to the execution environment, e.g., RAM addresses, instruction set.
4) *Software*. Changes to an application's binary code.
5) *Data*. Changes to the format of application data.

HOW to move is about the methods for (i) define valid states of the MP, and (ii) chose one valid state for the system. MTD techniques use three types of methods: Shuffling (randomization), Diversification, and Redundancy-based.

WHEN to move is about applying the state change, i.e., the decision process that triggers the MP value change. The literature identifies three types of decision processes: Time, Event, and Hybrid-based

B. Related Work: MTD for IoT

1) MTD Strategies for IoT: Even if more than 80 distinct MTD strategies exist [6]–[9], MTD strategies for the IoT are limited. Recent MTD surveys [6], [9] identify only 4 IoT-specific MTD strategies. Casola et al. [10] propose diversification of the network security protocols (*Network*) and firmware (*Platform*). The most explored strategy has been IPv6 address randomization [11]–[13] (*Network*). Mahmood et al. [14] proposed code execution partitioning and diversification (*Software*).

2) MTD-related IoT Frameworks: IoT-aimed Frameworks that can be applied to the MTD paradigm exist. Husain et al. [15] proposed a Reconfigurable symmetric-key based Encryption-decryption hardware Architecture (REA). REA intends to use fewer resources and be faster than AES, and to be the hardware foundation for MTD-based techniques. REA can be useful in the HOW design aspect of MTD strategies. Moshin et al. [16] proposed an ontology-based framework for the IoT aimed at helping defense systems against Advanced Persistent Threats. This proposal is at a higher level of abstraction than a particular MTD strategy and can be useful in the WHEN design aspect of MTD strategies.

C. Motivation

MTD techniques have improved the security of legacy systems. For example, Address Space Layout Randomization (ASLR) [17] is used in all modern general-purpose OSs. The MTD paradigm has the potential to improve the state-of-the-art of security for IoT systems. One of the main challenges is to design MTD strategies that use mechanisms suitable for the constraints of IoT. This may explain the quantitative difference between legacy and IoT MTD strategies, more than 80 against 4. Also, we found that IoT MTD strategies have weak cryptographic foundations. For example, in [10], [11], [14] the MP randomization method is not specified. In [12], [13] an unkeyed hash function (SHA-256) is used to construct a crypto-transformation that resembles a key-based mechanism. A more conservative security design would use a keyed-hash function. In this work, we propose a lightweight framework that intends to facilitate the design of IoT-adapted MTD strategies. Our framework intends to capture the common mechanisms required by MTD techniques. We also intend to contribute to the design of concrete MTD solutions adapted to the specific constraints of IoT systems.

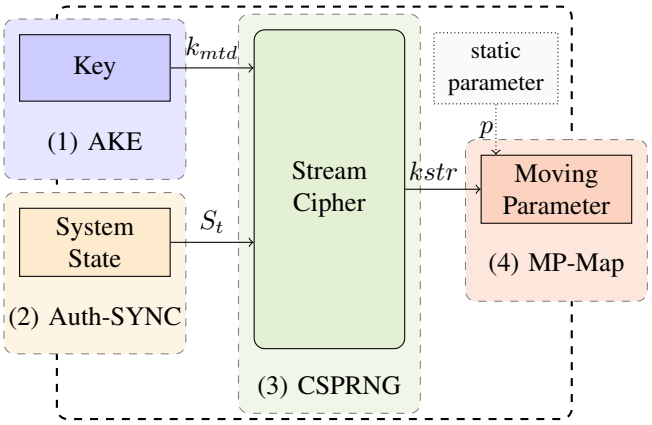

Fig. 1. IANVS MTD Framework components.

III. IANVS: An MTD Framework for IoT

In this section, we propose IANVS (pronounced *Janus*[1]). Our proposal is a generic framework that can be used to instantiate IoT-friendly MTD strategies. IANVS abstracts, generalizes, and links common components of MTD strategies. A concrete MTD strategy design can use IANVS as an archetype to build upon it. The main goal of IANVS is to leverage future design and implementation work, with the aim of having more robust and usable MTD for IoT. For example, components may be identical among different concrete proposals and could be re-used. Security proofs of a specific MTD strategy may ease or validate those of another one. We define two concrete IANVS-based strategies in Sec.IV, and implement and share their source code in Sec.V. In the rest of this section, we detail IANVS components and discuss their synchronization over distributed systems.

