Enabling Moving Target Defense for Real-Time CPS Security

Rajarshi Mukherjee
Department of Electrical and Computer Engineering
Virginia Tech
Arlington, Virginia, USA
rajarshim13@vt.edu

Mohamed Azab
Department of Computer and Information Sciences
Virginia Military Institute
Lexington, Virginia, USA
azabmm@vmi.edu,
mazab@vt.edu

Thidapat Chantem
Department of Electrical and Computer Engineering
Virginia Tech
Arlington, Virginia, USA
tchantem@vt.edu

***Abstract*—The Internet of Things (IoT) phenomenon has rather quickly enveloped modern society leading to regular interactions with several IoT devices and networks. Due to the exponential increase in the utilization of such devices, coupled with inherent limitations such as resource constraints and outdated firmware these devices are vulnerable and may be regularly subject to various threats. Most such systems lack any and all forms of encryption due to cost and resource constraints, as well as the general perception that such devices pose a low risk for attacks. Additionally, IoT networks are also present in several Cyber-Physical System (CPS) based applications such as water treatment facilities, where an entity can exploit these vulnerabilities to cause an attack with far-reaching consequences. With that in mind, a Moving Target Defense (MTD) is an emerging tactic to help shore up the defenses of such networks. This is done by regularly updating system parameters with the intention of putting up barriers for any malicious individual attempting to breach the network. This work deals with an attack scenario, wherein an attacker aims to take control of the sensors in a wastewater treatment facility, with the intention of deceiving the operators of said facility. An attack of this manner has the ability to tamper with the chemical composition of the treated wastewater which can end up devastating the ecosystems that this water gets released back into. This paper shows how an attacker can perform a Man-in-the-Middle (MiTM) attack using ARP (Address Resolution Protocol) spoofing and subsequently launch a Denial of Service (DoS) attack on the sensor network and disrupt the legitimate data stream. Based on this attack, an MTD approach is proposed, which is based on the routine updating of the IP and MAC addresses of the sensors with the aim to obfuscate as much of the network from the attacker as possible. The findings show that this method significantly improves the network's ability to resist ARP spoofing and allows it to gracefully recover from a DoS attack. Additionally, this work experimentally showcases the overhead these techniques may impose. This would allow for system designers to provision for the resources needed without affecting the timeliness of time-sensitive systems, i.e., Real-Time Systems (RTS).**

***Index Terms*—moving target defense, mqtt, sensors, rts**

This paper was supported in part by the National Science Foundation (NSF) under grant number CPS-2038726, and by the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development. For more information about CCI, visit www.cyberinitiative.org.

979-8-3315-4090-6/24/$31.00 ©2024 IEEE

I. Introduction

Cyber-Physical Systems (CPS) are consolidated systems where a mixture of computational elements and sensors are tasked with the monitoring, logging, and regulation of physical objects in real time. Over the years, such systems have been deployed to control critical infrastructure, some of which have recently been found to have several vulnerabilities, such as Supervisory Control and Data Acquisition (SCADA) systems [1], smart grids [2], and automobiles [3]. [4] shows the terrifying potential of such attacks with hospitals, power-plants , and oil pipelines all being targeted. This work looks at such a Real-Time CPS (i.e., CPS with timing constraints), that has been crafted using common sensors utilized in Internet of Things(IoT) networks, which communicate over Wi-Fi as is common in several applications. Thereafter, some common security challenges that such a system faces are analyzed and a Moving Target Defense (MTD) based approach is employed to help secure it. Security in such systems can be imagined to be similar to a constant game of hide-and-seek, where the network is trying its best to hide the true details of its configuration layout and schematics, while the attacker is always chasing down different pathways to glean more details. [5] best describes MTD as a method to increase the "uncertainty and unpredictability" of a system in a way that improves its security while still allowing it to maintain a similar level of utility. Previous work in this domain [6] have noted how MTD approaches can be broadly classified under shuffling, redundancy and diversity. While there have been approaches discussing IP randomization [7], SDN based MTD approaches [8], virtualization based techniques [9], and even IPv6 based approaches [10], there has been a dearth of research specifically looking at device level implementations for IoT networks. This research is an attempt to showcase the feasibility of having an MTD based approach on a realistic test-bed to tackle a real world threat model, and assess its benefits in contrast to the costs it imposes on system resources.

