LIGHTWEIGHT DDOS ATTACK DETECTION USING BAYESIAN SPACE-TIME CORRELATION

Gabriel Guimarães Braga dos Santos Mendonça

Tese de Doutorado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientadores: Edmundo Albuquerque de Souza e Silva
Rosa Maria Meri Leão

Rio de Janeiro
Setembro de 2025

LIGHTWEIGHT DDOS ATTACK DETECTION USING BAYESIAN SPACE-TIME CORRELATION

Gabriel Guimarães Braga dos Santos Mendonça

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientadores: Edmundo Albuquerque de Souza e Silva
Rosa Maria Meri Leão

Aprovada por: Profa. Ana Paula Couto da Silva
Prof. Donald F. Towsley
Prof. José Ferreira de Rezende
Prof. Rodrigo de Souza Couto

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2025

Guimarães Braga dos Santos Mendonça, Gabriel

Lightweight DDoS attack detection using Bayesian space-time correlation/Gabriel Guimarães Braga dos Santos Mendonça. – Rio de Janeiro: UFRJ/COPPE, 2025.

XI, 70 p.: il.; 29, 7cm.

Orientadores: Edmundo Albuquerque de Souza e Silva
Rosa Maria Meri Leão

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 64 – 70.

1. DDoS. 2. IoT. 3. Bayesian inference. 4. Machine Learning. I. Albuquerque de Souza e Silva, Edmundo *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

E tudo quanto fizerdes, fazei-o de
todo o coração, como ao Senhor,
e não aos homens.
Colossenses 3:23

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

DETECÇÃO EFICIENTE DE ATAQUES DDOS USANDO CORRELAÇÃO ESPAÇO-TEMPORAL BAYESIANA

Gabriel Guimarães Braga dos Santos Mendonça

Setembro/2025

Orientadores: Edmundo Albuquerque de Souza e Silva
Rosa Maria Meri Leão

Programa: Engenharia de Sistemas e Computação

Ataques DDoS continuam sendo uma das principais fontes de problemas na Internet e seguem causando perdas financeiras significativas para organizações de todos os portes. Para mitigar seu impacto, a detecção deve, preferencialmente, ocorrer próxima à origem do ataque (por exemplo, em roteadores residenciais ou servidores de borda). No entanto, depender de inspeção de pacotes pode acarretar sérios problemas de privacidade e escalabilidade. Propomos um sistema leve para detecção de ataques DDoS que utiliza apenas contadores de *bytes* e pacotes de roteadores residenciais convencionais. Para detectar ataques com essa quantidade limitada de informações, nossa principal contribuição consiste em definir duas camadas de detecção: (1) um classificador de aprendizado de máquina treinado com dados de usuários residenciais e de *malwares* reais; (2) e um modelo hierárquico Bayesiano que explora a natureza sincronizada dos ataques DDoS ao correlacionar alarmes de múltiplas residências para validar a abordagem em ambientes reais. Coletamos dados de ataques DDoS gerando tráfego real de ataque a partir das casas de um grupo selecionado de voluntários utilizando código-fonte de *malwares* reais. Nesse experimento, conduzido nas residências dos voluntários ao longo de um mês, nosso sistema detectou 99,1% de todos os ataques DDoS lançados, sem alarmes falsos.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

LIGHTWEIGHT DDOS ATTACK DETECTION USING BAYESIAN SPACE-TIME CORRELATION

Gabriel Guimarães Braga dos Santos Mendonça

September/2025

Advisors: Edmundo Albuquerque de Souza e Silva
Rosa Maria Meri Leão

Department: Systems Engineering and Computer Science

DDoS attacks are still one of the primary sources of problems on the Internet and continue to cause significant financial losses for organizations. To mitigate their impact, detection should preferably occur close to the attack origin, e.g., at home routers or edge servers. However, relying on packet inspection may bring serious privacy and scalability issues. We propose a lightweight system for DDoS detection that solely employs byte and packet counts from off-the-shelf home routers. To detect attacks with such a limited amount of information, our key insight consists in defining two detection layers: (1) a ML classifier trained with data from real home user and malware; (2) and a Bayesian hierarchical model that exploits the synchronized nature of DDoS attacks by correlating alarms from multiple homes to check the approach in the wild. We collect data on DDoS attacks by generating real attack traffic from the homes of a selected group of volunteers, utilizing authentic malware source code. In that experiment, conducted using the residences of volunteers and over one month, our system detected 99.1% of all DDoS attacks launched, with no false alarms.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

DDoS attacks remain a major problem for the internet today. Despite numerous efforts in recent years to mitigate their effects, they remain a major concern for both businesses and governments. During the COVID-19 pandemic, multiple providers of DDoS mitigation services reported a sharp increase in DDoS attacks, probably motivated by the increased dependency on remote connectivity [2]. DDoS attacks also play an important role in the context of cyberwarfare, as recently observed in the Russian invasion of Ukraine [3, 4]. Furthermore, as reported in [5] (2024), DDoS attacks had a 20% year-over-year increase.