A. IANVS Components

Fig.1 depicts IANVS and its fundamental components:

1) AKE: An Authenticated Key Establishment mechanism.
2) Auth-SYNC: An Authenticated state Synchronization mechanism (e.g., authenticated time).
3) CSPRNG: A Cryptographically Secure Pseudo-Random Number Generator, from the stream ciphers family.
4) MP-Map: A transformation that outputs values in the Moving Parameter domain with equiprobability (e.g., uniform hashing).

Optionally, a static parameter value p can be an MP-Map input. In the following, we detail each component.

(1) The AKE component provides a cryptographic key (k_{mtd}). The k_{mtd} is secret and must only be shared by the trusted parties of an MTD strategy. AKE is related to the

[1]In honor of the Roman god of transitions.

secure key bootstrapping problem. Key bootstrapping is a hard problem to solve and, in general, relies on pre-shared cryptographic material or a trusted third party. Suitable AKE component candidates for the constrained IoT setting are AKE protocols like Ephemeral Diffie-Hellman Over COSE (EDHOC) [18], our nonce-based AKE previous work [19], or a Datagram Transport Layer Security (DTLS) handshake with a lightweight cipher suite.

(2) The Auth-SYNC component provides a system state value (S_t). The S_t must be authenticated and fresh, but not necessarily secret. Auth-SYNC is related to the secure time synchronization problem. There is currently a lack of suitable solutions for the constrained IoT. We assess the state-of-the-art and provide a proposal in our previous work [20].

(3) The CSPRNG component is at the core of the IANVS framework. It requires two inputs: (i) a cryptographic key (k_{mtd}), and (ii) an authenticated system state (S_t). They are provided by the AKE and Auth-SYNC components, respectively. The CSPRNG produces one output, a cryptographically secure pseudo-random binary key-stream ($kstr$). A stream-cipher [21] must be used as CSPRNG. A stream-cipher produces long binary outputs with strong cryptographic guarantees. Also, many IoT suitable options exist [22] like software-based ChaCha20, or AES in CTR mode[2] that is present in most modern IoT hardware.

(4) The MP-Map component outputs the MP configuration value. It takes as inputs the key-stream ($kstr$) from the CSPRNG and -optionally- a static parameter value p. It maps the inputs to a value in the MP domain. The MP-Map component is related to the problems of Map structures, i.e., a collection of *key-value* pairs. If there is only one MP in the domain, one possible solution is to use hash tables, i.e., rely on a hash function to do the transformation. The Uniform Hashing Assumption must be approximated, i.e., every *key* should be mapped to a *value* domain with equiprobability.

B. Synchronized Movement over Distributed Systems.

IANVS was designed to help in the implementation of MTD strategies in distributed systems. The AKE and Auth-SYNC components provide this distributed capability. A particular MTD strategy in a distributed system requires that the involved parties agree on the MP value over time. As depicted in Fig.1, after having fixed the CSPRNG and MP-Map, this is solely determined by the current values of the Key k_{mtd} and the System State S_t. Thus, k_{mtd} and S_t must be synchronized in order for the MP value to also be in a consistent state.

The AKE component is meant to be executed rarely, e.g., potentially only once in the lifetime of the node. Having to re-key should only be needed in exceptional cases. For example, when all the possible values of S_t were used, one node should be revoked from the MTD strategy, or the key has been compromised. On the contrary, the Auth-SYNC component is meant to be executed often, e.g., potentially once per MP movement. The Auth-SYNC component should be used to synchronize the MP movement.

[2]It behaves as a stream-cipher.

Auth-SYNC with Time. If the nodes have real-time clock capabilities, time-based synchronization solutions are applicable. It suffices to agree on a period of movement, run a secure time synchronization protocol, and then the MP movement can be triggered internally by the nodes. The time synchronization protocol might be executed again, depending on the synchronized internal clock accuracy needed by a particular MTD strategy.

