Romania

University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

Advanced Cybersecurity

MASTER'S THESIS

Securing IoT Networks through Moving Target Defence

Scientific Advisor
Prof. dr. ing. Ion Bica

Graduate
Andrei Vlădescu

Bucharest
2025

Table of Contents

Abstract **iii**

List of Figures **iv**

1 Introduction **1**
1.1 Motivation and Context 1
1.2 Problem Statement 2
1.3 Research Contributions 2
1.4 Thesis Outline 2

2 Background and Related Work **4**
2.1 Security Challenges in IoT devices 4
2.2 Moving Target Defence Fundamentals 4
2.3 Software-Defined Networking Fundamentals 5
2.4 Related Work 6

3 System Architecture **8**
3.1 Threat Model 8
3.1.1 DDoS flood before being connected 8
3.1.2 DDoS flood after being connected 9
3.2 Architecture Overview 9
3.3 Proposed Algorithm 10

4 Testing and Evaluation **13**
4.1 Evaluation Metrics 13
4.1.1 Delay times 13
4.1.2 Failure rate 13
4.1.3 Power usage 13
4.2 Simulation Environment 14
4.3 Simulation Benchmark 14

5 Conclusions **17**
5.1 System Limitations 17
5.2 Future Improvements 17

Abstract

Abstract

This work evaluates the effectiveness of Moving Target Defense (MTD) strategies in bolstering the resilience of Internet of Things (IoT) networks against Distributed Denial of Service (DDoS) botnet attacks. With the rapid adoption of these diverse ecosystems of resource-constrained IoT devices, security gaps have widened significantly, increasing the number of exploitable attack vectors, adding vulnerabilities in said networks. In this thesis we aim to evaluate the application of MTD mechanisms along Software-Defined Networking (SDN) techniques to shift network parameters dynamically, with the aim of disrupting an adversary's ability to perform DDoS attacks.

Inherent to MTD, this approach isn't designed to block an attack outright, but rather make it costlier for the malign actor to sustain it. Through simulated experiments into a public WAN network, we demonstrate that this strategy makes coordinated DDoS campaigns against public IoT infrastructure economically unsustainable, ultimately enhancing the security posture.

Rezumat

Această lucrare evaluează eficiența strategiilor de tip Moving Target Defense (MTD) în consolidarea rezilienței rețelelor de dispozitive Internet of Things (IoT) împotriva atacurilor de tip Distributed Denial of Service (DDoS) lansate de botnet-uri. Odată cu adoptarea accelerată a acestor ecosisteme diverse de dispozitive IoT cu resurse limitate, decalajele de securitate s-au extins semnificativ, crescând numărul vectorilor de atac exploatabili și adăugând vulnerabilități suplimentare în cadrul rețelelor.

În această teză ne propunem să evaluăm aplicarea mecanismelor MTD în combinație cu tehnici de tip Software-Defined Networking (SDN), pentru a modifica dinamic parametrii rețelei, având ca scop perturbarea capacității unui adversar de a desfășura atacuri DDoS.

Prin natura sa, abordarea MTD nu este concepută pentru a bloca direct un atac, ci pentru a-l face mai costisitor și mai greu de susținut pentru actorul malițios. Prin experimente simulate într-o rețea WAN publică, demonstrăm că această strategie face ca desfășurarea unor campanii DDoS coordonate împotriva infrastructurii IoT publice să devină nesustenabilă din punct de vedere economic, îmbunătățind astfel postura generală de securitate.

List of Figures

3.1 Ixia Packet Generator Data Rate Curve 8
3.2 System architecture 9
3.3 Sequence diagram of the connection and data flow 11

4.1 Locust Nominal Statistics 15
4.2 Locust Unprotected DDoS Statistics 15
4.3 Power Usage of ESP8266 Webserver in Different Scenarios 16

1 Introduction

1.1 Motivation and Context

In the recent years, Internet of Things (IoT) devices have transformed both consumer and industrial landscapes by enabling sensors, actuators, appliances, and other smart devices to access the Internet. Today's IoT deployments range from simple home-automation hubs controlling lights or thermostats to large-scale industrial installations or critical-infrastructure monitoring, such as electrical grid sensors. As of 2024, it is estimated that over 18.8 billion IoT endpoints are deployed worldwide, a number projected to exceed 28 billion by 2027[1]. Despite this explosive growth, the majority of these devices remain resource-constrained: they often run on low-power microcontrollers, possess limited memory and compute capacity, and lack rigorous security controls[2]. The result is a highly heterogeneous ecosystem in which an adversary can easily compromise a subset of devices and leverage them for larger-scale attacks[3].

Studies regarding the attack surface and cybersecurity incidents statistics[4][5] show that malign actors favor these devices to penetrate the security of a network, then pivoting to attack the other segments of the network[5]. Also, IoT devices pose a danger by being added to botnets, such as the infamous Mirai botnet[4], whose code was made open-source and is now used by other botnets.

A particularly dangerous attack is the Distributed Denial of Service (DDoS) attack. Traditionally, DDoS campaigns have targeted servers, data centers and cloud infrastructure, consuming available bandwidth or exhausting system resources to render services unavailable. In the context of IoT, however, the attacker's cost for conducting such an attack is reduced significantly, since the device is not going to handle a lot of traffic. As previously stated, a botnet of other IoT devices, such as cameras, routers, or smart thermostats can be enrolled to generate massive volumes of traffic, thereby denying access to the device, or even breaking it. Worse still, the target may itself be an IoT service—such as a building-management system or a remote health-monitoring server—where even a brief outage can have serious real-world consequences. For resource-constrained IoT endpoints, conventional DDoS mitigation techniques (access control rules, firewall rules or signature-based intrusion detection) may not be viable: most off-the-shelf IoT gateways cannot handle high-speed packet inspection, and embedding sophisticated detection logic into every sensor node is simply not feasible.

Against such attack, Moving Target Defense (MTD) has emerged as a promisign mechanism [6][7][8] to increase the attacker's workload and reduce the risk of succesful compromise. MTD seeks to increase the attack surface by introducing uncertainty for the adversary, in any way, shape or form imaginable, ranging from IP changes to VM shuffling. These types of shuffles also foil the attacker's recconaissance and defeat the cyber kill-chain's[9] later phases. When combined with Software-Defined Networking (SDN), it becomes possible to orchestrate the changes at the network level centrally, through rules. Whereas SDN is already used in cloud environmentes, it wasn't yet tested in public infrastructure in the context of securing IoT networks against DDoS attacks.

1.2 Problem Statement

This thesis aims to address the following problems and find a common solution: how can we design and evaluate a relatively lightweight architrecture, that will have an impact against DDoS botnets? Moreover, given the current landscape of IoT ecosystems, is there a possible solution to also make it usable in the public infrastructure, while using IPv4?

