Analyzing the Impact of Rebooting and Network Degradation on IoT Botnets using Timed Automata

Alvi Jawad, Luke Newton
Systems and Computer Engineering Department
Carleton University, Ottawa, Ontario
{alvi.jawad, luke.newton}@carleton.ca

Abstract— **The Internet of Things (IoT) has stimulated the creation of a new era with the promise of ubiquitous connectivity aided by a proliferation of inexpensive IoT devices. The threat presented by the recent rise of botnets and their capability to infect numerous vulnerable IoT devices on the internet mandate a better understanding of their inner workings and investigation of alternative defensive mechanisms usable by resource-constrained IoT devices. In this project, we aim to model and simulate the dynamic behavior of the Mirai botnet as a network of timed automata using the modeling tool UPPAAL to delve deeper into the botnet infection process. Additionally, to determine its feasibility as a defensive measure, we examine the effectiveness of rebooting and network degradation on varying system configurations. The resulting formal analysis provides a solid understanding of the impact of such mechanisms on botnet growth and insight into their efficacy to weigh them against other available countermeasures.**

Index Terms—**Internet of Things (IoT), Botnets, Network Security, Modeling and Simulation, Timed Automata, Impact Analysis, Rebooting, UPPAAL**

I. Introduction

The Internet of Things (IoT) refers to the interconnected network of a vast number of internet-enabled physical objects around the world. The increased availability of low cost, low power sensors, and enhanced communication methods has caused an explosion in the number of IoT devices in recent years [1]. With this increased communication, data collection, and analysis, IoT has the potential to improve many facets of society, including, but not limited to, healthcare, infrastructure, supply chains, and the general home and office environments. Unfortunately, the relatively low computational resources available for typical IoT devices, coupled with a rushed productive mentality, have made the IoT infrastructure vulnerable to a number of security threats [2].

The sudden proliferation of IoT devices has also garnered the attention of vulnerable device herders known as botmasters. These attackers use the inherent and user-induced vulnerability present in exposed end-points in the IoT infrastructure to take control of these devices to later use them for nefarious purposes. One prime example is the Mirai botnet attack that affected millions of poorly configured IoT devices to stage a massive distributed denial-of-service (DDoS) attack in 2016 [3]. The last few years have seen a surge in the frequency and sophistication of botnet attacks, resulting in varying levels of impact on the individual devices, services, and the network alike [4], [5].

The defense against botnets involves the meticulous inspection of the intricate interaction among the entities inside the botnet. The Mirai botnet infrastructure, the way devices are infected, controlled, and used to infect or perform other malicious actions, have been studied in great detail by the security community [4], [6]. However, the analyses performed on the impact of such botnets, including Mirai, are mostly evidentiary, meaning we do not have sufficient information to predict the aftermath in the event of a botnet reaching an unprecedented size. Furthermore, many gray areas remain in determining all the distinct types of devices that were affected, limiting the flexibility and extensibility of such analyses.

Formal models have proven to be quite useful in assuring the security of a system. The development of a formal botnet model can considerably improve the understanding of botnet capabilities and how best to deal with them. The advantages of using a formal specification language are manifold: the correctness, as well as various properties of the model, can be verified using the full range of available formal methods and discover areas of ambiguity in the literature. Furthermore, we can model both individual device behaviors as well as see how different distributions of clusters of such devices affect the infection process, and in turn, the impact of botnets. Most importantly, the high decidability of specific modeling formalisms allows achieving highly accurate results and predictive capabilities.

The main contributions of this project are as follows.

- Modeling the individual entities in the Mirai botnet infrastructure and the behavior of different device clusters in an IoT network using timed automata.
- Simulation of the real-time infection and propagation of botnets in a network of heterogeneous devices with varying distributions to estimate the impact.
- Formal analysis of the effectiveness and feasibility of rebooting and network degradation as proposed measures to curb the spread of botnets under varying scenarios.

The rest of this paper is structured as follows. Section II discusses the modeling terminologies and concepts used for the remainder of the paper. Section III examines botnets and their malicious capabilities, and section IV outlines the botnet infection process along with our developed model. Section V details simulation parameters and the results from specific simulation runs, while Section VI discusses the implications of those results. Section VII discusses the challenges faced during the project. Section VIII provides an overview of pertinent botnet studies to date. Finally, Section IX concludes our work with a brief discussion of possible future extensions.

II. Background

In this section, we introduce the timed automata modeling formalism, the UPPAAL modeling tool, and some relevant terminologies and concepts used throughout this paper.