This work falls squarely under the shuffling category of MTD techniques. By continuously varying and hence hiding the system parameters, the aim is to show that by creating a

continually moving target for attackers, the overall difficulty of breaking into a system is significantly increased. The primary focus of this paper is to emphasize how projects both small and big in scale need to implement robust and proactive defense techniques to repel bad actors. These projects also include real-time CPS where overhead due to MTD must be accounted for in order to guarantee performance. In our CPS, for instance, data that is delivered to the system late may be of limited use and might be discarded. Thereby, any attacker who can even slightly delay the transmission of accurate data, can end up causing several missed deadlines, and significantly reduce performance. To that end, the total overhead introduced by this approach, compared to a defenseless network, reflects the costs associated with implementing an MTD setup in such a system. This work helps show how such a defense technique can easily be adopted on any general connected system that involves sensors communicating over commonly intercepted media such as Wi-Fi, and the benefits derived from having such a proactive approach to system design.

II. Targeted Threat Vectors

In this section, the primary threat types dealt with are examined, alongside a discussion on how a bad actor might carry out such attacks on a network in order to hamstring its ability to communicate reliably.

A. ARP spoofing

When devices on the same local network need to differentiate amongst each other, they do so on the basis of a 48 bit 12 digit hexadecimal number popularly known as the Media Access Control (MAC) address. By design, these addresses could potentially be thought of to be specific to a given device or computer, and something that is encoded into the hardware by the device manufacturer. However, there do exist several tools and software programs that allow individuals the ability to change their MAC addresses to any desired address, a fact which opens up the potential of spoofing a known device's MAC address and essentially taking the place of the target device in a network. On the other hand, when devices want to communicate globally on the internet across multiple networks, they require an Internet Protocol (IP) address to help enable communication. Assuming that the IP addresses of two devices are known, when communication between the two is attempted, a secondary protocol is required to resolve which physical device has been assigned a particular network address. To that end, the Address Resolution Protocol (ARP) is utilized to help the devices finally achieve communication [11].

ARP works on the basis of requests and replies, where the sending device first broadcasts an ARP request across the network containing the IP address of the intended recipient and requests its associated MAC address. The receiving device assigned to that IP address then posts an ARP reply, containing its MAC address, in turn verifying its identity. This communication is generally unencrypted and is hence highly prone to spoofing or poisoning attacks [12]. In practice, an attacker with access to a network could use several tools to eavesdrop upon the ARP requests and replies to ascertain the addresses of the devices communicating on it. Therein, they could choose to assume the identity of one of these devices by spoofing their MAC address. This opens up a lot of channels of attacks [13] where attackers could choose to listen in on privileged communication, assume the identity of a chosen device and inject fake data, or stop the chosen device from communicating freely.

B. DoS attack

Denial of service attacks, as the name suggests, attempt to prevent a machine or a network node from providing its regular service by overwhelming its capabilities through a constant barrage of directed requests. DoS attacks are similar to a battering ram, and they come in with the sole intention to completely overcome a target's ability to deal with the traffic directed their way, and thus degrade the performance of entire systems. IoT systems with their scarce resources offer a lucrative target for bad actors to launch DoS attacks on. DoS attacks are also commonly used in conjunction with a spoofing attack to block actual data from being transmitted while the attacker can inject manipulated data in its stead.

C. System specifications

For evaluations three ESP 8266 NodeMCU devices are used, with each simulating a sensor in a network. Each NodeMCU has an Xtensa LX106 microcontroller, with 4 MB of flash memory, and an on-board Wi-Fi module. The low-cost nature of these boards, along with their ability to communicate over both bluetooth and Wi-Fi, makes them an ideal choice for several Internet of Things (IoT) applications. A Mosquitto server is deployed on a Linux based Raspberry Pi 4 Model B (quad core ARM Cortex-A72 processor, 2GB RAM), and set up with a Message Queue Telemetry Transport (MQTT) based communication network to best mimic an actual IoT application. MQTT is a popular and lightweight communication protocol that is preferred for many resource-constrained applications. It operates on a publisher-subscriber model where devices can communicate on topics or channels. Every ESP8266 device on the network functions as a sensor that communicates to the Mosquitto broker over individual topics dedicated to the readings from that sensor. The broker processes these readings and channels them to the designated destinations. This choice of a test-bed to evaluate these techniques on is based on a desire to show the validity of a similar MTD approach on an easily reproducible setup.