To describe a simplistic environment, in today's IoT deployments, a significant challenge arises from the fact that not many devices are available with an IPv6 networking stack, thus being dependent on IPv4, often with NAT gateways. This is an inherent problem of the IPv4, since we cannot use as many IPs without exhausting the address space. This isn't a problem with IPv6, as we can freely have global unique address for each device. This limitation is present when, for example, the system is deployed in the public infrastructure, with an IP provided by the ISP's DHCP server. The IPs are most of the time shuffled when the lease time expires and the router borrows another one. The idea described in this work is that we can use the DHCP's random address leasing as a free IP changing mechanism.

To elaborate more, given that conventional DDoS attacks imply volumetric attacks to bring down services, we can "dodge" these attacks by requesting new IPs from the ISP's DHCP, so that a pool of IPs is not required at all times to perform this switch.

1.3 Research Contributions

We aim to contribute to the larger scientific community by combining MTD and SDN in a specialized environment, where a test microcontroller is the ultimate target of an artificial DDoS botnet, using realistic testing methods. The proposed architecture uses a device-agnostic approach, where the end-user may be free to add other DDoS detection methods than the ones used in this work. Also, the end-user is enabled to use as many layers of indirection to defeat other types of attacks, not tested in this thesis.

We built a simulation environment comprised of Virtual Machines (VMs) and Docker containers, as well as physical devices, to have a realistic testbed. In this testbed we not only measured if the system is protected from volumetric TCP/UDP flood attacks, but also measured key parameters, including failure rates, power consumption and latency.

The SDN architecture is implemented with the help of Python scripts, and the Northbound/Southbound APIs are implemented using MQTT for ease-of-use and later modifications.

1.4 Thesis Outline

The remainder of this thesis is organized as follows:

In chapter II, named "Background and Related Work", theoretical key concepts are introduced to the reader, as a technical foundation for the rest of the thesis. This is taking into account how MTD works as a paradigm, along with some already-used MTD mechanisms, and how SDN is structured. Finally, we survey existing works done in the direction of MTD, in partiucular network-level defenses, and the type of surface dynamic (IP or port hopping) and scope (host-level, network-level).

In chapter III, "System Architecture", we begin by formalizing our threat model and defining security and performance assumptions. We then present detailed design constraints specific to IoT networks (e.g. physical connections and data transfer medium). Next, we describe the proposed SDN-driven MTD framework, including block diagrams, algorithms, and pseudocode detailing how address shuffling and flow-rule updates are orchestrated. Implementation details, such as controller logic, switch configurations, and the mechanism for client notifications, are also covered.

After making defining how the system works, in the fourth chapter, ”Testing and Evaluation”, we define the metrics used to evaluate the solution, including latency, failure rates and power consumption. We describe our experimental environment and outline specific test scenarios. We present statistical results in the form of graphs, then analyze the trade-offs between security gains and performance overhead. We also discuss a few edge cases, such as controller failure and high-churn IoT populations.

The final chapter is reserved for conclusions. We are summarizing how the system performed in accordance with our initial goals. We discuss the main findings and limitations of our current prototype. Finally, we propose ways this work can be improved, such as adding more intelligent DDoS detection systems and how to better test the system in a more realistic scenario.

2 Background and Related Work

In this chapter, we first discuss the security challenges posed by the mass-adoption of IoT devices, then introduce the fundamental concepts behind Moving Target Defence (MTD) and Software-Defined Networking. We conclude this chapter by reviewing existing works in the current literature and state-of-the-art.

2.1 Security Challenges in IoT devices

Studying the current landscape of IoT devices, these can differ in form and function, thus requiring a different approach in how they operate. Such IoT devices often run bare-metal code, a variant of real-time operating systems (RTOS), such as FreeRTOS or VxWorks or even a flavor of embedded Linux distribution, albeit with greatly reduced capabilities and performance compared to a tradional x86. However, all IoT devices have something in common, which is a resource-constrained environment in which they function.

As stated before, IoT devices are heterogenous in their appearance and function, but they are often small devices, with low performance, in terms of CPU power, RAM and storage capacity. More often than not, they don't include displays, and instead rely on other means of interaction, such as a button interface with audio feedback through buzzers, or they implement Bluetooth, WiFi or similar protocols to be used in conjunction with phones. Such a combination of a higher protocol reliance on interaction makes these devices a target for malign actors. Arguably, the most infamous example of IoT devices being attacked and succesfully hacked is the Mirai Botnet[10].

The reality which we face is the IoT global landscape contains a plethora of different IoT ecosystems which are not being maintained and secured by their makers, which in turn weakens the security posture of the individuals or companies that use them. What makes them hard to maintain is their differences in operating systems or codebase, since a security update on a smart-bulb from company 'X' will not reduce the security bugs on, let's say, a smart wallplug, made by company 'Y'. The lack of maintenance and constant updates combined with a lack of standardization, the deployment of these limited devices on a global scale and a diverse forms of connectivity makes them prime targets for malign actors.

2.2 Moving Target Defence Fundamentals

The abstract concept of Moving Target Defence (MTD) long predates the modern age-it may even predate written history. One of the most famous uses of this strategic thinking can be seen in ancient China[11], more specifically in the Late Warring States period in China (3rd century B.C.), where the general Li Mu defeated the armies of Bai Qi, by buying more time to raise more levies. He kept his main force on the move in the borderlands, rotating small detachments through several smaller forts, mountain passes and watchtowers, set dummy concentrations of garrisons and put traps to confuse the attackers. This bought him time to raise enough armies to win the war against the Qin State.

The MTD concept can best be described as the classical ”shell-game”, where a stone is hidden under three shells or cups, and the gambler tries to guess the location of the stone, after a fast shuffle has occured. The definition of an attack's lifetime can be described with the following formulas:

$$t_{\text{attack}} = t_{\text{probe}} + t_{\text{construct}} + t_{\text{launch}} \tag{2.1}$$

$$t_0 + t_{\text{attack}} \in S_{\tau_k} \tag{2.2}$$

$$S_{\tau_k} = [t_{\tau_k}, t_{\tau_{k+1}}) \tag{2.3}$$

As shown in equations 2.1, 2.2 and 2.3[12], the attacker's total time splits into three phases. In Equation 2.1, the attack time is expressed as the sum of periods for proving, constructing the attack and then launching said attack. The successful attack condition is expressed in Equation 2.2, as the starting time, t_0, combined with the time to attack, t_{attack}, represents the time it took for the attack to launch, during the same state of the service, S_{τ_k}. In Equation 2.3, the service state is defined as the extent of time from the period t_{τ_k} until the next period, $t_{\tau_{k+1}}$.

Moving target defenses have been proposed as a way to make the setting more difficult for the attacker to exploit a vulnerable system, by changing characteristics of said system to present attackers with a varrying attack surface[6]. The ultimate goal of a moving target defense is to increase unpredictability in the target, causing difficulties in delivering an attack to the correct (vulnerable) target.