A. Timed Automata

Timed automata is a hybrid mathematical modeling formalism in which a finite set of real-valued clocks are used to represent continuous time in a discrete-event system [7]. Any real-time system whose behavior involves a predetermined set of actions can be represented as a network of timed automata. The system can be decomposed into its constituent components, and each component can be modeled as a timed automaton, which is a finite-state machine extended with clock variables [8]. A timed automaton is a tuple (L, lo, C, A, E, I) where,

L is a set of locations
$lo \in L$ is the initial location
C is the set of clocks
A is a set of actions, co-actions, and the internal τ-action
$E \subseteq L \times A \times B(C) \times 2c \times L$ is a set of edges
$I : L \to B(C)$ assigns invariants to locations

A timed automaton accepts timed words; infinite sequences in which each symbol is associated with a real-valued time of occurrence [7]. At the simulation start, every single automaton in a network of timed automata begins with all of its clocks initialized to zero. The elapsed time is reflected in the change of clocks, representing the advance of time in reality. All clocks are independent of each other and can be reset at each transition to keep track of the elapsed time since the last reset. A transition may only be taken if the associated time constraint imposed upon that transition is satisfied by the current clock values. This feature allows us to model the timing properties of real-time systems and capture interesting qualitative and quantitative aspects such as periodicity, bounded response, and timing delays.

B. Modeling in UPPAAL

UPPAAL is an integrated tool environment that supports the modeling, validation, and verification of real-time systems as networks of timed automata [8]. The model can be extended with data types (e.g., bounded integers, arrays) while also allowing model checking for verification and validation of system properties. This academic endeavor, jointly developed by Uppsala University in Sweden and Aalborg University in Denmark, has received considerable research community support for research on timed automata in recent years. Our work uses the UPPAAL 4.1 development snapshot available under a free academic license[1].

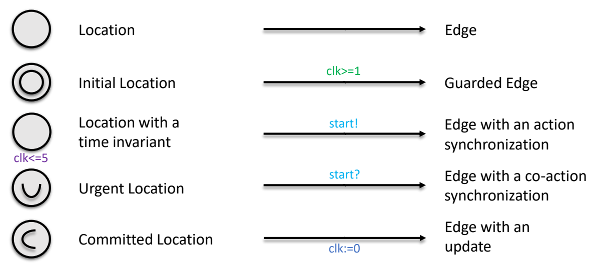

Fig. 1. UPPAAL graphical notations

Figure 1 shows some basic graphical notations used to model real-time systems in UPPAAL. Circles denote regular *locations* that represent the state of an automaton performing a specific function. The *initial location* (only one) of every automaton is depicted by an additional inner circle, specifying the starting behavior of that automaton when the system starts. Regular and *initial locations* can be accompanied by a time invariant (depicted in violet text), which are time constraints imposed upon the location. A location with a time invariant must be left within the specified time constraint. Locations are also called *states*, which is the term we will be using extensively throughout the paper.

Urgent states, denoted by an additional U inside the circle, represent states that are time-critical. These states cannot have any time invariants, and no time can pass while an automaton is in such a state. However, if any other automaton in the network is capable of performing an instantaneous action, that action is allowed. *Committed states* (marked by an additional C) are even more restricted states that allow neither the passage of time nor any action by any other automaton unless that state is left.

Edges, depicted by arrows, represent the transition from one state to another. UPPAAL uses a handshaking mechanism to synchronize communication between different automata. During a transition, an *edge* can send an action synchronization message (depicted in light blue text). This action message (`start!`) must synchronize with another *edge* with a corresponding co-action synchronization message (`start?`), enabling the latter transition to take place.

Individual automata can use state variables to record and share their state with other automata in the network. In addition, clock variables can be used to record the passage of time and impose timing constraints on locations and transitions. *Guarded Edges* represent guarded transitions, where the transition is only allowed once the conditions (depicted in green text) imposed using state and/or clock variables on the transition, are satisfied. Additionally, edges can be accompanied by an *update* command (depicted in blue text)

[1]http://www.uppaal.org/

that updates the value of a state and/or clock variable once the transition is completed.

III. Botnets in IoT

Botnets are one of the major security issues faced in the current IoT landscape [9]. Recent years have seen considerable growth in both original and different variants of botnets, causing a significant impact on various aspects of IoT networks [4]. This section summarizes the impact of recent botnet attacks and explores the threat landscape of modern IoT networks.

A. The Impact of Botnets

A botnet is a network of devices compromised by malware that can be instructed by a botmaster through a Command and Control (C&C) infrastructure. Although individual bots in an IoT botnet do not pose a threat, a large enough botnet can have catastrophic impacts. One prime example of this is the Mirai botnet, which launched Distributed Denial of Service (DDoS) attacks against security blog KrebsOnSecurity and French cloud computing company OVH, with malicious traffic peaking at 620 Gbps and 1.1 Tbps respectively in 2016 [4]. More recently, the largest ever DDoS attack was targeted at Amazon Web Services in February 2020, which saw sustained traffic at 2.3 Tbps [10].