III. Threat Model

In a wastewater treatment facility, various IoT sensors are deployed to measure different quality indicators such as pH levels, turbidity, and chlorine levels. These sensors are distributed across multiple water tanks and send their readings to a central control system via the MQTT protocol. The Human-Machine Interface (HMI) screens display these readings to the operators for monitoring and controlling the facility. The

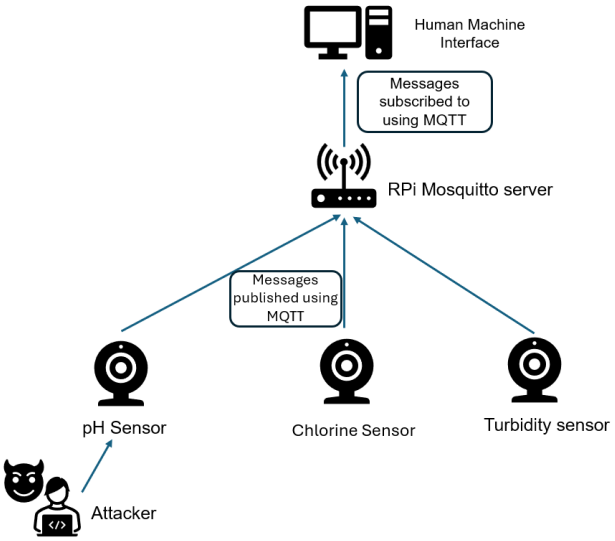


Fig. 1. The threat model showing the attacker, the sensors, the server and the human machine interface.

attacker aims to manipulate the pH level readings without changing the displayed values on the HMI screens. This can cause potential safety and compliance issues. To achieve this, the attacker performs a Man-in-the-Middle (MiTM) attack using ARP spoofing and then launches a Denial of Service (DoS) attack to disrupt the legitimate sensor data stream.

It can be assumed that the attacker has good knowledge of the facility's network to identify the IP and MAC addresses of the sensors responsible for measuring pH levels and the MQTT broker. The attacker performs ARP spoofing to intercept the traffic between one of the pH sensors and the MQTT broker to understand the conversation and capture the MQTT Topic and the data stream format. The attacker records the normal pH levels reported by the actual sensor to facilitate future replay attacks. The communication between the sensor and the MQTT broker uses the MAC address of the sensor as a unique sensor identifier. By sending malicious ARP replies, the attacker tricks the MQTT broker into trusting the compromised node. The attacker can then easily replay an old, captured message showing a normal pH reading. The facility is assumed to not have any encryption for the sensor data, a situation unfortunately common in closed, isolated, and presumably physically secure networks.

The attacker begins injecting fake pH level readings into the MQTT traffic. These fake readings are designed to appear normal but slightly deviate from the actual sensor data. To ensure their fake data remains undetected and to avoid any suspicions due to discrepancies in the compromised sensor data stream readings and the other sensors' readings, the attacker launches a Ping Flood [14] attack (a type of DoS attack) against the actual sensor to block or delay their traffic. This results in the legitimate sensor data being either blocked or delayed, thereby allowing the attacker to upload their own manipulated data-points to the HMI screens. At this point, physical manipulation of the pH concentration will not be easily detected, and the attacker can cause serious damage. As the wastewater treatment facility operators rely on the HMI screens to monitor pH levels, the fake data could cause the system to either underdose or overdose different chemicals, leading to potential safety hazards such as insufficient disinfection or chemical overuse. The manipulation might also cause regulatory compliance issues, resulting in fines or operational shutdowns.

IV. Moving Target Defense Framework

This section will include a discussion on the techniques utilized to protect against the threat model discussed in the previous section. It can be ascertained that the network has to be reinforced against two primary scenarios. Firstly, it should attempt to hide from an attacker's view the true schematics of the network including the list of its device and network addresses. The proposed method, as described below is to shuffle these addresses to make it difficult for an attacker to keep up with the ever-changing parameters. Secondly, in the event that an attacker does get a hold of legitimate and active network parameter values and attempts to block the actual sensor from transmitting data, an approach to remove the attacker from the network and regain control of the communication channel is discussed below.

A. Shuffling

In any typical IoT network, the MAC addresses, and the IP addresses assigned to the devices on it are fixed, and thereby offer a very convenient attack surface for a malicious individual. The attempt here is to make any network appear as a moving target where the MAC addresses of the NodeMCU devices are programmatically generated and periodically updated. The new MAC addresses are randomly generated and updated at every chosen interval of time. Each time a new MAC address is assigned, each device requests a fresh IP address from the network router. Fig.4 shows how a single sensor with a single device ID can have multiple MAC and IP addresses assigned to it as per this moving target defense methodology.