Examples of MTD already used in modern devices and services include defense stratagems in many layers of computing. Address Space Layout Randomization (ASLR)[13] and Instruction Set Randomization (ISR)[14] operate in the lowest levels of computing, whereas Honeypots (decoy nodes) and Honeynets (decoy networks) operate at the network level. We can also consider CAPTCHAs as being part of MTD, since bots find it increasingly difficult, even with machine learning (ML) to solve tricky puzzles. All of these strategies imply that the attacker will either miss the target, or worse, hit a decoy node and signal the other defense measures in place.

2.3 Software-Defined Networking Fundamentals

Software-Defined Networking (SDN) is a networking architecture that decouples the control plane - decision making - from the data plane - packet forwarding[15]. In an SDN architecture, a central controller, which is visible to all network operators, issues flow rules to network switches, based on policies or API calls. This allows a dynamic reconfiguration of forwarding paths without manual(physical) intervention.

SDN uses application programming interface (API) calls, which are a set of standard procedures that define the proper way of communicating beween logical nodes. The control plane is governed by the Northbound API, where upstream applications can control the flows, while the Southbound API is used by the network controller to modify the behaviour of the network switches and routers.

Software-defined networking is used in lots of domains, since networking is a core part of most environments. One use case where it was quickly adopted is in virtualized networks, where a physical network pool of resources is used by separate virtual networks. Other uses include Cloud Computing, by automating the on-demand nature of cloud infrastructure or Content Delivery Networks (CDN).

2.4 Related Work

Research in moving target defence concepts has been conducted by implementing random permutations of resources [16][17][6] which require the movement to be randomized and take place either in predetermined periodic intervals, making it a pro-active approach, or at unpredictable random times, meaning the approach is proactive.

One notable example of an early MTD strategy for defending networks is Mutable Networks (MUTE) [6]. In MUTE, both IP addresses and port assignments are abstracted from the underlying hosts and bound to entries in a software-defined routing table. At each reconfiguration interval, a cryptographically secure algorithm, driven by private keys, reshuffles these virtual addresses and ports, ensuring that the mapping remains unpredictable and consistently synchronized across all switches. To further disrupt reconnaissance, MUTE transparently alters the way hosts respond to probes, making traditional fingerprinting techniques far less effective. To be able to do this, the mappings are controlled globally by the system, thus making portability into a non-100% controlled environment not viable.

MacFarland et al. [8] describe an SDN-based anti-reconnaissance scheme, in which a custom DNS server collaborates with the SDN controller to obscure real host identifiers. When a client issues a DNS query, the controller selects a synthetic IP from its pool and the DNS server returns that address. Simultaneously, NAT rules on the SDN switch translate the fake IP (and a corresponding synthetic MAC) back to the host's actual MAC and IP. By providing each client with a potentially unique mapping, the system makes it far more difficult for an attacker to reliably enumerate live hosts.

Frequency-hopping[18] is a method of transmitting data over the air using electromagnetic waves by changing the frequencies at which the data transfer happens. This concept is widely used in electronic warfare (EW) to resist jamming. Luo et al. [19] adapt the frequency-hopping concept from electronic countermeasures into a network context with their TAP-based Port and Address Hopping (TPAH) scheme. Operating entirely in user space via a TAP virtual network driver, TPAH periodically rotates both the layer-3 IP addresses and the layer-4 port numbers according to a configurable schedule, thereby disrupting DDoS flooding and port-scanning attempts.

Random Host Mutation (RHM) [16] continuously assigns short-lived, randomized IP addresses to hosts. An special device, the MTG gateway, which is based at the network edge, maintains a mapping between these virtual IPs and the hosts' real IPs, and an MT Controller handles inter-host communication. Once a session expires, these mappings are eliminated by the controller. However, because each mutation requires updates to the routing table and the address space may become oversubscribed, this approach can introduce significant overhead and latency as the gateway processes frequent, large-scale routing updates.

Making use of the big addressing space of IPv6, a protection model can be devised, thus making congested IPv4 schemes obsolete. An IPv6, explored and tested approach is Moving Target IPv6 Defense (MT6D)[20], earlier than what the other schemes proposed. This approach is a proactive type, which changes IP address of the sender and receiver periodically during a session - the session is not however interrupted in any way. This protection model makes two implementations available: one is where a host-based embedding is used in software, and the other when it's used inside a gateway device. While the overhead is small, the packet loss does vary depending on the file sizes, since the kernel buffers are used up in the transfers. Latency is also an impediment and the authors believe that re-writing the solution in another low level programming language would solve the problem, as opposed to their Python implementation. Other limitations that are observed include the possibility of colliding IPv6 addresses with other hosts, which, in theory is possible, as the adress space is limited.

Implementations based on Software-Defined Networking, such as those using the OpenFlow protocol, have also been explored in the literature. One such solution is OpenFlow Random Host Mutation (OF-RHM) [17], which incorporates OpenFlow switches alongside dedicated random host mutation (RHM) gateways and a central SDN controller. In OF-RHM, the SDN controller centrally manages virtual IP translations and updates the flow rules on switches accordingly, maintaining continuous and seamless host-to-host communication. This implementation has been further analyzed and validated under modern scenarios, by using SYN flooding attacks to evaluate effectiveness [21]. The cited work benchmarks OF-RHM's performance specifically by monitoring CPU usage during DDoS events. The conclusion of these tests is that OF-RHM reduces processor utilization by a large margin when under attack. However, both studies emphasize a critical trade-off: increasing the frequency of IP address mutations enhances security but proportionally raises system overhead.

Another OpenFlow-based solution is proposed by Gudla et al. [22], where the authors compare network behavior in scenarios with and without MTD enabled. Their findings indicate that during periods of IP address shuffling the network experiences significant jitter spikes in the MTD scenario, whereas jitter levels remain consistently low without MTD. The increased jitter during shuffling events is attributed to temporary congestion and additional overhead, resulting in a measurable increase in packet loss.

While some earlier MTD strategies primarily rely on rotating the IP addresses of protected assets periodically, another possible approach is to alter the network's configuration based on the specific identity of each client. Changes triggered by client identity can be described as a form of "spatial mutation," in contrast to "temporal mutation," where configurations change periodically with time. An approach that integrates both these spatial and temporal mutation strategies is proposed by Jafarian et al. [7]. Through both analytical modeling and practical testing, their study illustrates the effectiveness of combining these mutation strategies in mitigating threats such as reconnaissance, worm infections, and advanced persistent threats (APTs).

To mitigate the overhead typically associated with frequent shuffling in MTD strategies, a streamlined and lower-complexity solution, named SDN-oriented Cost-effective Edge-based MTD Approach (SCEMA)[23], is proposed. This approach prioritizes minimizing deployment and operational costs while preserving or even improving security effectiveness. Evaluated using a Mininet-based setup, SCEMA was shown to effectively enhance unpredictability within the network, significantly raising the attacker's operational costs.