Contrary to the majority of exploits involving DDoS attacks, botnets are not limited to DDoS attacks alone. The following lists several malicious capabilities of botnets [4], [9]:

1) DDoS: A large enough botnet can flood network endpoints or links with enough traffic to severely degrade or completely disallow legitimate traffic through the targeted location.
2) General bot traffic: Continuous communication and propagation consumes network bandwidth and can result in decreased performance in infected devices.
3) Spam or Malware dissemination: Rather than sending all bot traffic to one location like in a DDoS attack, botnets can also be used to distribute malicious payloads to a wide variety of targets.
4) Firmware corruption: Botnets like BrickerBot, once commanded, can access and destroy a device's firmware.

B. The Threat to IoT

IoT, with its sheer number of vulnerable connected devices, has become a tempting target for botmasters. At its peak, Mirai may have held up to 400,000 connected devices, hinting at a small picture of what could happen if all of these infected devices were used simultaneously to perform an attack [4]. Off-the-shelf IoT devices are rarely designed with security in mind, and their relatively low computational resources mean that defenses used by more conventional computers, such as anti-malware software, are not feasible to implement. Moreover, a large proportion of IoT devices are deployed with their initial configuration and easily guessable weak and/or default credentials, and the embedded nature of these devices can make it challenging, often impossible, to patch vulnerabilities. All of these come together to make IoT devices ideal candidates for botnet infection and present a greater need for research into botnet behaviors and the evaluation of possible defenses.

Rebooting has been suggested as an effective defensive mechanism to control the spread of botnets [11]. While botnets can be used by the botmasters for their sole gain, they can also lend the attacking capabilities of a large-enough botnet to external interested parties in exchange for something valuable [4]. The precondition behind this contract is the possession of a large attack-capable botnet and the maintainability of the botnet size over an extended period. One core theme of our analysis of the botnet infection process is to see whether rebooting alone is capable of preventing the attacker from amassing and/or maintaining their attacking capability over a certain period. If so, we wish to see what rate of rebooting is the most effective and whether such a frequency is feasible for an actual IoT device.

IV. Modeling the Infection Process

To model the botnet under study, Mirai, with the final objective of performing our feasibility analysis, we first need to understand the Mirai infection process, all the entities involved in the process, and how these entities interact with each other. In this section, we detail the Mirai infection process and present our timed automata model of the Mirai botnet developed in UPPAAL.

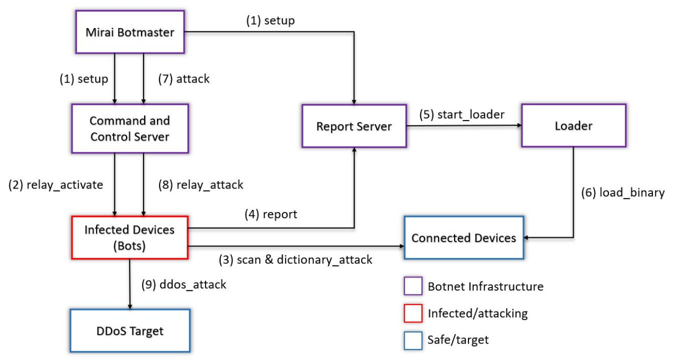

Fig. 2. Mirai Infection Process

A. Mirai Infection Process

Mirai falls under the category of classical centralized botnets that revolve around a single central C&C server to spread [12]. Figure 2 illustrates the primary components of the centralized infrastructure (a C&C server, a Report server, and a Loader) as well as the way these components interact. These entities are under the control of the Mirai botmaster, who actuates and directs the botnet activity through the infrastructure. The botnet is also comprised of the devices that have already been infected, also known as bots, that actively seek out other vulnerable connected devices in the network to infect. The bots under control can be further directed to compromise a DDoS target with a variety of DDoS attacks under Mirai's disposal [5].

The Mirai infection process can be outlined in a number of simplified steps [5], [6]:

1) The botmaster begins the infection process by setting up the remote C&C server and the Report server.
2) The C&C server activates the first bot, one that is infected from the beginning, to initiate the scan-attack-report cycle of a bot.
3) The bot statelessly scans the network for open Telnet ports at randomly selected IP addresses, and initiates a brute force remote-login attempt by randomly selecting 10 out of the 62 default or commonly used credentials hard-coded in the Mirai binary.
4) If a brute force attack is successful, the attacking bot reports the victim *id* (device IP and the credentials that worked) to the Report server.
5) If the reported device is not part of the botnet, the report server adds it to its records, and subsequently, activates the Loader to load the malware binary into the new vulnerable device.
6) The Loader determines the architecture of the target device and uploads a hardware-specific malware, transforming the target device into another bot. Steps 3 to 6 continue until the botnet is large enough for the attacker's purpose or about all the vulnerable devices in the network have been compromised.
7) Any time, ideally with an adequately sized botnet, the botmaster can issue an attack command to the C&C server.
8) The C&C server relays the attack command to all the bots that are currently under control.
9) The bots stop their usual scan-attack-report cycle and start performing a DDoS and/or other malicious attacks on the attack target.