As outlined in Fig.2 , this approach involves an initial connect by the sensor to the MQTT network using its original manufacturer provided MAC address. It is assumed that this initial addition of devices is carried out in a supervised fashion where an operator manually whitelists these MAC addresses and allows the devices to start communicating on the network. Once a network has established its first connection, the shuffling technique is implemented following a pre-specified interval of time after which each sensor then declares its newly randomized MAC address on a secure MQTT topic. It is assumed that the attacker is under no condition able to gain access to the messages being relayed on this channel. Strict server rules are implemented that only allow certain whitelisted MAC addresses to establish connections to the server. Then the server on receiving this broadcast adds the new MAC address to its list of whitelisted addresses, and also checks if it is due to purge some of

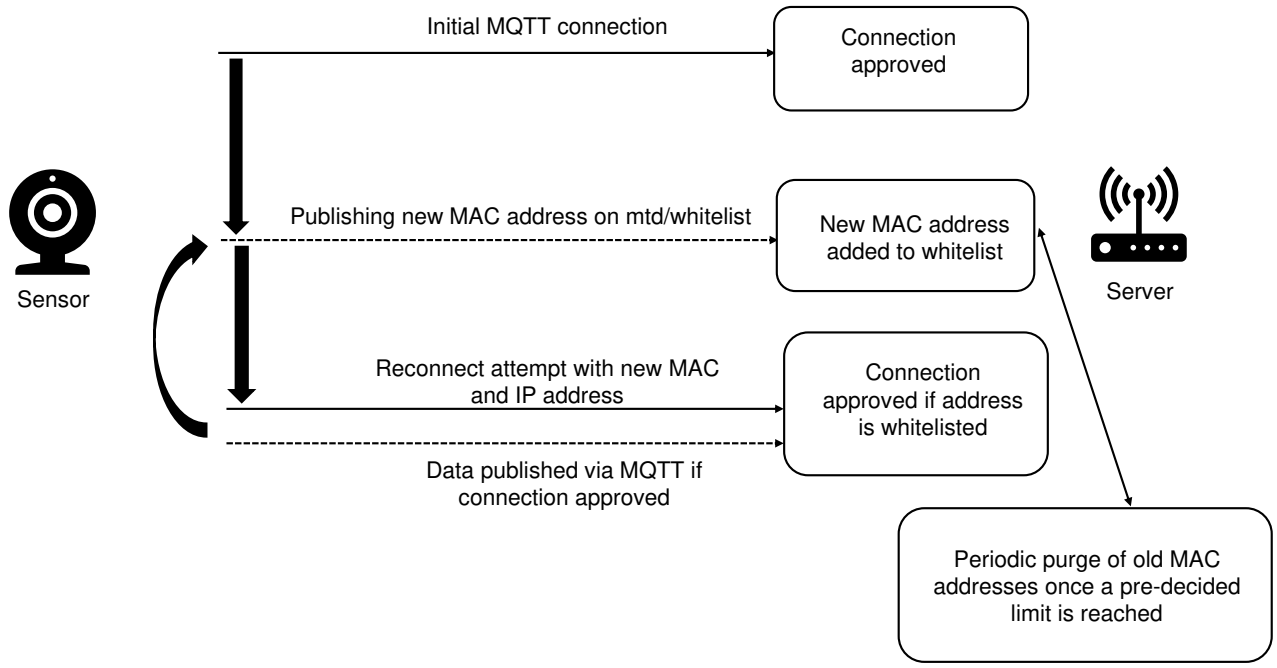


Fig. 2. The Moving Target Defense methodology showing the MAC address whitelist being updated and subsequent communications via MQTT.

the oldest MAC addresses from its whitelist. This limit of how many whitelisted addresses to store is dynamic and can be decided as per the application and the total number of devices that might be required in the network. This move ensures that stale addresses get removed in a timely manner, so that any attacker that manages to get a hold of any whitelisted MAC address has only a finite amount of time before that address gets its approval rescinded. Subsequently, right after broadcasting its new address value, the sensor terminates the existing Transmission Control Protocol (TCP) connection, gets assigned a new randomized MAC address, requests a fresh IP address, and then attempts a reconnect over MQTT. Once its MAC address is validated against the approved list of addresses, the device is again allowed to publish freely on the network till another reshuffle is initiated. This approach aims to make it significantly more challenging for an attacker to burrow into a network, reduces the window during which a network can be successfully scanned, and thus aims to mitigate some of the common problems that plague IoT networks. Additionally, certain modifications are made to address specific concerns that arise when the system has been actively breached, as demonstrated below.