3 System Architecture

3.1 Threat Model

In this work, we assume the adversary aims specifically to bring our simulated infrastructure down using a volumetric DDoS attack, in either TCP SYN Flood or UDP Flood. We also assume that the attacker controls a botnet which wasn't blocked or flagged beforehand, so other security appliances will not intervene to thwart this attack.

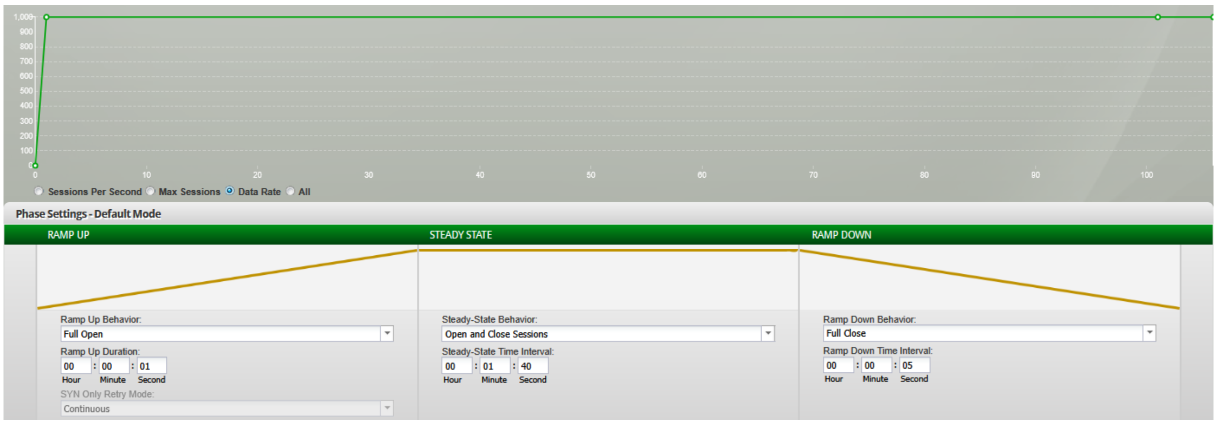
The network medium for the software side is a virtual network, made using VirtualBox's network driver, and also the Docker Network stack. To connect these virtualized networks, we are using a 1Gbps capable SOHO router, to which the IoT target device is connected to. A maximum of 65535 bots will be simulated through the virtualized environment, taking up a whole /16 subnet. We consider the botnet size to be adequate, given that the infamous Mirai botnet had a peak of an estimated 600.000 devices[10], as we consider comparable sizes of botnets to be more frequent. Also, we expect the attacker to be able to generate up to 1Gbps of traffic in total, as the network is expected to impose bandwidth limitations regardless.

The attacker generates the 1Gbps traffic using the Ixia Breakingpoint traffic generator appliance, with the specific traffic curve shown in Figure 3.1[12].

The control plane between the SDN components are not a target for this scenario, and also other attacks that aim to disrupt by hijacking the infrastructure itself are not the aim of this work. The two scenarios being tested are as follows in the next subsections.

3.1.1 DDoS flood before being connected

This case is a standard in the case of DDoS botnets, and is differentiated by the fact that the bots will not try to connect to the system through conventional methods, such as querying the authoritative DNS server, but will try to flood directly to the IP which is obtained by a scanner bot.



Sessions Per Second Max Sessions [✓ Data Rate] All

Phase Settings - Default Mode

Ramp Up Behavior: Full Open
Ramp Up Duration: 00 Hour : 00 Minute : 01 Second
SYN Only Retry Mode: Continuous

Steady-State Behavior: Open and Close Sessions
Steady-State Time Interval: 00 Hour : 01 Minute : 40 Second

Ramp Down Behavior: Full Close
Ramp Down Time Interval: 00 Hour : 00 Minute : 05 Second

Figure 3.1: Ixia Packet Generator Data Rate Curve

3.1.2 DDoS flood after being connected

This is an edge case, when the botnet manages to query the authoritative DNS server, without triggering any security alarm, or DoS detection. In this case, the DDoS is going to hit the returned IP from the DNS server, and the protection is more exposed.

3.2 Architecture Overview

The backbone of the system is structured as such:

- Master node
- Proxy node(s)
- Nest node(s)
- IoT device(s)

Figure 3.2[12] shows how these components are pieced together, and how they interact on an infrastructural level.

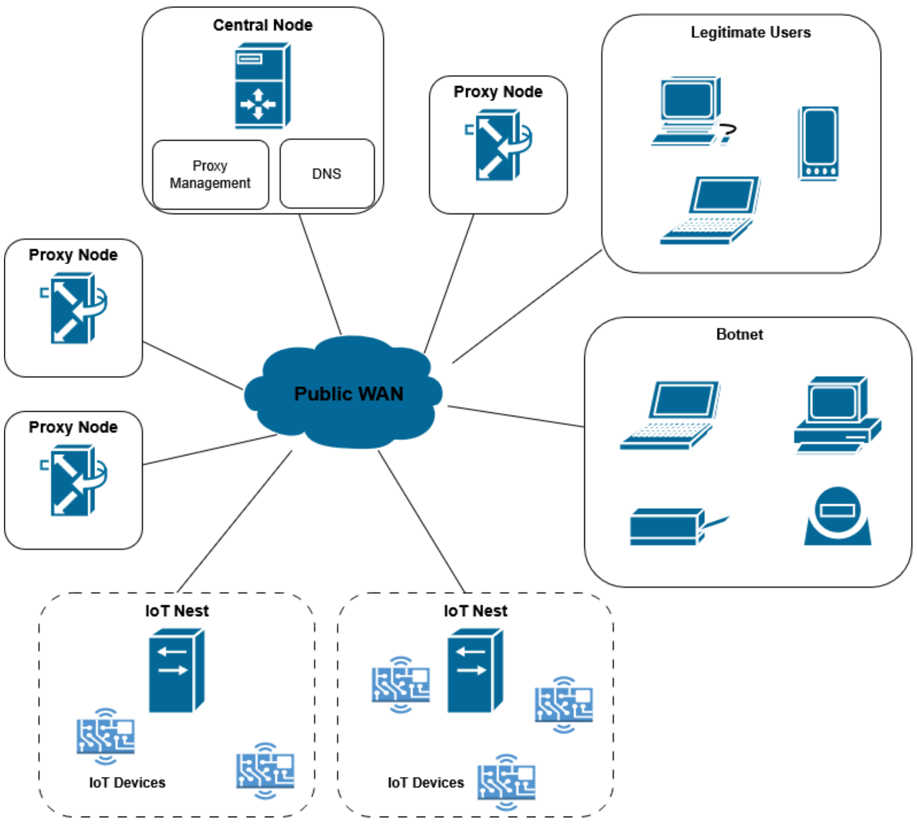


Figure 3.2: System architecture

Starting hierarchically, the master node is what can be considered an SDN controller, since it incorporates the program that manages proxies through the southband MQTT protocol, and in addition handles DNS requests with the help of an embedded DNS server. This master node is considered unique in the system.