B. Modeling the Mirai Botnet Infrastructure

We start by modeling the key entities - the C&C server, the Report server, and the Loader - in the Mirai botnet infrastructure. Each entity is modeled as a separate automaton that engages in multiple channel synchronizations to reflect the behavior of a botnet in a real-time network. For a brief discussion about the notations used extensively throughout the modeling process, please refer to section II-B.

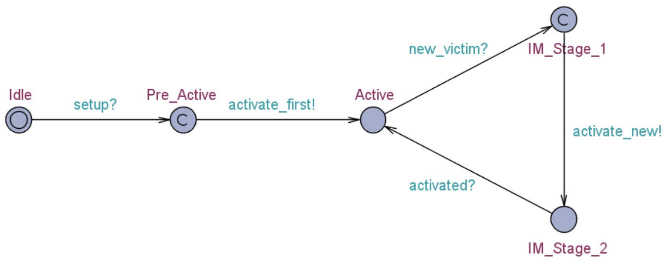

Fig. 3. The Command and Control Server Automaton (CNC)

Figure 3 presents the Command and Control server automaton (CNC) with its two primary states `Idle` and `Active`. The server is set up by the abstract Mirai botmaster entity, after which CNC activates the first bot (pre-infected) and transitions to its `Active` state. In this state, CNC waits for raw socket connections from new victims, only to activate them later so they can start their own infection process.

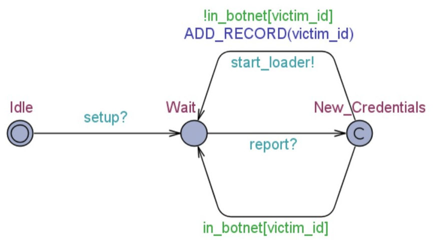

Fig. 4. The Report Server Automaton (RPT)

Figure 4 illustrates the behavior of the Report server automaton (RPT). Similar to CNC, RPT is also set up the Mirai botmaster, allowing the transition from the `Idle` state to the `Wait` state. Once in the `Wait` state, the RPT listens for incoming reports of a new victim *id* sent via each bot on port 80 every minute. Any new *id* is checked against the current records. LDR adds a new record as well as activates the Loader automaton if the *id* proves to be a vulnerable device that is yet to be infected.

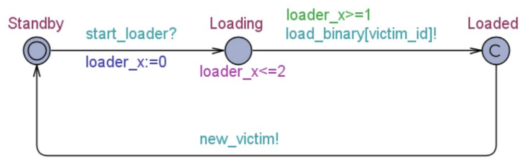

Fig. 5. The Loader Automaton (LDR)

Figure 5 shows the Loader automaton (LDR) that reflects the inner workings of the proxies called loaders [5]. LDR moves from its initial `Standby` state to the `Loading` state after it has received the start_loader command from the report server along with a vulnerable victim *id*. In the `Loading` state, LDR takes anywhere between one to two time units to log into the victim using the provided *id* and instructs the victim to download and execute the hardware-appropriate binary, essentially turning the victim into a functional bot. After this process is over, LDR informs CNC about this new botnet member and returns to its `Standby` state.

C. Modeling Always-Connected Devices

After the botnet infrastructure, we focus our attention on modeling the individual device behavior inside the botnet. The device behavior here refers to the behavior when the device acts as a normal device as well as when the device is infected and being used by the botmaster for their nefarious purposes. Figure 6 shows our first device automaton (DEV_A) where the states involved in the three different modes of behavior - (1) when the device behaves like a regular device, (2) when it is being infected, as well as (3) when it is acting as a bot - are marked in colors blue, orange, and red, respectively.