1) Periodic polling of existing connections: A potential problem still exists where should an attacker be able to infiltrate the network by spoofing a whitelisted MAC address, the approach described above would not be able to terminate an existing MQTT connection, since no checks are performed after the initial connection is established. To that end, a background task on the server is used that can be scheduled to run periodically every few minutes. As shown in Fig. 3, the script goes ahead and first retrieves a list of all active MAC addresses connected to the network. Then, all these addresses get verified against the present whitelist, which has been gradually updated by the valid devices on the network, and every connection with a MAC address which is not present in the whitelist is immediately flagged as an unauthorized device and the connection is terminated. After the flagged connection is terminated, the firewall rules are updated to ensure that a legitimate sensor is still able to connect using that MAC address at a later stage. Thus, even if the attacker does spoof a legitimate MAC address and is able to establish an MQTT connection, they would only be allowed to transmit data for a finite amount of time before they are shut out of the system,

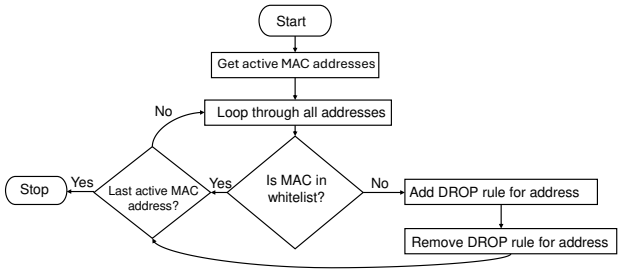
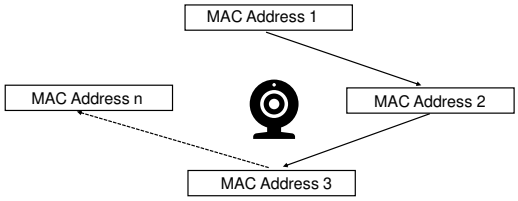


Fig. 3. Flowchart showing the technique to purge old connections periodically.

and they have to go through the procedure of obtaining a fresh and valid MAC address all over again. The frequency of this polling methodology can be dynamically altered depending on the needs of the system, the nature of the attacker, and the type of timing constraints under consideration.

2) Maintaining a ledger of whitelisted MAC addresses on sensors: Now, to address the second issue that was broached earlier, about the scenario where an attacker gets a hold of the details of the devices on the network and chooses to block off or delay a sensor from reporting its readings using a ping flood attack on a device's IP address. Such an attack would drastically reduce performance and might even prevent any traffic from going from the affected sensor to the server. This means, that there might be a scenario such that, when the sensor attempts to reshuffle its MAC and IP addresses, it might be unable to establish a stable enough connection to publish those details on the whitelist topic. To that end, a very simple mechanism is proposed where each sensor internally maintains a ledger of the last few MAC addresses that it had successfully been allowed to broadcast with. It is assumed that at the time of the attack, the sensor had been broadcasting for a while to have these ledgers already populated. Subsequently a routine is added to the regular MQTT reconnect procedure where should a sensor fail to connect a few times in a row, as can be expected in case it hasn't been able to communicate its new MAC address yet, the sensor will attempt reconnects with the old MAC address values stored in its ledger. Multiple addresses are stored to account for scenarios where either the attacker has ping flooded other IP addresses that might still be associated with an old physical address, or in case the oldest MAC addresses already got purged from the whitelist. Using a pre-vetted MAC address, the sensor can now reconnect, and quickly initiate another shuffle before the attacker can catch on. This ensures that even when the attacker has identified and blocked off several physical and network addresses that have been used previously by a sensor, the network has an inherent ability to rebound from this attack and try to continue to broadcast information.



Device ID	Network Address	MAC Address
xyz	192.158.12.2	MAC Address 1
xyz	192.158.12.4	MAC Address 2
xyz	192.158.12.8	MAC Address 3
xyz	192.158.12.n	MAC Address n

Fig. 4. A single sensor in a network can assume multiple MAC and IP addresses.