Auth-SYNC without Time. If the nodes do not have real-time clock capabilities, the MP movement should be actively triggered by a node in the system and distributed to the others. A Master-Slave solution could be applied. The protocol should ensure that the trigger message is authenticated and fresh. To guarantee freshness (i.e., to avoid replay attacks), at least a two-message nonce-based protocol is needed. The protocol will be executed at every MP movement.

IV. IANVS Instantiation: Two MTD Strategies

In this section, we define two concrete MTD strategies that instantiate IANVS in a real IoT platform.

A. Common Components Definitions

In order to instantiate IANVS its four components should be clearly defined. In the MTD literature, the MP is one of the most important qualities that define a particular strategy. In IANVS, the MP-Map component determines the MP. We propose two Network-based strategies that differ only in the MP-Map component. The rest of the components are the same and defined as follows:

- *AKE*: A pre-shared symmetric Key of 128-bits.
- Auth-SYNC: We use periodic MP changes and the Network Time Protocol (NTP)[3] to synchronize the Real-Time Clock (RTC) of the MTD distributed nodes.
- CSPRNG: We use ChaCha20 with 20 rounds. The two inputs are the unmodified 128-bit key from AKE, and a 64-bit nonce derived form the NTP time.

B. Strategy I: Single Port-Hopping

This strategy corresponds to the *Network* category of MTD strategies. The MP in this strategy is the UDP port number of a service. TCP and UDP port number pseudo-randomization has been previously proposed in the literature [23]–[25], and is known as *port-hopping*. Well-known port numbers are necessary for network services discovering and use. However, the static nature allows for straightforward Denial-of-Service (DoS) attacks [26] (e.g., flooding a well-known port). Also, they are the entry point for adversaries in the *reconnaissance* phase of more sophisticated attacks that target higher layers.

MP-Map definition. UDP port numbers range from 0 to 65535 (16-bits). The MP domain cardinality is thus $|MP| \leq |2^{16}|$. We offer a strategy for a single-port hopping. Multiple port-hopping poses additional challenges and is discussed later. If the hopping-port is the only UDP port open, it is straightforward to use the 16-bits for port-hopping ($|MP| = |2^{16}|$). Let

[3]Network Time Security for the NTP must be used in a real deployment.

p be the well-known port number to transform. We apply a bit-wise `xor` with the first 16-bits of the ChaCha20 output $kstr$. This transformation is equivalent to the use of the ChaCha20 stream-cipher to encrypt p. The hopping port p_{mtd} equals $p \oplus kstr_{0..15}$. If other UDP ports are open, standard non-hopping ports may have a port range of 0-32767 (15-bits). Thus, the p_{mtd} should range from 32768 to 65535 ($|MP| = |2^{15}|$). The 16th bit of p_{mtd} should be set to 1.

About Multi-Port Hopping. Multiple MP in the same domain and codomain require a more complex MP-Map transformation. The transformation should be invertible, which is the case for the `xor` operation (bijective). However, security issues arise depending on the construction chosen. The current proposal, if used for multiple-ports, is prone to a simple well-known attack of stream ciphers: nonce-reuse. If the same $kstr$ is used to `xor` different inputs, it becomes a two-time pad. This has security-related consequences[4]. Therefore, a transformation should be used where a nonce-reuse is not that severe (e.g., nonce misuse-resistant).

C. Strategy II: CoAP `/.well-known/core` URI

This strategy corresponds to the *Network* category of MTD strategies. The MP in this strategy is the CoAP [27] `/.well-known/core` resource URI [28]. This resource is mandatory to implement for a CoAP Server. If a Client sends a GET request to the `/.well-known/core` URI, the Server responds with a payload that contains a set of resources available. For an IoT node with 2 resources, the size of the payload is of ≈50 Bytes. If an attacker wants to perform a simple DoS attack on a remote CoAP Server, sending GET requests to the `/.well-known/core` is one of the most straightforward ways to achieve it. If the node is energy-constrained, the increased use of the network interface will also lead to battery-exhaustion. Our proposal aims at mitigating these type of remote DoS attacks.