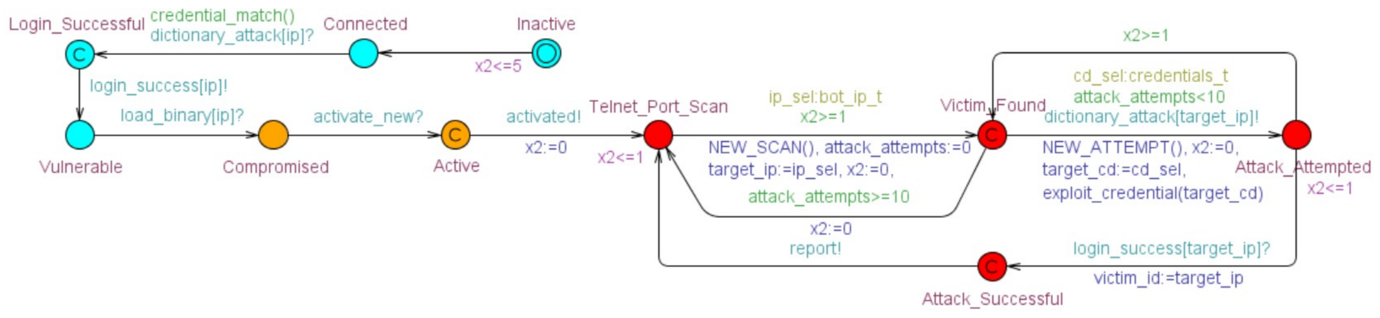

Fig. 6. Device Automaton A (DEV_A) - Always-Connected

Always-Connected devices, according to RFC 7228, are representative of IoT devices that are always-on (P9) and stay connected to the network all the time [13]. All the states marked in blue represent the regular and intended behavior of a device. DEV_A starts in the `Inactive` state and connects to the network after a specific time (within five time units of the simulation start in our model), moving to its `connected` state. This is the only state where another DEV_A in their infected mode (red) can perform dictionary attacks on a DEV_A. If an attack is performed with a matching credential, then DEV_A moves to its `vulnerable` state where it now has its *id* leaked to the report server.

States depicted in Orange represent the transition state of a device during its infection process. In the `Vulnerable` state, the device can be logged in to download the Mirai binary into the device. Once this process is complete, DEV_A moves to the compromised state, where it waits to establish a raw socket connection with the C&C server to receive further commands. After establishing the connection, DEV_A moves to the `Telnet_Port_Scan` state where it now starts to act upon the commands relayed by the C&C server. The red states in the right portion of DEV_A represent the same device's behavior once it is infected, activated, and has started performing activities befitting that of a bot. In the `Telnet_Port_Scan` state, the infected device now statelessly scans the network for pseudorandom IPv4 addresses for open Telnet 23 ports, and once found, moves to the `Victim_Found` state. This is the state where a bot tries to gain a working Linux shell, attempting to randomly select and use 10 out of 62 hardcoded default or commonly used credentials stored in the original Mirai binary [5]. An unsuccessful attack chain of ten takes the bot back to the `Telnet_Port_Scan` state, whereas a successful attack leads to the `Attack_Successful` state first, where the victim information is reported to the Report server.

D. Modeling Reboot-Capable Devices

Next, we model the behavior of devices that are capable of rebooting and reconnecting to the network once rebooted. This capability is crucial as the Mirai binary is stored in the dynamic memory and is cleared once an infected device is rebooted, essentially restoring the regular device behavior [11]. To introduce rebooting to our device automaton, we present a modified automaton in figure 7 with this capability.

Our second device automaton (DEV_B) represents clusters of *Reboot-Capable* devices that can reboot either periodically or manually with user intervention. One particular example of this would be router configurations that allow a router to preserve its health by turning off for an hour each day when not under use. Devices of category E1 (battery-powered devices with a fixed primary battery replacement interval) and P0 (normally turned off, only attached to the network when needed), also fall under the same category and can be modeled with simple attunement of the timing constraints [13].

DEV_B is extended with two new adjustable parameters for rebooting. Each device of type DEV_B now starts with a *reboot_time* that represents a predetermined time after which each device will turn off, moving to the `Rebooting` state regardless of its infection status. Additionally, another parameter, *reboot_length*, determines how much time a device takes to reboot and/or the time a device stays off after turning off. These parameters can be adjusted all at once or manually before each simulation, and the fact that we can represent stochastic timing behavior gives us great flexibility in modeling the various device clusters mentioned before. Moreover, this lets us model different distributions of various devices of type DEV_B, or DEV_A and DEV_B combined, and allows us to extend the model with other device automatons (e.g., with patching capability) in the future.

V. Experimental Results

Finally, we perform simulations to verify the correctness of the model as well as to simulate an IoT network with varying network configurations. In this section, we present the experimental results of our simulation runs using the model developed in the previous section.

A. Device Type Comparison

In all simulations, all devices connected to the network are considered to be using a weak and/or default password, and as such, are vulnerable to a Mirai infection. All infected devices will use a malware binary with a dictionary consisting of 62 commonly used credentials to attack. Unless otherwise mentioned, all devices are one of two types: *Always-Connected* or *Reboot-Capable*, as described in sections IV-C and IV-D.