B. Summary of Framework

In summary, this framework can be described as:

- Dynamically changing the MAC and IP addresses of the sensors on the network.
- Maintaining a whitelist of approved MAC addresses that gets routinely updated and maintained.
- Periodically checking to ensure the existence of only approved devices on the network.
- Maintaining a short local ledger on sensors to attempt reconnects with in the event of a DoS attack.

V. System Evaluation

In this section, the experimental results obtained are discussed and evaluated. Firstly, the discussion will focus on the overhead that the MTD approach introduces into the network as opposed to a naive system without any form of proactive security. Secondly, the discussions will revolve around the observations from conducting an attack on the system as described in the threat model, and thus showcase the benefits of using a moving target approach in such situations.

A. Overhead due to Moving Target Defense

In this experiment, the spotlight is on quantifying the impact of a moving target defense on overall system performance, specifically narrowing down on the total overhead imposed by this dynamic shuffling technique. A message exchange between a sensor and an actuator controller server operating within a mission-critical RT application is simulated. In this scenario, it is assumed that the system has firm deadlines, as delays in message exchange could result in delayed actuation, potentially leading to significant issues or system failure. The experiments are conducted by repeatedly transmitting a fixed set of messages over the network, first without any shuffling, and then by gradually introducing the defense methodology with varying shuffling intervals, and record the resulting overhead produced. In the context of an RTS, the messages can be considered to be periodic tasks with fixed periods and relative deadlines. For a task to be able to meet its deadline, it needs to successfully be transmitted over MQTT onto a specific topic and received and processed by the server within a required amount of time. It is additionally considered that tasks are released, or become ready for execution once the previous task has been published.

Fig.5 and Table I show a distinct and predictable trend with regards to the frequency of the shuffling attempts. The number of deadline misses and the overhead are derived for 200 messages-each having a specific deadline. Additional overhead due to network latency, jitter, and transmission delays are accounted for on top of the delays due to the fresh reconnect attempts every time a new MAC and IP address are allotted. Tasks are scheduled to be released 500 ms after each other, and have an additional 100 ms after being released to be processed by the server. All the timeliness calculations occur on the server end in order to avoid issues with clock drift which would require additional synchronization. If the actuation server does not receive a new message from the

sensor within 600 ms of the previous one, it discards that message and records a deadline failure. It can be observed that with an increase in address shuffling interval from 30 to 70 seconds, the network overhead drops precipitously from over 36000 ms to 8900 ms, and so do the number of deadline misses from 5 down to 1. Additionally, it is observed that when the system is not having to deal with an active attack on any sensors (as in this case), it is able to predictably across multiple experiments maintain a stable overhead, that allows system designers more confidence when it comes to guaranteeing system behavior and for the building of systems more compliant to potential deadline misses. In order to show the nature of the overhead observed by varying other system parameters as well, Fig.7 can be referred to see how the system behaves when the period of the tasks, or the inter-arrival time between tasks is varied. By increasing the period from 200 to 1000 ms and keeping a constant shuffling interval of 50 seconds. The trend is complementary of the behavior observed in Fig. 5 and can be explained as follows. As the inter arrival time of tasks is increased, keeping the number of tasks constant, the total time needed to transmit all of the tasks increases as well. This leads to encountering more reshuffles while waiting for the tasks to be released, and hence more overhead.

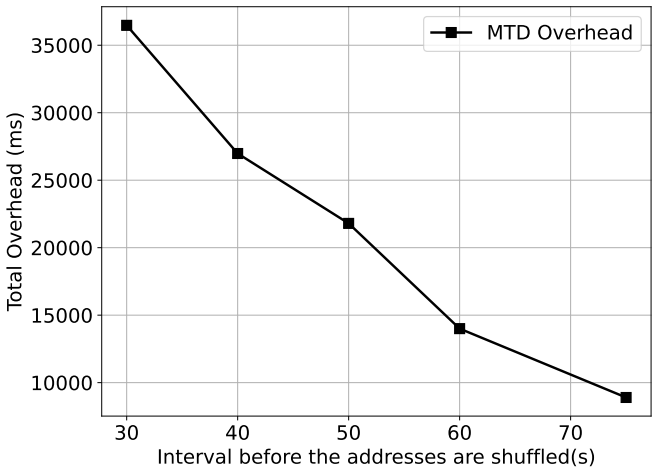


Fig. 5. Total overhead due to MTD as a function of time between shuffling.