MP-Map definition. The CoAP protocol encodes a GET request to the `/.well-known/core` URI as two Uri-Path CoAP Options. They contain an ASCII-encoding of `.well-known` and `core` and have an Option Length of `11` and `4` Bytes, respectively. We propose to `xor` the ASCII-encoding with the $kstr$ of ChaCha20. The Server must only respond if the GET request corresponds to the current MTD representation of `/.well-known/core`. The MP moves in a codomain of 15 Bytes ($|MP| = |2^{120}|$). A collision can happen with a non-MTD URI of 11+4 bytes with a probability of $2^{-120} \approx 10^{-36}$. As opposed to UDP port-hopping ($|MP| = |2^{16}|$), this has a low probability of realization. If zero-collision is needed, other measures have to be taken, e.g., avoid combinations of Uri-Paths of `11+4` Bytes length.

Security Considerations. **MTD for a single CoAP resource URI is not a replacement for application layer security or an authorization framework**. For example, DTLS, OSCORE (Object Security for Constrained RESTful Environments), or ACE-OAuth[5] should be used to achieve security services such as confidentiality, authentication, or authorization. In general, MTD is not a replacement for information security. However, it is a complementary measure that can improve the system's resilience at a negligible cost. For example, if a IANVS-based strategy is hopping the CoAP default UDP port (`5683`) at the IoT node; with almost no increased cost, it can also be used to apply MTD to the `/.well-known/core` URI. A multi-layer proposal can use the first 2 Bytes of the $kstr$ for port-hopping and the following 15 Bytes for the CoAP URI.

[4]The `xor` of the cipher-texts equals the `xor` of the plain-texts.

Net. Protocol	Moving Parameter(s)	|MP|	MP-Map
UDP	port number	2^{16}	$p \oplus kstr$
CoAP	`/.well-known/core` URI	2^{120}	$p \oplus kstr$
UDP+CoAP	*(both above)*	2^{136}	$p_1 \| p_2 \oplus kstr$

TABLE I
PROPOSED MTD STRATEGIES USING IANVS WITH CHACHA20.

D. Proposals Summary

We provided two proposals that illustrate the use of IANVS to instantiate concrete MTD strategies. Both are compatible with legacy non-MTD components in the system, as neither modifies the underlying network protocols they are applied to. The composition of multi-layer MTD strategies was briefly discussed. The incurred incremental cost for additional Network-layers is negligible once IANVS is already in place. Table I resumes the proposals discussed in this section.

V. IMPLEMENTATION AND EVALUATION

In this section, we implement and evaluate the IANVS-based proposals. We share the source code and data in [29]. We use a real IoT Hardware platform in an IP Network. We expose the nodes to a remote attacker performing a DoS attack. We measure the effectiveness of the MTD proposal in terms of the reconnaissance-phase mitigation of the attack.

A. System: IoT Hardware Platform

We use Pycom LoPy4 nodes [30] as a hardware platform. A LoPy4 node has an Espressif ESP32 SoC (32-bit architecture @240 MHz, 520KiB RAM), a Real-Time Clock (RTC), 4MB of external RAM and 8MB of Flash. It has Wi-Fi, Bluetooth, Sigfox, and LoRA (Semtech SX1276) as physical network capabilities. We chose it because a node can run MicroPython code and has many network interfaces; this allows for flexible and fast prototyping. We used the Expansion Board 3.0 to flash the LoPy4 from a UNIX-based PC, and power it through USB.

B. Attacker Model

The attacker is remote. It is physically external to the IoT network but has IP access to it. The attacker knows the IP address of a target IoT device that hosts a CoAP Server over UDP. The attacker's goal is to perform a DoS attack targeted at

[5]Authentication and Authorization for Constrained Environments using the OAuth 2.0 Framework

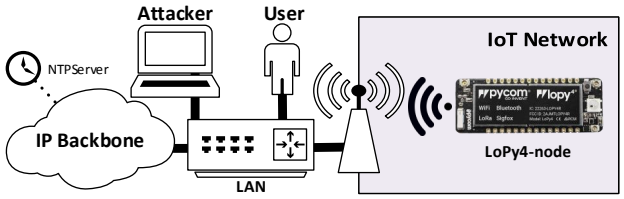

Fig. 2. Experimental Setup.

this IoT node. In order to do so, it floods the target node with CoAP GET `/.well-known/core` messages over UDP.