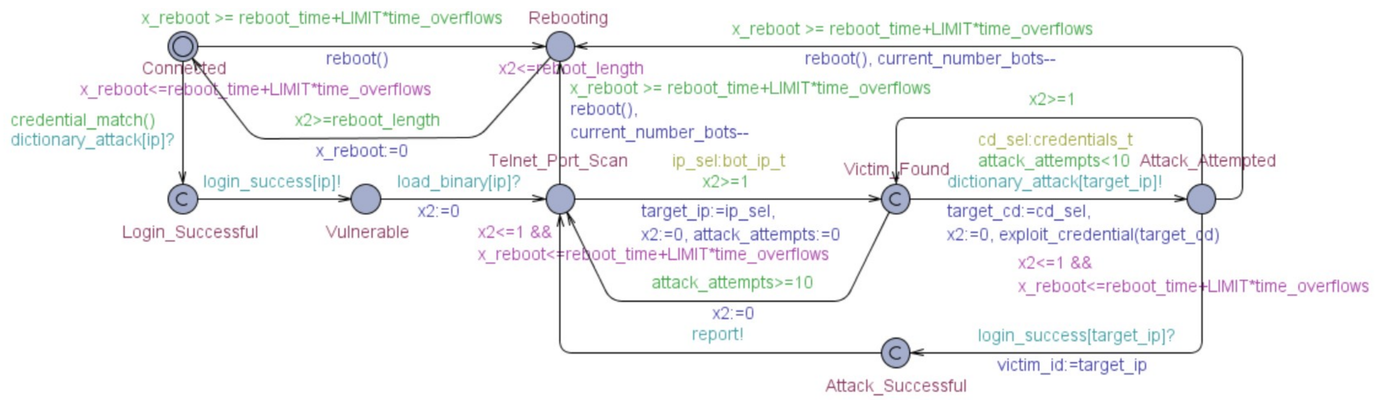

Fig. 7. Device Automaton B (DEV_B) - Reboot-Capable

TABLE I
SYSTEM PARAMETERS

Parameter	Default Value	Other Values Used
Number of devices	100	250, 500
Round Trip Time	100ms	1s
Simulation time	1 day	1 week
Reboot frequency	24hr	1hr, 30min, 10min, 5min
Percentage of time bots propagate malware	100%	50%, 10%, 1%

In addition to these constant parameters, Table I summarizes the adjustable parameters in our model. We perform multiple sensitivity analyses by capturing the model behavior with all parameters set to default and then adjusting one parameter at a time to observe variations in behavior. This allows us to examine how different attributes of the malware, or that of the device network, impact botnet growth.

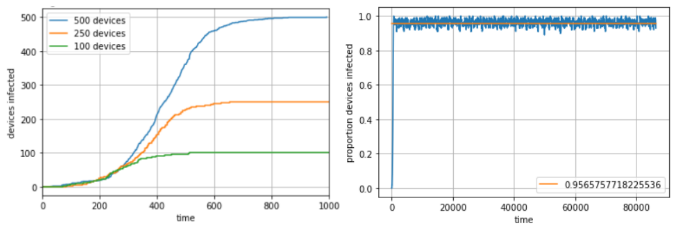

Fig. 8. Botnet Growth for Always-Connected Devices (left) and Reboot-Capable Devices (right)

Figure 8 shows the typical behavior of both types of devices. *Always-Connected* devices maintain their compromised state once infected, producing the expected classical logistic malware growth curve [14]. Once all devices have been infected, no new behavior can occur, so this device type is simulated for a shorter time. *Reboot-Capable* devices, on the other hand, can remove their infection through rebooting as this wipes the in-memory malware on the device [6]. This produces a graph with a similar logistic growth at the beginning, which eventually settles into a steady-state as devices begin to reboot. This steady-state value is captured for most of the experiments conducted, as this provides an indication of how successful a specific frequency of rebooting is in preventing the botmaster to maintain a certain percentage of devices under their control.

Simulation time is measured relative to the round trip time (RTT) parameter. In Figure 8, the RTT is one second, so when the *Always-Connected* devices are all infected within 1000 time units, this is equivalent to 1000 seconds, or 16.67 minutes. The simulation for *Reboot-Capable* devices runs for 86400 time units, which for an RTT of one second, is equivalent to one day.

B. Impact of Rebooting on Percentage of Infected Devices

Table II presents the average percentage of infected devices for two networks of 100 devices with different RTTs. Uptime is calculated as the percentage of time a device is connected to the network. For example, a ten minute reboot period means that each device will reboot ten minutes after being connected, and with a fixed reboot length of one minute, has a 90.91% uptime compared to the total eleven minutes of simulated time. The results indicate that as rebooting is performed more frequently, the average percentage of infected devices decreases, and bots on a slower network are impacted more severely by increased levels of rebooting.

TABLE II
AVERAGE PERCENTAGE OF INFECTED DEVICES FOR DIFFERENT REBOOT PERIODS

Reboot Period	Uptime	Average percentage of devices infected (100ms RTT)	Average percentage of devices infected (1s RTT)
24hr	99.93%	99.9%	99.0%
1hr	98.36%	97.8%	95.6%
30min	96.77%	96.0%	92.1%
10min	90.91%	89.6%	76.3%
5min	83.33%	80.7%	46.9%

C. Stealthy vs Active Botnets

It may be advantageous for a bot to strike a balance between the percentage of time it spends propagating malware, and the percentage of time it stays idle. *Active* botnets such as Mirai [4] spend more time propagating themselves, so they can grow faster but risk being detected more easily, while *stealthy* botnets will grow slower but are harder to detect. Table III reports how different levels of activity, reboot period, and network speed impact the size of a botnet. Generally, botnets with lower levels of activity are shown to achieve lower levels of infection, however, bots can be very stealthy without losing much of their propagation capability. As with rebooting, an increase in RTT results in a more dramatic change in the percentage of infected devices as the activity level changes.