The proxy nodes serve as the foundational layer of abstraction between the clients and IoT servers. They conceal the IoT networks and permit only traffic that has been independently approved by the master node. Two proxies may not exist in the same configuration, since their routing tables will not contain any common client-nest pairs. The routing stack configuration is done through nftables, and each proxy perform NAT masquerade, with DNAT and SNAT.

Nest nodes are the final destination between a client's packets and the IoT device. They are configured to accept and forward traffic only from the approved proxies, and serve as the gateway of the IoT LAN. In a more realistic scenario, this gateway may be swapped with a regular IoT gateway/edge router, since IoT devices are known for using IoT-specific transmitting mediums, such as Z-Wave, Thread, Matter, Zigbee, Bluetooth, in the case of consumer IoT devices, or Modbus over IP, RS-485 in industrial/SCADA devices.

The target IoT device being tested is an Espressif ESP8266 development board. We consider this MCU an equivalent for most of the devices on the market as of the time of writing this thesis, and has the following characteristics: 32bit single core Tensilica L106, running at either 80 or 160MHz, and uses the built-in WiFi driver to connect to our nest node gateway. To simulate the service, we are using a combination of temperature reading from the built-in sensor, using the ADC, combined with a generated random number. All of this is hashed using SHA2 256, using calls to the cryptographic engine of the ESP8266 SoC. We believe this is a fair computational workload, since it uses composite resource access, complete with the HTTP server running in a separate FreeRTOS thread.

3.3 Proposed Algorithm

An initial DNS query will take place, before the client may access any resource on the Internet, as it is normal. The recursive DNS server will redirect the query to the autorithative DNS server, embedded in the master node, and this query marks the initial step of our algorithm. When the client C, using it's ephemeral IP from the ISP, C_{IP_K} requests the IP of the service of our controlled domain, D, the recursive DNS server will evenutally lead to the DNS held by our system, the defender side. The master node will then make a decision based on the reputation of the connecting client's IP, if he's been marked before or not. In the case of a "clean" client, it will forward an IP of one of the proxies. The algorithm is also described in Algorithm 3.1[12].

Algorithm 3.1: DNS Resolution with Proxy Assignment and SDN Integration

```
Result: Client connects to IoT Device
$C_{IP_K} \xrightarrow{DNSQuery} S_k$;
$DNS \xrightarrow{IP_K} M_{SDN}$;
if $IP_K \notin IP_{Blacklist}$ then
    if $IP_K \in \{ IP_{Client} \mid (IP_K, IP_{Proxy}) \in P_{Assigned}\}$ then
        return $IP_{Proxy} \mid (IP_K, IP_{Proxy}) \in P_{Assigned}$
    else
        Choose $IP_{Proxy} \sim \text{Random}(\mathcal{P})$;
        Add $(IP_K, IP_{Proxy})$ to $P_{Assigned}$;
        return $IP_{Proxy}$
else
    Close connection;
```

The end result of this algorithm is the client is allowed on the system, and is given the IP of the proxy tied to the final nest gateway. The designated proxy will have been modified by the master node through the MQTT southbount API to allow the traffic through iptables rules with DNAT and SNAT.

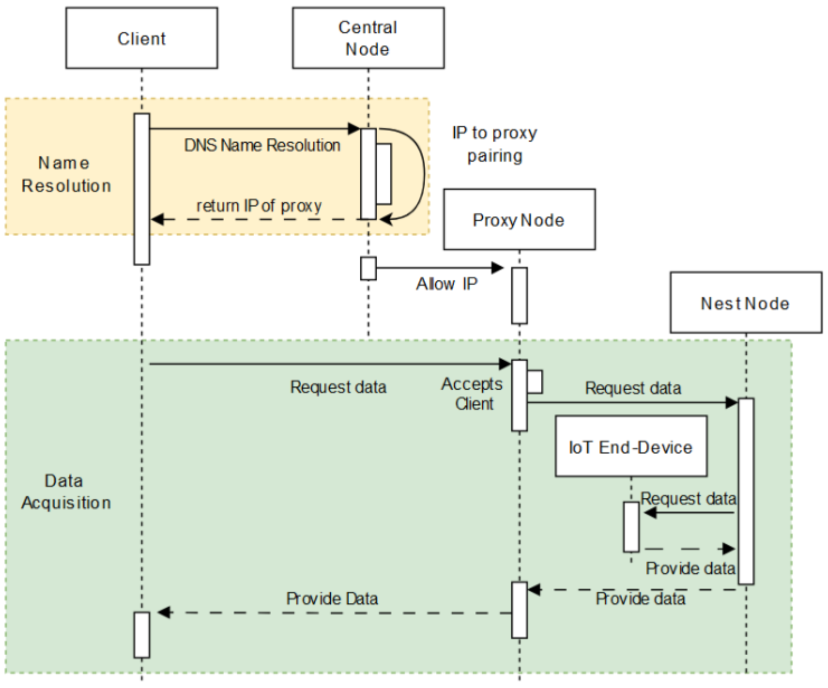


Figure 3.3: Sequence diagram of the connection and data flow

After this connection steps, the client may request data from the IoT server seamlessly, as shown in Algorithm 3.2[12]. Using regular HTTP, for example, he will first make a TCP three-way handshake, but with the proxy, not with the IoT. The IP masquarade will be an opaque wall from the point of view of the IoT nest and the client. After the TCP handshake, the two entities may exchange data on HTTP seamlessly, with an added delay of another hop, practically, since the traffic is relayed. A good way of visualising the whole connection algorithm and data exchange is expressed in Figure 3.3[12], where the yellow part is the connection phase, and the green segment is the data transfer.

Algorithm 3.2: Client-Server Communication Dataflow

```
Result: Client retrieves data from IoT Device
$C_{IP_K} \xrightarrow{\text{GET Request}} P_{IP_{Proxy}}$;
if $IP_K \in P_{Allowed}$ then
    $NAT(IP_K \rightarrow IP_{Proxy}, IP_{Proxy} \rightarrow IP_{Nest})$;
    $IP_{Nest} \iff IP_{IoT}$;
else
    Close connection;
```

A DDoS attack is expressed in Algorithm 3.3[12], where the number of requests, R, is measured with a fixed or dynamic threshold, R_{th}. R is computed as the number of requests in a period of time, $\frac{N_{req}(T)}{T}$. If the threshold is bigger, then a series of events happen: the SDN master node, M_{SDN}, will receive a DDoS alert from the proxy, signaling it will need a new IP address from the ISP. Also, the offending IPs are added to the master node's blacklist, so that the DNS will not forward them an available proxy IP. After receiving a new IP, the service will restart itself.