Mirai is a malware that builds large IoT botnets, reaching a population of approximately 200,000-300,000 devices. Mirai emerged later and shared techniques similar to those of the earlier Bashlite malware, particularly exploiting weak credentials of IoT devices. However, it introduced more advanced features and attack vectors [6]. In 2016, it was responsible for a record-breaking distributed denial-of-service (DDoS) attack that peaked at around 623 Gbps, according to a report by Akamai [7]. Recent reports mention that Mirai-based botnets continue to thrive [8–12]. Almost three years after Mirai's source code went public, security companies still reported activity from Mirai command-and-control (CnC) servers [13]. Furthermore, new Mirai variants continue to emerge from time to time targeting more devices and exploiting new vulnerabilities, while retaining attack vector source code (e.g, UDP flood, TCP SYN flood) mostly unchanged, which includes notable examples such as Mirai_ptea [14], Mukashi [15] and Satori [16]. Mirai variants are capable of infecting a wide range of IoT devices, such as Android devices [17], wireless presentation systems [18], network-attached storage (NAS) devices [15], routers and cameras [19]. Based on data from [20], the number of IoT devices is predicted to increase to 16.7 billion by the end of 2023, and in subsequent years, a growth rate

of over 15% is projected. Therefore, it is increasingly important to develop robust mechanisms to identify DDoS attacks originating from IoT devices.

We provide in Figure 1.1 a simplified state diagram of a *bot* infected by the Mirai malware. Each infected device is responsible for finding other vulnerable devices, allowing the malware to spread forming very large botnets. When the CnC server, controlled by the so-called *botmaster*, sends an attack command, the bots start to send attack traffic to the IP address of one or more victims following the received instructions (e.g, protocol, duration, packet size) [6].

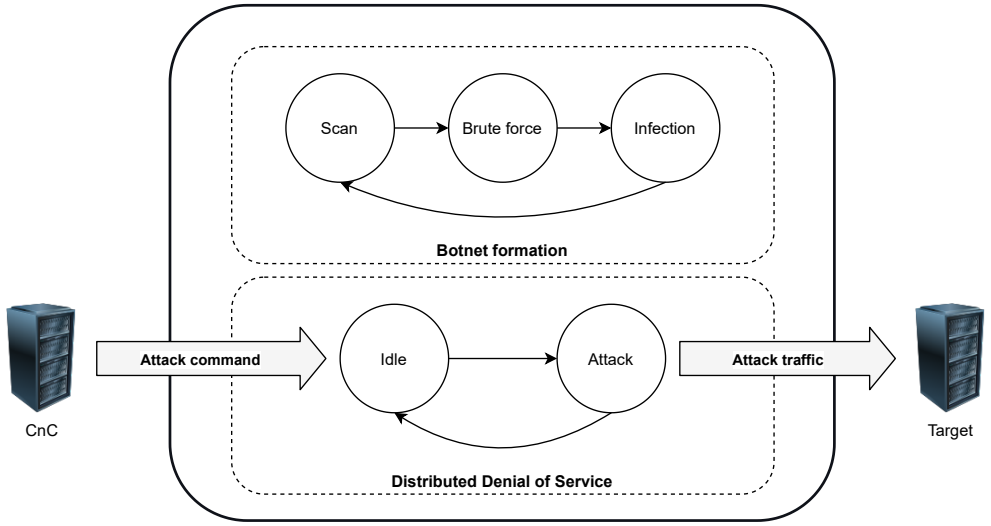


Figure 1.1: Simplified state diagram of a device infected by the Mirai malware.

1.2 Challenges

Despite all of the advances in DDoS detection (and mitigation), DDoS attacks remain a threat, especially for ISPs. Machine learning methods offer several advantages when applied to the detection of DDoS attacks: improved accuracy, adaptability, scalability and real-time detection. However, it's important to note that the effectiveness of these machine learning models is significantly influenced by the quality of the data utilized for their training and evaluation.