TABLE III
AVERAGE PERCENTAGE OF INFECTED DEVICES FOR DIFFERENT LEVELS OF BOTNET STEALTH

Percentage of time propagating malware	Reboot Period	Average percentage of devices infected (100ms RTT)	Average percentage of devices infected (1s RTT)
100%	1hr	97.8%	95.6%
50%	1hr	97.7%	93.2%
10%	1hr	95.8%	69.6%
1%	1hr	71.5%	0.0067%
100%	24hr	99.9%	99.0%
50%	24hr	99.7%	98.7%
10%	24hr	99.4%	97.8%
1%	24hr	97.4%	86.0%

VI. DISCUSSION

We can see from Table II that for the more realistic network RTT of 100ms, the botnet size consistently remains very high even with frequent rebooting. This is a major problem for the high availability requirement of IoT networks, as even with network devices offline about 17% of the time, the botnet is still able to infect above 80% of devices. This frequency of rebooting is likely already undesirable for many IoT applications, with more frequent rebooting only further degrading the functionality of the uninfected devices. It is, therefore, evident that rebooting on its own is not effective as a primary defense against botnets; something more preventative such as blocking ports or changing default credentials would likely be much more effective.

The fact that a botnet can effectively hide its malicious presence on a network by simply remaining idle yet still commanding a large number of bots to perform an attack anytime is extremely concerning. In our experiments, where a botnet remains idle 99% of the time on a network with one second RTT and hourly rebooting, the botnet cannot effectively spread, showing that stealthy botnets can be thwarted with these basic defenses. However, the level of botnet activity is outside the control of the IoT network designer, and without pertinent information, should not be expected to be low.

Both Table II and Table III show how network speed affects the growth of a botnet in various scenarios. In Table II, the slower one second RTT network with very frequent rebooting is able to slow the botnet growth to an average size that may not be enough to perform an effective attack, while the slower network in Table III is able to completely stop the stealthiest botnet in its tracks. It may be possible for a network designer to throttle their network to curb botnet spread, but this will impede the IoT network capability bringing us back to the security vs. performance issue. A network with one second RTT will likely not provide the level of service necessary for the IoT devices to perform as intended and thus cannot be considered a suitable defense against botnets.

Networks with different distributions of *Always-Connected* and *Reboot-Capable* devices did not produce significantly different results. Battery-operated devices were also considered, but these typically have lifespans too long to simulate with our current limitations.

VII. CHALLENGES FACED

The challenges encountered during the project come from two sources: technological limitations, and a lack of available studies on IoT device execution and rebooting times.

At the time of writing, UPPAAL is available only as a 32-bit program on devices running Windows, which were the only device types available to us. This, coupled with UPPAAL's single-threaded simulations, led to a software bottleneck that ultimately limited what was feasible to simulate on our devices. UPPAAL also stores numbers as 16-bit signed integers, which became problematic as we were simulating many small interactions over a long time. It was necessary to include additional variables for the number of integer overflows, which allowed us to model a higher maximum value, but made recording values and specifying parameters more complicated.

The overall timing of the propagation process is determined by device execution times and network latency, which are both difficult to estimate. The default passwords used by Mirai have been mapped to the devices and vendors [6], so we considered using this with knowledge of those vendors' devices to reason about execution times. In the end, we determined that this is likely negligible compared to latency as the propagation process is computationally simple. Latency can vary wildly for different networks, so we selected a conservative time of 100 ms for most simulations.

Reboot frequency and the duration of a device's reboot process were also difficult to determine. We explored works on rebooting times, user behaviors, and manufacturer recommendations for reboot frequency, but were mostly unsuccessful. A few IoT device manuals [15] suggest that their reboot process takes about 60 seconds, so we decided to use this time in our model. As we were unable to find any concrete information on the recommended reboot frequency, we adjusted our research focus to see how varying this affects the botnet size.

We should note that UPPAAL is a research tool and is under active development, so many of the technical challenges may be solved in future versions of the tool. Access to faster computation resources and alternative operating systems would allow for a wider range of scenarios to be tested on our model. The model is designed to allow timings to be altered easily, so if papers are published in the future on IoT rebooting behaviors, the model can be updated accordingly to obtain potentially more accurate results.

VIII. Related Work

The Mirai botnet has been studied in great detail in [4], [5], [6], forming the basis of our developed model. [4] and [6] describe the roles of bots and each component of the botnet infrastructure, while [5] gives an in-depth description of the specific activities each infected bot performs in propagating the malware. The information gained from these three studies allowed us to accurately depict each component of the botnet infrastructure and their interactions to observe and examine the botnet behavior.