C. Experimental Setup

The setup is shown in Fig.2. The LoPy4 nodes use the Wi-Fi interface. They are one-hop from the Wireless AP. The attacker uses a Lenovo ThinkPad-i7 T460 PC running an Ubuntu 19.10 OS. It is connected to the LAN using an Ethernet 100BASE-TX port. The User has the same configuration. For time synchronization, we used an external NTP Server.

D. Experiment 1: UDP Port-Hopping Effectiveness against Reconnaissance-Phase of Attack

The goal of this experiment is to measure to which degree the UDP port-hopping can mitigate the reconnaissance phase of an attack.

1) Hypothesis: The attacker cannot eavesdrop other packets from the LAN. It knows that MTD is applied to the CoAP UDP port, and the range of ports used for hopping. It does not know the PSK nor the period of movement. He knows the NTP Time, but he cannot spoof the NTP Server.

Reconnaissance Success/Fail: If the attacker sends a UDP packet with a given port to the target IoT node (*udp-ping*), it will learn if that port is in use or not (*success* or *fail*).

2) Probabilistic Model: We define N as the number of ports used for port-hopping ($N \leq 2^{16}$). The actual port in use is uniformly chosen over N. Reconnaissance success for the attacker after a single *udp-ping* over N possible ports can be modeled as a random variable (r.v.) that follows a Bernoulli distribution with probability $p = \frac{1}{N}$. Over a single MTD period, the attacker can perform n number of *udp-pings*. The attacker cannot discard previously tested ports because it does not know when the port changes. Thus, the *udp-pings* are independent and identically distributed. Then, the number of reconnaissance successes over a single MTD period follows a Binomial distribution[6] $B(n, p)$, with n number of trials, and p probability of success of a single trial.

3) Implementation and Execution: We implemented port-hopping for LoPy4 nodes as specified in Sec. IV-B. We used `microCoAPy`[7] library and Joachim Strömbergson's `chacha.py`[8]. The attacker code uses the `hping3` packet generator tool. It randomizes the chosen port using the bash function `$RANDOM` (15-bits); if needed, applies a modulo N operation to restrict the result to the port-hopping range.

[6] $\sum_{i=1}^{n} Bernoulli_i(p) \sim Binomial(n, p)$, with n = number of trials.
[7] https://github.com/insighio/microCoAPy
[8] https://github.com/secworks/chacha/

N (#ports)	MTD period P (seconds)
512	{5,70,177,365,589}
1024	{10, 140, 355, 730, 1179 }
2048	{21, 281, 710, 1461, 2357 }

Attacker Speed = 2 attacks/s

TABLE II
UDP PORT-HOPPING EXPERIMENT PARAMETERS.

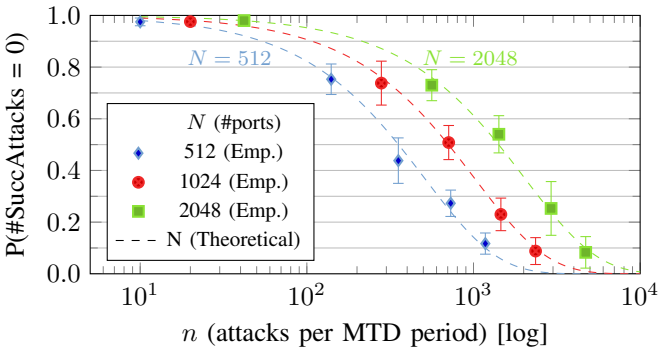

Fig. 3. Port-Hopping: Empirical probability of zero successful attacks over one MTD period as a function of attacks per period, for different #ports N.

The LoPy4 logs internally if the used port was found for a given MTD period. In this study, we focus on the probability of the port not being found at all. We tested several combinations of port-hopping range N, and MTD period lengths P. The attacker's *udp-ping* period is fixed to `500ms` (2 att./s). The parameters for the experiments are in Table II. In our experiments $n = P \times 2$ att./s.