Labeled DDoS datasets are often difficult to obtain [21, 22]. Many available options have significant limitations: some are outdated, such as the CAIDA 2007 dataset [23] and the NSL-KDD dataset [24], while others rely solely on synthetic data, including UNSW-NB [25], CIC-DDoS2019 [26], and Bot-IoT [27]. Furthermore, several datasets use testbeds to emulate traffic from selected IoT devices, such as the N-BaIoT dataset [28], McDermott et al.'s dataset [29], IoT-23 dataset [30], LATAM-DDoS-IoT Dataset [22], ACI IoT Network Traffic Dataset 2023 [31], and the recent CIC-BCCC-NRC TabularIoTAttack-2024 dataset [32]. These datasets typ-

ically generate data in controlled environments to emulate smart home networks. In contrast, our dataset captures real-world traffic data from over four thousand residences. Although labels are difficult to obtain (since it is difficult to know when DDoS attacks occur), real traffic data from home users is also very scarce, partly due to privacy concerns.

Another challenge in the development of efficient DDoS attack detection systems concerns the location in the network where detection occurs. To mitigate the impact of botnet-based DDoS attacks launched by IoT devices, detection should preferably occur close to the origin of the attack. With traditional solutions that operate at the network core [33–38], detection usually occurs far from the attack source, making it difficult to minimize the impact of attacks that come from inside the ISP's network.

Detecting DDoS attacks close to their source, such as at home routers, is beneficial, particularly from a network-wide perspective. Many DDoS attacks originate from compromised devices, like IoT devices, integrated into botnets. Therefore, mitigating threats at the source—specifically at home routers—can prevent malicious traffic from leaving the local network and reaching its target. This approach can significantly reduce the attack's impact on upstream networks and the intended victim. Thus, the detection on home routers can be seen as a better alternative than the detection near the victim, as these devices are close to the attack sources and can often be managed remotely by ISPs. However, these devices are typically limited in memory and processing power. In particular, the use of deep packet inspection or information gathered from packet headers does not bode well for such devices. Furthermore, detection must not interfere with the performance of home users or violate their privacy. Although detection of DDoS attacks on home routers has been considered before [28, 29, 39–44], the proposed solutions rely on information extracted from packet traces for attack detection (e.g., source / destination IP addresses, protocols, ports). Moreover, the reported results come from limited IoT testbeds with a handful of devices.

Since most DDoS attacks are synchronized and aggregate a myriad of devices, we expect the traffic from distinct "infected" homes (which contain an IoT device infected by malware) to be correlated. Even devices that respond to different botnets may simultaneously launch DDoS attacks on the same victim [45]. Therefore, the idea of running DDoS detectors on edge servers, with or without the help of home-router-based detectors, has also been explored recently [46–48]. With the exception of [48], most proposed solutions suffer from the same issues: dependence on sensitive home user information extracted from packet headers and evaluation using very limited datasets.

In summary, developing an effective and practical DDoS attack detection system using machine learning poses multiple research challenges. These include the need for

a lightweight defense system that allows rapid detection of attacks, the acquisition of a dataset that represents real user traffic and attack patterns for model training, and the capability to effectively detect existing attack variants. Our proposed solution addresses these challenges and provides effective solutions to them.

Goals In this work, we address the following questions.

1) How feasible is it to detect DDoS attacks through an ultra-lightweight approach without relying on packet header information?

2) How can we effectively leverage the synchronized nature of DDoS attacks to develop more robust attack detection mechanisms?

1.3 Contributions

We propose a two-layer system to detect DDoS attacks. The first layer – the *local detector* – runs at the home user level; the second – the *spatio-temporal correlation layer* – runs at the ISP level. The *local detector* is a lightweight machine learning classifier, designed for off-the-shelf home routers. It utilizes features derived from byte and packet counters rather than depending on features extracted from packet header data. The *spatio-temporal correlation layer* employs an innovative hierarchical Bayesian model that leverages the spatio-temporal correlation among alarms from various home routers. This approach not only enhances the likelihood of accurately detecting attacks but also contributes to shorter detection times. Our key contributions are summarized as follows.

(1) **Detection at home routers with minimal information and preserving user privacy.** We utilized a labeled dataset to train a lightweight Machine Learning model, serving as the local detector running on home routers. We have found that simple statistics derived solely from byte and packet counts – metrics that are natively accessible on embedded Linux platforms – are adequate for achieving a high attack detection rate with a low False Positive Rate. An important contribution of our work is understanding the most relevant features to classify a given sample as an attack through a surrogate model.