Algorithm 3.3: Handling of an Attack Through Naive Detection

```
Result: Attacker is interrupted proactively
while Proxy Server Active do
    $R \leftarrow \frac{N_{req}(T)}{T}$ ;
    if $R > R_{th}$ then
        $M_{SDN} \xleftarrow[RenewIP]{\text{DDoS Attempt}} Proxy$;
        $M_{\text{SDN}} \xleftarrow[\text{Blacklist Update}]{\text{IP}_{\text{offending}}} Proxy$;
        $M_{SDN} \xrightarrow[ProviderAPI]{\text{Request new IP}} ISP$ ;
        $IP_{Proxy_{Old}} \xleftarrow{\text{Replace}} IP_{Proxy_{New}}$
```

4 Testing and Evaluation

This chapter assesses the proposed system's performance under a realistic load, complete with attack scenarios. The goal is to evaluate the trade-offs between availability, resilience and latency, under a directed volumetric attack. The target we are using is a NodeMCU development board, with an ESP8266 microcontroller SoC, running an HTTP microservice.

4.1 Evaluation Metrics

The effectiveness of the proposed solution is measured using multiple metrics. Each of these metrics play a different role in determining how viable our solution would be in a real world scenario, from the point of view of the client and the infrastructure.

4.1.1 Delay times

Delay times, or in short, latency, is crucial to the user experience and is measured in both favorable, nominal and stressful conditions. This is what the client experiences when he uses the service under our current system. From his point of view, the increased latency shouldn't be more than what the average service can be accessed, otherwise the user may choose to abandon the service.

In the tests which we conducted we used the timing statistics generated by the built-in function of Locust library from Python.

4.1.2 Failure rate

Beyond raw loading times, another very important metric is the failure rate. Considering our architecture and algorithm, the failure rate presents itself as a problem inherent to our solution only in the case of a DDoS attack. Simply by implementing an IP renewal strategy, the failure rate is not going to be perfect for the legitimate clients. These tests are also measured using the Locust and Requests libraries.

4.1.3 Power usage

Given that most IoT devices are operated in environments with limited power sources, energy efficiency serves a dual purpose - it indicates how much time the IoT device can function, given a limited power supply, such as a battery, and also the performance at which it can run, since this is also tied to the limited power supply problem. Simply by using the system more, the application running on the OS will consume more power, when queried by a client.

The power usage is measured using a laboratory bench power supply-the development board has its power rail tied to the output of the power supply unit (PSU). The PSU can measure the power draw with a 1Hz refresh rate, and can export the statistics to the PC. After exporting the relevant data, a Matlab script was used to compile the results into Figure 4.3. The refresh rate of the PSU is not an impediment, as an average of the power consumption being measured once per second is adequate for capturing macro-level behavior and longer-term trends. While it may not reflect very short spikes in consumption at a millisecond level, the measurement resolution is sufficient for observing the effects of sustained traffic, similar to those caused by a volumetric flood.

4.2 Simulation Environment

The testbed used for evaluation simulates a microservice-based IoT system using a NodeMCU development board, with an ESP8266 microcontroller SoC. The clients are simulated through docker containers, the critical infrastructure is composed of different VMs, and the botnet is simulated both by using the Locust library and with an Ixia Breakingpoint packet generator appliance, capped at 1Gbps traffic.

Four traffic scenarios were considered:

- A baseline, idle system with no active clients, to establish reference power consumption and responsiveness.
- A normal usage pattern, with up to 10 requests per second, to represent a functioning system in typical operation.
- A DDoS using the Locust framework on the IoT device, during which the system is not active, to establish a comparison baseline.
- A final DDoS scenario using volumetric traffic from simulated clients, mimicking the behavior of a large-scale botnet attempting to overwhelm the service.

4.3 Simulation Benchmark

In this section we explain the results of our testing.

In nominal usage of the microservice, we extracted the following statistics from Figure 4.1[12]. The requests per second (RPS) metric is between 3 and 4, even though the total number of users equates to 10. What is observable is that even without a high number of users, there still are failures, which is surprising, given the low number of clients. This shows how vulnerable are IoT devices of this kind to a volumetric DDoS.

A response time of around 60ms is observed as a median, and as an extreme is peaking up to 150ms. This is acceptable, 150ms is not a big delay for an HTTP request, as most websites load in more time than the peak of this device.

As a big contrast, Figure 4.2[12] shows a different statistic, in the case of an unprotected DDoS. The maximum user base is limited to 10,000, with users ramping with a rate of 100/s. The users will experience a big delay, which Locust does not record more than 20s, but this practically shows that the IoT microservice is almost instantly flooded, as the request per second curve matches, albeit in a delay, the failure per second curve.

To test how our system behaves with the security measures active and in-place, we used the Ixia Breakingpoint, with the specific data rate curve expressed in Figure 3.1[12]. This assures that the attackers are generated in a truly realistic scenario, without having software components to throttle the system-the Ixia packet generator can easily generate 1Gbps traffic of HTTP