For each tuple (N, P) of experiment parameters, we ran between 120 and 600 periods (samples). The total net run-time of the experiments is around 480 hours or 20 days. As an example, 120 runs of the ($N = 2048$, $P = 2357$) experiment have a net run-time ≈ 78 hours.

4) Results: For each tuple (15 in total), we calculated the empirical probability (i.e., the relative frequency) of a sample with zero successful attacks. In Fig.3, we can see the results. To measure the uncertainty of this value, we partition the tuple-experiment sample set in 5+ equally-sized subsets. We calculated the standard deviation of the empirical probability of zero successful attacks from each subset. The theoretical probability $P(\#SuccAttacks = 0)$ from the corresponding Binomial distributions $B(n, p = 1/N)$ is also plotted.

5) Analysis: As expected, the empirical data fits the probabilistic model. The parameters that determine the underlying probability are N and n. N can be controlled directly by the system; in this port-hopping strategy, it should be maximized as there is no additional cost for the system. The parameter n depends on the attacker-defender relationship between P and attacker speed. Adjusting P to arbitrarily small values will incur increased costs for the system (e.g., offline time, fine-grained synchronization); it should be determined according to the particularities of the real system and use case.

VI. Discussion and Future Work

A. Discussion

Composable Security. We designed IANVS to be modular. Each module draws from a security research field on its own. Unfortunately, the security of a complex system is not additive but "as strong as the weakest link". Even worse, in general, security is not even composable. The composition of secure sub-systems can lead to an insecure system. However, in the current form, IANVS is using a *secret key-authenticated nonce* pair and a stream cipher to encrypt data (MP). The security is as strong as that of the underlying stream cipher. As an example of non-composability, suppose using an AES-CBC block cipher instead. In that case, a time-nonce AUTH-Sync IANVS will be broken. AES-CBC requires the nonce to be unpredictable for an attacker. Thus, time-nonce solutions should not be used.

Multiple MP in the same domain. We briefly discussed multiple MP in the same domain in Sec. IV-B. The same stream cipher $keystream$ must not be used to directly transform different MPs. It would be equivalent to using a two-time pad. The two-time pad is prone to many straightforward attacks. We plan on developing appropriate MP-Map transformations that support multiple MPs in the same domain.

System Trade-offs. Security vs. cost trade-offs should be thoroughly evaluated in a real system. It will depend on its particularities. In Sec. V-D5 we gave an example for the port-hopping use case. In general, there will be a spot where the increased *potential* security is not worth the incurred system cost. In that trade-off decision, the resources of the attacker are also determinant. Fortunately, MTD has more than one domain in which to *play* the trade-off *game*. Naturally, the domain of *time*, but also the domain of the MP possible values.

B. Future Work

Future work includes the implementation and evaluation of the CoAP URI proposal. We plan to implement IANVS in more constrained IoT platforms and deploy it in an LPWAN setting like a LoRa network. The inherent constraints of the physical network layer might be beneficial from an MTD defender's perspective. We envisage the study of MP-Map transformations that securely support multiple MPs in the same domain. We also want to develop cryptographic proofs of the composability of IANVS components.

VII. Conclusions

MTD is a cyber-defense paradigm that has already contributed to improving the overall security in legacy systems. However, its usage in the IoT context is almost non-existent. In this work, we proposed IANVS, a generic framework that helps in the design of concrete MTD strategies for the IoT. We designed two of them. We implemented IANVS in software and evaluated one of the MTD strategies in real IoT hardware. We show that MTD can effectively mitigate otherwise trivial attacks. With this work, we hope to increase the interest of the IoT research community in the MTD paradigm and to help in the design of novel strategies or frameworks.

References

[1] M. D. Saulles. (2019) Iot statistics - information matters. [Online]. Available: https://informationmatters.net/internet-of-things-statistics/

[2] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "Ddos in the iot: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[3] O. Alrawi *et al.*, "Sok: Security evaluation of home-based iot deployments," in *2019 IEEE Symp. on Security and Privacy*. IEEE, 2019.

[4] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, 2018.

[5] A. Ghosh *et al.*, "Moving target defense co-chair's report-national cyber leap year summit 2009," *Tech. Rep., Federal NITRD Program*, 2009.