Our approach offers an additional benefit whereby user data is directly analyzed on the home routers. This ensures that only the outcomes generated by the locally executed model are transmitted to a central server, guaranteeing user data privacy and confidentiality. Furthermore, it enables the implementation of local mitigation strategies, such as rate limiting and dynamic IP blocking.

(2) **Spatio-temporal correlation.** We consider spatial correlations from distinct homes during the same time interval, exploiting the fact that most DDoS attacks are synchronized. The spatio-temporal correlation among homes is captured using a Bayesian hierarchical model. The model is used to decide whether or not

there is an attack (Bayes factor) and to obtain the posterior distribution of latent variables. The results show that detection can be improved by one or more orders of magnitude when only a small fraction of home routers report an attack. The Bayesian model is the second layer of detection, further improving the true positive and false positive rates.

The characteristics of the network traffic may change over time. Home user traffic patterns may change, as seen in 2020 during the COVID-19 pandemic [48, 49]. Also, DDoS attacks may become more/less aggressive, becoming easier/harder to detect. Such factors may alter the performance of the DDoS detector. Therefore, we show how we can cope with eventual traffic changes by dynamically adjusting our Bayesian model parameters over time using posterior predictive distributions.

(3) **Unique labeled dataset.** We partner with an ISP to carry out a data collection campaign on home gateways. Volunteers from 14 different cities in the state of Rio de Janeiro / Brazil had an instrumented version of a home router installed by the ISP. These routers gather information about network usage. For this work, we only use byte and packet counts collected from the network interfaces of 4,870 home routers over a 20-day period. These byte and packet count traces constitute our baseline. We executed controlled experiments to emulate Mirai and BASHLITE attacks, considering a number of typical attack vectors. Then, we add these attacks to the baseline traces, resulting in the labeled dataset. To enable reproducibility and address the scarcity of public botnet datasets [21], we make our labeled dataset public and available upon request.

(4) **Detection of attack variants**. Our DDoS detection approach differs from previous work by focusing on an accurate representation of the home user's traffic profile, derived from home user measurements, rather than on the specifics of the attack vector. As a result, the proposed framework is capable of detecting *variants of attack vectors* that were not present in the training dataset. Our classifier also shows good performance when we assume the *bot master* employs suboptimal attack parameters to possibly avoid detection and when we consider mislabeled training data to account for uncertainty in the baseline traffic data.

(5) **Detecting real DDoS attacks.** We carried out a controlled experiment using 10 volunteers and a device in each of their homes. A Raspberry running real DDoS attacks, using source code extracted from Mirai and BASHLITE malwares, was placed in each volunteer's home. Periodically, these Raspberry Pi devices launched simultaneous attacks at a server. The proposed two-layer system was able to detect 99.1% of all attacks in the controlled experiment, over a period of 30 days, while not raising false alarms. We are unaware of previous work evaluating the performance of a DDoS detector in a real scenario like this.

1.4 Document structure

The remainder of this document is organized as follows. Chapter 2 presents an overview of the state-of-the-art DDoS attack detection techniques. In Chapter 3 we describe the proposed two-layer system for the detection of DDoS attacks. The datasets are described in Chapter 4. Chapter 5 reports the results and Chapter 6 presents the conclusions.

Chapter 2

Related Work

Extensive research has been conducted on DDoS attack detection mechanisms, as documented in [50] and [51]. However, for the purpose of our literature review, we narrow our focus to works that share a similar emphasis to ours, specifically examining the latter stages of botnet operations when flooding attacks are executed and DDoS mitigation takes place at home routers and edge servers. Table 2.1 summarizes the related work discussed in this chapter.

2.1 Network core

Common solutions for DDoS detection and mitigation are based on the analysis of packet flows, headers and/or payloads at the network core [33–37]. By making use of packet-level information, such solutions are efficient to detect previously identified attack signatures [34–36] or misbehavior of flows with respect to an equilibrium [33] or with respect to statistical properties of header fields [37].

Liaskos et al. [34] propose a traffic engineering strategy to deal with link-flooding attacks. By continuously re-routing traffic, bots are forced to adjust their strategies. Such adjustment, in turn, helps the detection of the bots through temporal correlations in their traffic. Nevat et al. [36] also leverage the temporally correlated nature of traffic generated by bots to detect them.