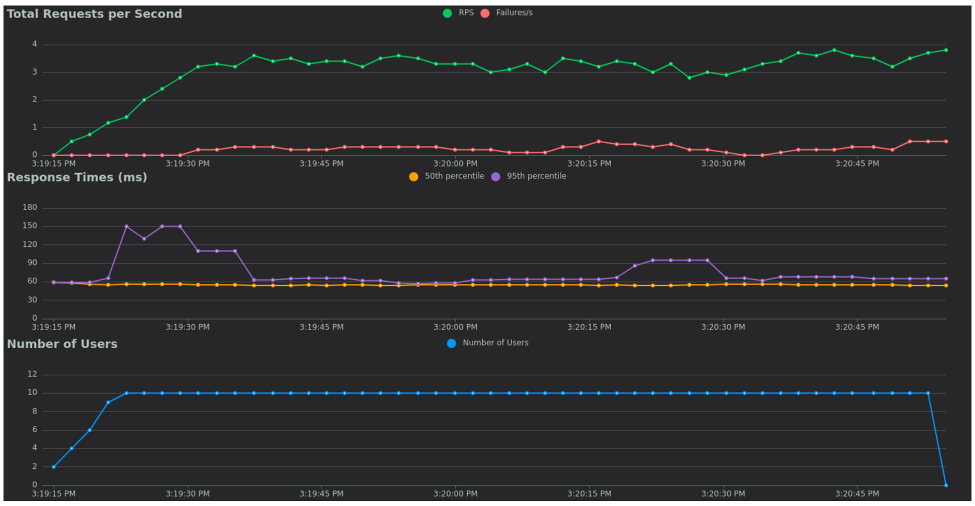


Figure 4.1: Locust Nominal Statistics

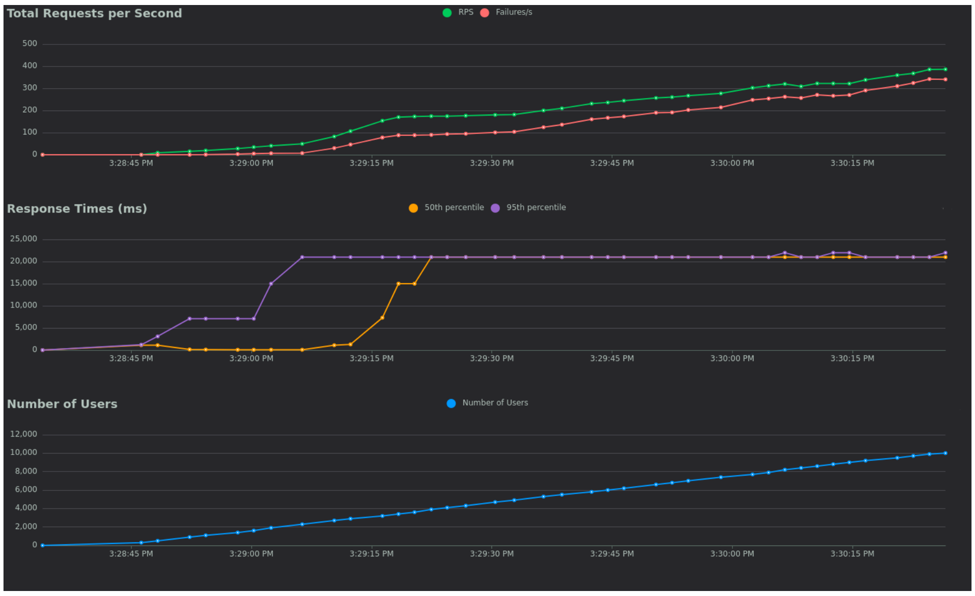


Figure 4.2: Locust Unprotected DDoS Statistics

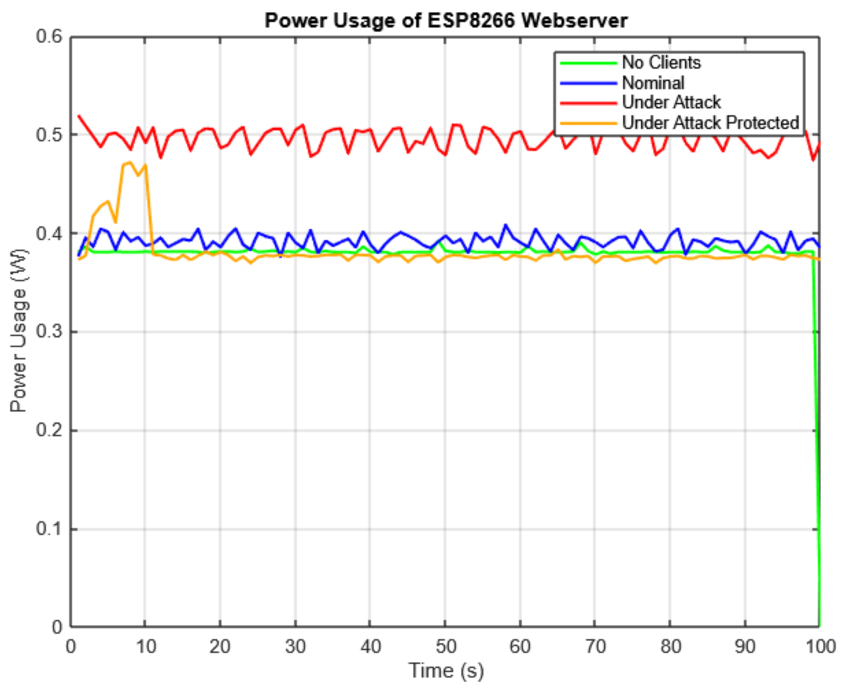


Figure 4.3: Power Usage of ESP8266 Webserver in Different Scenarios

requests. To find just how vulnerable the IoT microservice is, we can consult Figure 4.3[12], where we can see curves of power usage for each of the cases tested.

We can see with green a baseline without clients, where the power draw averages a value almost constant to 3.9 Watts. With blue we can differentiate how clients consume data from the IoT server, since the power usage spikes reach up to 4 Watts.

In the realistic DDoS scenario, we can see the red curve, with jumps above the 5 Watt marker, thus consuming more than 165% power compared to the nominal load. This equates to diminishing the IoT's battery from a nominal usage of one year to about seven months, time in which it isn't available due to the high traffic. This study doesn't take into consideration the dangers of overloading an SoC, but there are certainly problems stemming from continous usage, with an increased temperature footprint. The last green sample is not to be taken into consideration, as it represents a mismatch in data length, not an outlier of data.

When the system protection is activated, the power curve is similar to the one colored with orange. The power usage is slightly shifted, since the green would overlap the orange, we believe this anomaly is due to a measurement error in the decimals of the laboratory power supply combined with other external factors, such as temperature or transient currents. There is a big spike observed in the first 10 seconds, where the system blocks most of the offending hosts, and the IoT can serve legitimate clients.

5 Conclusions

The evaluation results highlight the practical benefits and limitations of deploying this system in an IoT microservice infrastructure. Under simulated DDoS conditions, the system demonstrated an advantage in reducing the sustained attack, compared to an unprotected baseline. The metrics confirm that a moving target defense approach can significantly degrade a coordinated botnet flooding attack. In the resulted tests, the delays remained within acceptable thresholds, suggesting that legitimate clients can access the service with minimal degradation in their experience.

One of the most notable benefits this research contributes is the low barrier of integration, as the defense mechanism operates at the network layer, making use of IP shuffling reactively. Moreover, the services on top of the TCP/IP stack do not require special handling or changes to assure a working state, nor does the system require a pool of reserved IP addresses from the ISP to function properly in the public Internet. This alone qualifies the appraoch as being a viable option in a real-world scenario, as the cost of implementing a defense need to be low.

The power consumption measurement assured that the IoT device can be closely monitored through a side-channel for disruptions and status of the service. We consider this approach crucial in an IoT protection system's benchmark, as it also shows how much the tested system conserves energy during a volumetric flood.

5.1 System Limitations

As a notable observation, the system is not to be used solely without other security measures. MTD is not a concept to supercede conventional security measures, but rather complement them. We can observe this clearly in the big delay between blocking an IP address from the attacker side and the power consumption of the IoT device in Figure 4.3.limited device.

However, the system also introduced measureable trade-offs, the most prominent being an increased overhead in the IP shuffling strategy. Dropping the IP of the proxy suddenly, means that all clients, regardless of their intentions, would be disrupted. The disruption couldn't be established from a realistic point of view, as this requires communication with the ISP and needs special care to happen fast.

Moreover, the system has a big downside, as opposed to conventional DDoS protection methods, such as packet inspection, since it doesn't match the signatures of the received packets to determine if they are malign packets or not. The system at the moment implements a volumetric detection, be it false positive or true positive, meaning that in a usage surge, legitimate clients with an increased request per second rate will be flagged as malign actors trying to destabilize the system.