TABLE I
SHUFFLING FREQUENCY VS. DEADLINE MISSES AND TOTAL OVERHEAD

Shuffling Frequency (s)	Deadline Misses	Total Overhead (ms)
30	5	36475
40	4	26978
50	3	21800
60	2	14001
70	1	8900

B. Response to ARP spoofing and DoS attacks

Continuing with the example from above, for this experiment a sensor and an actuator communicating in a time-critical application are considered. The attacker, after attempting to and failing to connect using random or outdated MAC addresses, finally manages to find an opportunity to connect using a scanned MAC address which is still whitelisted. The attacker is then able to connect, and both listen in on and publish fake messages on the network. Since the legitimate sensor has not been completely blocked yet, it still manages to publish valid data on the server. However, as soon as a few reshuffle attempts are made, and the background poller task activates, it promptly disconnects the attacker from the network due to its MAC address not being whitelisted anymore. After being unable to have any sort of long-term success with just trying to spoof a MAC address, the attacker tries to block the sensor from communicating with the actuator by performing a DoS attack on an active IP address they managed to scan. Fig.6 shows the delays or total overhead imposed on a naive and an MTD framework equipped system, when under DoS attack. The MTD framework can be seen to lead to a slightly higher overhead across the first few messages, which is primarily due to the multiple reconnect attempts made before the sensor successfully reverts to a MAC address from its internal ledger. These changes alongside a change in its IP address helps repel the attempt to block communications. It can be clearly observed how over time the MTD framework is able to resist the DoS attack, and deliver a much lower overhead, and hence much better performance than a naive system. Thus, unlike a regular system, the attacker must constantly struggle to gain the upper hand against the system's inherent characteristics.

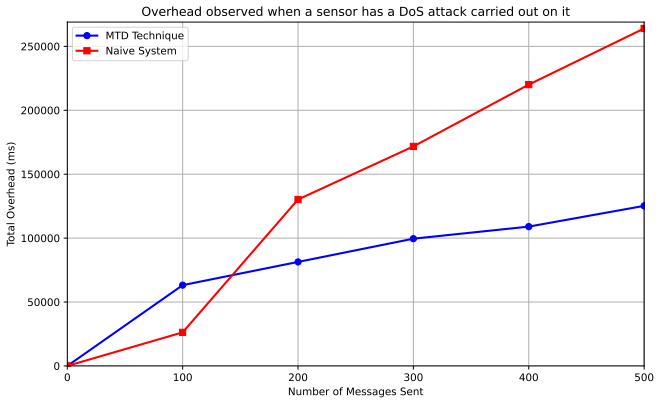


Fig. 6. Comparing the overhead of MTD to a naive approach when under a DoS attack

VI. FUTURE WORK

A. Authentication

MQTT communication can be made significantly more secure by using the Transport Layer Security (TLS) protocol [15]. This involves using certificates to validate the authenticity of all the clients as well as the server. Since MQTT is a light-weight protocol, TLS helps provide security against known network level threats by providing a layer of encryption to the communication, and thereby makes it

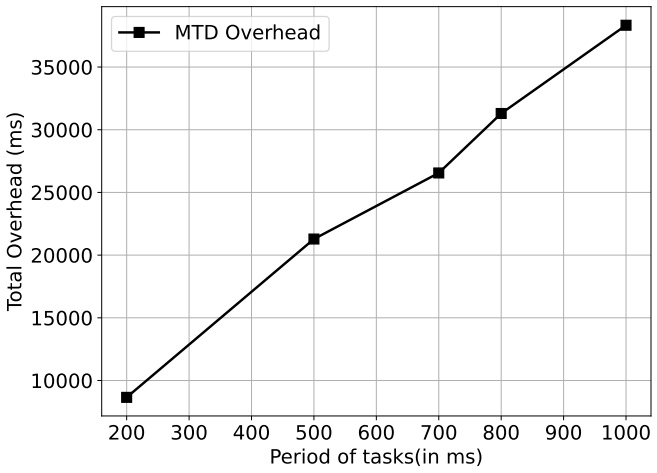


Fig. 7. Total overhead observed due to MTD vs. varying task periods

difficult for any bad actor to decipher the contents of the messages exchanged on the network. Due to the resource constrained nature of IoT applications, encryption is often forsaken for want of performance, however moving forward the aim is to showcase a way to include encryption and still maintain timeliness constraints. While encryption alone might not be enough to deter determined attackers, an MTD based approach coupled with encrypted data channels promises to be challenging to most bad actors.