Marín et al.[35] evaluate the feasibility of detecting attack traffic using *raw*, non-processed packet data. According to the authors, the proposed Deep Learning model captures the underlying statistics of malicious traffic without resorting to "expert handcrafted features". Instead, the model uses the first n bytes of each packet's payload as features, where n is a fixed threshold parameter. We show in this work that, with simple features like the standard deviation of the upstream packets per second rate during a 5-minute window, traditional Machine Learning models can successfully detect attacks close to the attack source without the need of complex DL-based models like LSTMs.

Table 2.1: Comparison of related work.

Ref.	Placement	Input data	Dataset	Real malware	Model
[34]	Network core	Traffic matrix	Simulated network	No	Relational algebra
[35]	Network core	Pkt traces	USTC-TFC2016 [52]	Yes	LSTM; CNN
[36]	Network core	Pkt traces	MAWI [53]	No	Markov Chain
[37]	Network core	Pkt traces	Scorpius [54]	No	ARIMA; Ant colony optim.
[55]	Host	Pkt traces	Simulated IoT network	No	Neural Network
[56]	IoT device	Pkt traces	Testbed with 2 devices	No	Bit-pattern matching
[28]	Home router	Pkt traces	Testbed with 9 devices	Yes	Deep Autoencoders
[29]	Home router	Pkt traces	Testbed with 2 devices	Yes	LSTM
[39]	Home router	Pkt traces	Testbed with 3 devices	No	Neural Network; Random Forest
[40]	Home router	Pkt traces	Testbed with 9 devices[1]	Yes[1]	Decision Tree
[41]	Home router	Pkt traces	Testbed with 9 devices[1]	Yes[1]	SVM; Isolation Forest
[42]	Home router	Pkt traces	Testbed with 6 devices[2]	No	Random Forest; PCA
[43]	Home router	Pkt traces	Testbed with 8 devices	No	Decision Tree
[44]	Home router	Pkt traces	Testbed with 7 devices	No	Random Forest
[46]	Home router; Edge server	Pkt traces	CIC-DDoS2019 [26]	No	LSTM; CNN
[47]	Home router; Edge server	Pkt traces	Emulated IoT net; IoT-23 [30]; Bot-IoT [27]	No	Random Forest; FIM
[48]	Home router; Edge server	Byte/pkt rates	812 residences	Yes	PARAFAC; Random Forest
Our work	Home router; Edge server	Byte/pkt rates	4870 residences; 10 residences with infected Raspberry Pi	Yes	Random Forest; Hierarchical Model

[1] Same dataset as [28].

[2] The system was deployed to 22 home networks, but detection results are restricted to data collected at a smart home sandbox.

Pena et al.[37] propose anomaly detection methods based on unsupervised learning that do not rely on attack signatures or labeled traffic. Such methods have the advantage of generalizing to novel attack vectors, but still require packet level information.

Although solutions at the network core are well-suited to detect different sorts of anomalies, they fall short when dealing with botnet-based DDoS. This is because the detection (if successful) usually occurs far from the attack source, making it hard to minimize the impact of the attack. Additionally, the processing of large traffic volumes leads to scalability issues, which may translate into low detection rates [57]. Nonetheless, these solutions can complement our proposed approach.

2.2 Network hosts

In [55, 56], the authors propose solutions wherein attacks are detected at network hosts. Although such methods are effective to localize and isolate botnet sources, their execution may not be feasible at resource constrained IoT devices. It is well known that IoT devices usually have severe resource constraints, limiting the complexity of detection algorithms. In addition, software/firmware updates for such devices are generally sporadic. Updating IoT devices typically removes their vulnerabilities and exploits, preventing malware infections. However, the patching behavior is very erratic and alternative solutions that do not rely exclusively on the patching of network hosts should be sought [58].

2.3 Home routers

Botnet attacks detected at home routers can be mitigated without compromising the whole network. For instance, DDoS attack traffic can be blocked at the home network and either the home user or the ISP can be notified of the detected attack. Multiple solutions for DDoS attack detection in the home routers have been proposed before [28, 29, 39–44]. The proposed solutions rely on information extracted from packet traces for attack detection (e.g., source / destination IP addresses, protocols, ports). Besides that, reported results usually come from restricted sandboxes with a limited number of IoT devices (e.g, [28, 29, 39–44]).

Doshi et al. [39] and Anthi et al. [43] use supervised Machine Learning models to classify individual network packets as benign or malicious. Results are based on emulated attack data, generated in a limited testbed with a few devices (3 devices in [39] and 8 in [43]). While in [39] the authors aim at detecting DDoS attacks, the architecture proposed in [43] also aims at detecting and identifying other types

of attacks, like man-in-the-middle and spoofing. In our work, instead of relying on emulated data, we use real home user traffic from more than 4,000 residences.