[6] J. Zheng *et al.*, "A survey on the moving target defense strategies: An architectural perspective," *Journal of Comp. Science and Tech.*, 2019.

[7] G.-l. Cai *et al.*, "Moving target defense: state of the art and characteristics," *Frontiers of Information Tech. & Electronic Engineering*, 2016.

[8] B. C. Ward *et al.*, "Survey of cyber moving targets second edition," MIT Lincoln Laboratory Lexington United States, Tech. Rep., 2018.

[9] J.-H. Cho *et al.*, "Toward proactive, adaptive defense: A survey on moving target defense," *IEEE Comm. Surveys & Tutorials*, 2020.

[10] V. Casola *et al.*, "A moving target defense approach for protecting resource-constrained distributed devices," in *2013 IEEE 14th International Conf. on Information Reuse & Integration (IRI)*. IEEE, 2013.

[11] M. Sherburne *et al.*, "Implementing moving target ipv6 defense to secure 6lowpan in the internet of things and smart grid," in *Proc. of the 9th Annual Cyber and Information Security Research Conf.* ACM, 2014.

[12] K. Zeitz *et al.*, "Designing a micro-moving target ipv6 defense for the internet of things," in *2017 IEEE/ACM Second International Conference on Internet-of-Things Design and Implementation (IoTDI)*. IEEE, 2017.

[13] ——, "Changing the game: A micro moving target ipv6 defense for the internet of things," *IEEE Wireless Communications Letters*, 2018.

[14] K. Mahmood *et al.*, "Moving target defense for internet of things using context aware code partitioning and code diversification," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016.

[15] M. I. Husain *et al.*, "Lightweight reconfigurable encryption architecture for moving target defense," in *MILCOM 2013*. IEEE, 2013.

[16] M. Mohsin *et al.*, "Where to kill the cyber kill-chain: An ontology-driven framework for iot security analytics," in *2016 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2016.

[17] S. Bhatkar *et al.*, "Address obfuscation: An efficient approach to combat a broad range of memory error exploits." in *USENIX*, 2003.

[18] G. Selander *et al.*, "Ephemeral Diffie-Hellman Over COSE (EDHOC)," IETF, I-D draft-selander-lake-edhoc-00, Nov. 2019, work in Progress.

[19] R. E. Navas, M. Lagos *et al.*, "Nonce-based authenticated key establishment over oauth 2.0 iot proof-of-possession architecture," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016.

[20] R. E. Navas and L. Toutain, "Late: A lightweight authenticated time synchronization protocol for iot," in *2018 Global Internet of Things Summit (GIoTS)*. IEEE, 2018.

[21] S. Banik *et al.*, "Towards low energy stream ciphers," *IACR Transactions on Symmetric Cryptology*, pp. 1–19, 2018.

[22] M. A. Philip *et al.*, "A survey on lightweight ciphers for iot devices," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*. IEEE, 2017, pp. 1–4.

[23] H. C. Lee and V. L. Thing, "Port hopping for resilient networks," in *IEEE 60th Vehicular Technology Conference*. IEEE, 2004.

[24] G. Badishi *et al.*, "Keeping denial-of-service attackers in the dark," in *International Symposium on Distributed Computing*. Springer, 2005.

[25] Y.-B. Luo *et al.*, "Effectiveness of port hopping as a moving target defense," in *7th International Conf. on Security Tech.* IEEE, 2014.

[26] L. Liang *et al.*, "A denial of service attack method for an iot system," in *2016 8th International Conference on Information Technology in Medicine and Education (ITME)*. IEEE, 2016.

[27] Z. Shelby, K. Hartke, and C. Bormann, "The Constrained Application Protocol (CoAP)," RFC 7252, Jun. 2014.

[28] Z. Shelby, "Constrained RESTful Environments (CoRE) Link Format," RFC 6690, Aug. 2012.

[29] H. Sandaker and R. E. Navas, "IANVS: An MTD Framework for a Resilient IoT. Port-hopping code and data." May 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3839686

[30] (2020) Lopy4: a quadruple bearer micropython enabled development board. [Online]. Available: https://pycom.io/product/lopy4/