The general infection process of botnets has been modeled using various other modeling formalisms. [16] and [11] use agent-oriented Petri nets to model the Mirai and Hajime botnets, exploring the possibility of using rebooting and Hajime as an innocuous botnet to reduce the Mirai infection rate. Both botnets are, however, modeled only as black box entities without modeling the underlying infrastructure, and the scalability of such an approach is limited as the authors fail to extend the network beyond 25 nodes. Epidemiological approaches, such as [17] and [18], effectively model time, but are limited in the amount of detail that can be modeled. Other attempts involve the use of game theory, machine learning, and economic models [19]. Our approach aims to combine the benefits, in particular, of Petri Nets and Epidemiological models, by providing considerations for time and concurrency while also presenting an extensible model with a fine-grained level of adjustable detail.

Other approaches to study botnets involve collecting data from real-world network traffic and devices, including the use of honeypots [6], [9], DNS traffic logs [6], [20], tracing DDoS attacks back to their source [6], and scanning for devices with bot-like behavior [6], [20], [21], [22]. These methods have the benefit of generating real-world data, which can lead to potentially more accurate results. However, the data collection process requires time and violates ethical considerations, making those approaches out of scope for this project.

IX. Conclusion

In this project, we developed a timed automata model of the Mirai botnet in UPPAAL and observed the infection rate and interactions between individual botnet entities. Additionally, we considered clusters of different device types and simulated large networks of such clusters to see the influence of different device distributions. Finally, we examined the impact of rebooting on both stealthy and active botnets to determine the frequency of rebooting capable of curbing the botnet growth.

Our analysis shows that device rebooting, by itself, is ineffective at reducing the spread of botnets unless performed at a level that would degrade the performance of a device or the availability of the IoT network. The level of botnet activity, as well as decreased network speed, are shown to be notable factors that affect the maintainability of a large botnet over an extended period. While the breadth of our analysis was limited by the available hardware, the existing model is highly scalable and allows for easy adjustment of parameters to simulate myriads of scenarios.

In future work, we aim to explore the effectiveness and viability of other suggested solutions in a network consisting of over ten thousand devices. In particular, we wish to see the impact of patching as well as how the inclusion of devices with limited battery lifespan affect our analysis.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security: A survey," *Journal of Network and Computer Applications*, vol. 88, pp. 10–28, 2017.

[3] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[4] 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.

[5] G. Kambourakis, C. Kolias, and A. Stavrou, "The mirai botnet and the iot zombie armies," in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pp. 267–272, IEEE, 2017.

[6] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*, (Vancouver, BC), pp. 1093–1110, USENIX Association, Aug. 2017.

[7] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[8] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on uppaal," in *Formal methods for the design of real-time systems*, pp. 200–236, Springer, 2004.

[9] S. S. Silva, R. M. Silva, R. C. Pinto, and R. M. Salles, "Botnets: A survey," *Computer Networks*, vol. 57, no. 2, pp. 378–403, 2013.

[10] Cloudflare, "Famous ddos attacks — the largest ddos attacks of all time."

[11] H. Tanaka, S. Yamaguchi, and M. Mikami, "Quantitative evaluation of hajime with secondary infectivity in response to mirai's infection situation," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*, pp. 961–964, IEEE, 2019.

[12] M. Shanmughapriya, G. Sumathi, and K. Aarthi, "Bot net of things–a survey," *International Journal of Engineering and Computer Science*, vol. 7, no. 05, pp. 23926–23930, 2018.

[13] C. Bormann, M. Ersue, and A. Keranen, "Terminology for constrained-node networks," 2014.

[14] W. Stallings and L. Brown, *Computer security: principles and practice*. Pearson, 3 ed., 2015.

[15] Comtrend, "Wireless-n broadband router user's manual," Jun 2012.

[16] H. Tanaka and S. Yamaguchi, "On modeling and simulation of the behavior of iot malwares mirai and hajime," in *2017 IEEE International Symposium on Consumer Electronics (ISCE)*, pp. 56–60, IEEE, 2017.

[17] J. O. Kephart and S. R. White, "Measuring and modeling computer virus prevalence," in *Proceedings 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pp. 2–15, IEEE, 1993.

[18] J. A. Jerkins and J. Stupiansky, "Mitigating iot insecurity with inoculation epidemics," in *Proceedings of the ACMSE 2018 Conference*, pp. 1–6, 2018.

[19] P. Wainwright and H. Kettani, "An analysis of botnet models," in *Proceedings of the 2019 3rd International Conference on Compute and Data Analysis*, pp. 116–121, 2019.

[20] M. Feily, A. Shahrestani, and S. Ramadass, "A survey of botnet and botnet detection," in *2009 Third International Conference on Emerging Security Information, Systems and Technologies*, pp. 268–273, IEEE, 2009.

[21] S. Herwig, K. Harvey, G. Hughey, R. Roberts, and D. Levin, "Measurement and analysis of hajime, a peer-to-peer iot botnet.," in *NDSS*, 2019.

[22] D. Acarali, M. Rajarajan, N. Komninos, and I. Herwono, "Survey of approaches and features for the identification of http-based botnet traffic," *Journal of Network and Computer Applications*, vol. 76, pp. 1–15, 2016.