B. Multi-factor decision making algorithm

As observed in the previous sections, despite the system having many dynamic parameters, there still remain several key items such as the address shuffling interval, the number of addresses to whitelist at any given time, and the period of the background poller task, that are constant during program execution. This work can be extended to allow for these parameters to be changed on the fly as per the circumstances of the system. Further research will make use of modern learning based techniques to be able to flag critical situations when the system is under extreme duress. Under such circumstances, the system would react by reducing the shuffling interval, and increasing the frequency of the background poller to quickly and efficiently gain an upper hand over most attackers. This would allow the most judicious use of system resources due to the ability to switch between modes of operation based on perceived threat levels.

VII. Conclusion

This paper is an attempt to draw attention to the need to develop Real-Time Cyber Physical Systems with a direct emphasis on the security needs that may plague it. With attackers getting smarter and better equipped, the need of the hour is a proactive design mindset. This research has shown how making the target mobile makes it significantly harder to hit, and even in the scenario that the attacker catches up to the defense technique, how their ability to cause damage can be limited. This moving target defense method is shown to be useful against attackers attempting to breach a network, and the results can easily be replicated across multiple combinations of hardware and firmware with minor system specific changes. Additionally, this aims to give other checks and balances that might be present in surrounding systems, or even the humans in charge, the time to spot irregularities in the readings and prevent the situation from snowballing into something more serious.

References

[1] X. Gao, T. Shang, D. Li, and J. Liu, "Quantitative risk assessment of threats on scada systems using attack countermeasure tree," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, 2022, pp. 1–5.

[2] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 4, pp. 370–379, Dec. 2014.

[3] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *2010 IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2010, pp. 447–462.

[4] W. Duso, M. Zhou, and A. Abusorrah, "A survey of cyber attacks on cyber physical systems: Recent advances and challenges," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. JAS-2021-1215, p. 784, 2022.

[5] J. Zheng and A. S. Namin, "A survey on the moving target defense strategies: An architectural perspective," *Journal of Computer Science and Technology*, vol. 34, no. 1, pp. 207–233, 2019. [Online]. Available: https://jcst.ict.ac.cn/en/article/doi/10.1007/s11390-019-1906-z

[6] Y. Magdy, M. Azab, A. Hamada, M. R. M. Rizk, and N. Sadek, "Moving-target defense in depth: Pervasive self- and situation-aware vm mobilization across federated clouds in presence of active attacks," *Sensors*, vol. 22, no. 23, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/23/9548

[7] A. Clark, K. Sun, L. Bushnell, and R. Poovendran, "A game-theoretic approach to ip address randomization in decoy-based cyber defense," in *Decision and Game Theory for Security (GameSec 2015)*, November 2015, pp. 3–21.

[8] M. Azab, M. Samir, and E. Samir, ""mystify": A proactive moving-target defense for a resilient sdn controller in software defined cps," *Computer Communications*, vol. 189, pp. 205–220, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422000949

[9] Y. Huang and A. K. Ghosh, *Introducing Diversity and Uncertainty to Create Moving Attack Surfaces for Web Services*. New York, NY: Springer New York, 2011, pp. 131–151. [Online]. Available: https://doi.org/10.1007/978-1-4614-0977-9_8

[10] K. Zeitz, M. Cantrell, R. Marchany, and J. Tront, "Designing a micro-moving target ipv6 defense for the internet of things," in *Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS)*, April 2017, pp. 179–184.

[11] R. W. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*, ser. Addison-Wesley Professional Computing Series. Addison-Wesley, January 1994.

[12] R. Kaur, G. Singh, and S. Khurana, "A security approach to prevent arp poisoning and defensive tools," *International Journal of Computer and Communication System Engineering 2312- 7694*, vol. 2, pp. 431–437, 07 2015.

[13] J. S. Meghana, T. Subashri, and K. Vimal, "A survey on arp cache poisoning and techniques for detection and mitigation," in *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, 2017, pp. 1–6.

[14] D. Stiawan, M. E. Suryani, Susanto, M. Y. Idris, M. N. Aldalaien, N. Alsharif, and R. Budiarto, "Ping flood attack pattern recognition using a k-means algorithm in an internet of things (iot) network," *IEEE Access*, vol. 9, pp. 116 475–116 484, 2021.

[15] E. Rescorla, "Rfc 8446: The transport layer security (tls) protocol version 1.3," USA, 2018.