5.2 Future Improvements

To improve the architecture of this system we propose the adoption of a dedicated DDoS packet-inspector, such as novel machine learning (ML) models. This will significantly increase the reliability of the system, as the threshold of packets is virtually eliminated, and is taken care of by the ML model.

Another modification that would improve the security is to add an intermediary layer of proxies. This wouldn't diminish the result of a DDoS considerably, although it would reduce the efforts of other types of attacks, such as malicious payloads being sent, or reccoinassance campaigns that aim to scan the network and find the IoT device. This additional layer of indirection can be assymetrical and heterogenous, as in using a graph pathfinding algorithm with nodes being proxies, regardless of their usage, or if they are used as ”gateway proxies” - meaning the first layer of indirection of a client.

We also believe that the adoption of IPv6 can help in implementing a hybrid moving target defense approach, in the sense that a block of IPv6 addresses can be reserved, given the big addressing space of IPv6. These addresses would dampen the disruption caused by the IPv4 shuffling, as the ISP needs to allocate from its internal DHCP an unreserved IP address.

Bibliography

[1] I. Analytics, *Number of connected iot devices growing 13% to 18.8 billion globally*, Accessed: June 3, 2025, 2024. [Online]. Available: `https://iot-analytics.com/number-connected-iot-devices/`.

[2] M. Selvaraj and G. Uddin, *A large-scale study of iot security weaknesses and vulnerabilities in the wild*, 2023. arXiv: `2308.13141 [cs.CR]`. [Online]. Available: `https://arxiv.org/abs/2308.13141`.

[3] C. Xenofontos, I. Zografopoulos, C. Konstantinou, A. Jolfaei, M. K. Khan, and K.-K. R. Choo, "Consumer, commercial, and industrial iot (in)security: Attack taxonomy and case studies," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 199–221, 2022. DOI: `10.1109/JIOT.2021.3079916`.

[4] W. Ahmed, "Cybersecurity in the era of iot: A review of vulnerabilities, threats, and mitigation strategies," *Premier Journal of Science*, Jan. 2024. DOI: `10.70389/PJS.100038`.

[5] M. T. Jafar, L.-X. Yang, G. Li, and X. Yang, *Optimal control of malware propagation in iot networks*, 2024. arXiv: `2401.11076 [cs.CR]`. [Online]. Available: `https://arxiv.org/abs/2401.11076`.

[6] P. K. Manadhata, S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, "Moving target defense: Creating asymmetric uncertainty for cyber threats," in *Advances in Information Security, vol. 54*, J. M. Wing, Ed., Springer, 2011, ISBN: 978-1-4614-0976-2. [Online]. Available: `https://libgen.li/file.php?md5=9367b1bdc68f4e65c6a1e8bba2dabbcf`.

[7] J. Jafarian, E. Al-Shaer, and Q. Duan, "Spatio-temporal address mutation for proactive cyber agility against sophisticated attackers," vol. 2014, Nov. 2014, pp. 69–78. DOI: `10.1145/2663474.2663483`.

[8] D. C. MacFarland and C. A. Shue, "The sdn shuffle: Creating a moving-target defense using host-based software-defined networking," in *Proceedings of the Second ACM Workshop on Moving Target Defense*, ser. MTD '15, Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 37–41, ISBN: 9781450338233. DOI: `10.1145/2808475.2808485`. [Online]. Available: `https://doi.org/10.1145/2808475.2808485`.

[9] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," Lockheed Martin Corporation, Tech. Rep., 2011, Accessed: June 3, 2025. [Online]. Available: `https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf`.

[10] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the mirai botnet," pp. 1093–1110, Aug. 2017. [Online]. Available: `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis`.

[11] R. D. Sawyer, "Li mu and the defense of zhao," in *A Military History of China*, D. A. Graff and R. Higham, Eds., Boulder, CO: Westview Press, 2002, pp. 137–140.

[12] I. B. Andrei Vladescu, "Securing iot networks through moving target defence," in *Proceedings of the 25th International Conference on Control Systems and Computer Science (CSCS25), International Workshop on Applications of Moving Target Defense*, Presented at AMTD 2025 track of CSCS25, Bucharest, Romania, May 2025.

[13] H. Shacham, M. Page, B. Pfaff, E.-J. Goh, N. Modadugu, and D. Boneh, “On the effectiveness of address-space randomization,” in *Proceedings of the 11th ACM conference on Computer and communications security*, 2004, pp. 298–307.

[14] G. S. Kc, A. D. Keromytis, and V. Prevelakis, “Countering code-injection attacks with instruction-set randomization,” in *Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 272–280.

[15] N. Feamster and J. Rexford, *Why (and how) networks should run themselves*, 2017. arXiv: `1710.11583 [cs.NI]`. [Online]. Available: `https://arxiv.org/abs/1710.11583`.

[16] E. Al-Shaer, Q. Duan, and J. H. Jafarian, “Random host mutation for moving target defense,” in *Security and Privacy in Communication Networks*, A. D. Keromytis and R. Di Pietro, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 310–327, ISBN: 978-3-642-36883-7.

[17] J. H. Jafarian, E. Al-Shaer, and Q. Duan, “Openflow random host mutation: Transparent moving target defense using software defined networking,” in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN ’12, Helsinki, Finland: Association for Computing Machinery, 2012, pp. 127–132, ISBN: 9781450314770. DOI: `10.1145/2342441.2342467`. [Online]. Available: `https://doi.org/10.1145/2342441.2342467`.

[18] Z. Kostic, I. Maric, and X. Wang, “Fundamentals of dynamic frequency hopping in cellular systems,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 11, pp. 2254–2266, 2001. DOI: `10.1109/49.963811`.

[19] Y.-B. Luo, B.-S. Wang, X.-F. Wang, X.-F. Hu, and G.-L. Cai, “Tpah: A universal and multi-platform deployable port and address hopping mechanism,” in *2015 International Conference on Information and Communications Technologies (ICT 2015)*, 2015, pp. 1–6. DOI: `10.1049/cp.2015.0230`.

[20] M. Dunlop, S. Groat, W. Urbanski, R. Marchany, and J. Tront, “Mt6d: A moving target ipv6 defense,” in *2011 - MILCOM 2011 Military Communications Conference*, 2011, pp. 1321–1326. DOI: `10.1109/MILCOM.2011.6127486`.

[21] R. Swami, M. Dave, and V. Ranga, “Mitigation of ddos attack using moving target defense in sdn,” *Wireless Personal Communications*, vol. 131, no. 4, pp. 2429–2443, 2023.

[22] C. Gudla and A. H. Sung, “Moving target defense application and analysis in software-defined networking,” in *2020 11th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2020, pp. 0641–0646. DOI: `10.1109/IEMCON51383.2020.9284847`.

[23] A. Javadpour, F. Ja’fari, T. Taleb, and M. Shojafar, “A cost-effective mtd approach for ddos attacks in software-defined networks,” in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 4173–4178. DOI: `10.1109/GLOBECOM48099.2022.10000603`.