Wan et al. [42] adopt a hybrid approach, combining both supervised and unsupervised ML algorithms to detect known attacks (based on signatures) and unknown attacks (using anomaly detection). Along with the flooding attacks, the proposed system tries to detect other stages of the botnet lifecycle, like the scanning and brute force (infection) phases. Even though the authors argue the system was deployed in 22 home networks, the performance evaluation presented in the paper is limited to a smart home sandbox consisting of 6 IoT devices while ours uses data collected from 4870 residences.

McDermott [29] and Meidan et al. [28] use Deep Learning (DL) algorithms for DDoS attack detection. In [29], the authors follow a supervised approach, using Long Short Term Memory (LSTM) Neural Network models. The results from their experimental setup (containing two IP cameras) showed that the most relevant information for attack detection was extracted from packet payloads. The analysis of packet payloads introduces serious scalability issues for off-the-shelf home routers. Furthermore, unencrypted packets may contain sensitive information from the home users, so privacy-preserving alternatives should be sought. The authors of [28] follow an unsupervised approach, using deep autoencoder models to detect anomalous network packets from IoT devices. The proposed system relies on training an autoencoder model per device model in the network. It is clearly impractical to train a Deep Learning model to understand the baseline operation of each available IoT device model. While the dependence on packet flow data already limits the applicability of the proposed model, the use of DL algorithms makes it impractical for execution on home routers. Subsequent work [40, 41, 44] provided evidence that simple ML models can be used to efficiently detect attacks in place of more complex Deep Learning models. In particular, the authors of [40, 41] use the same dataset used in [28]. Salman et al. [44] evaluate both ML and DL models, with standard Machine Learning models (Random Forests) presenting better results in most scenarios. Like in [28, 29, 39–44], the proposed framework depends on packet flow data, and is evaluated on a small IoT sandbox (7 devices).

Note that whereas most previous solutions classify individual packets as either benign or malicious [28, 29, 39–41, 43], our goal in this work is to detect the presence of attacks within a given time window. This seemingly subtle difference leads to important consequences, as discussed in Chapter 5. In contrast to approaches that rely heavily on packet header information, our method questions its necessity, while also leveraging authentic home user traffic.

2.4 Edge servers

Other works propose the use of edge servers for DDoS attack detection, either with [47, 48] or without [46] the cooperation of detectors running on home routers. The rationale is that servers located at the edge of the network, with more computational power than home routers, can provide a good balance between good detection rates and scalability while being relatively close to the source of the attacks.

Jia et al. [46] propose FlowGuard, an edge defense mechanism that combines data from multiple home networks at edge servers, uses Deep Learning algorithms (LSTM and CNN neural networks) to detect malicious network flows, and employ flow filtration rules for attack mitigation. The authors make use of the CICDDoS2019 dataset [26], built with synthetic home user traffic and DDoS attack data generated by traffic simulators. In our work, we use real DDoS attack traffic data generated with Mirai and BASHLITE source code and real home user data collected from more than 4,000 homes.

Sudheera et al.[47] and Streit et al.[48] adopt an approach similar to the one proposed in this work, correlating the outputs of attack detectors across multiple homes. In [47], the authors introduce Adept, a framework that combines a lightweight anomaly detection algorithm on home routers with a frequent itemset mining (FIM) algorithm at the edge server to identify attack patterns correlated across time and space. However, Sudheera et al. do not use real malware source code to collect attack traffic signatures. Moreover, like the vast majority of proposed solutions, the Adept framework relies on sensitive packet header data for its operation. In [48], another study from our research group, Streit et al. propose a method for anomaly detection based on tensor decomposition that does not require packet header inspection. While their evaluation uses data similar to our *labeled datasets*, the authors do not provide results from experiments involving real DDoS attacks.

2.5 Our work

In this work, we propose a two layer DDoS detection system capable of detecting attacks with a minimal amount of information from home users. Instead of relying on packet header or payload data, our lightweight ML classifier uses features extracted from byte and packet counters of off-the-shelf home routers. Our novel hierarchical Bayesian model leverages the spatio-temporal correlation between alarms from different home routers to further improve the probability of detecting attacks, while reducing detection times.

We see in Table 2.1 that the usage of two DDoS detection layers – one at home

gateways and other at edge servers – is not a novel idea. However, from our knowledge, the present work is the first to combine:

1. Lightweight features derived exclusively from byte and packet counters
2. Real home user traffic data
3. Traffic signatures from real malware
4. Evaluation with real DDoS attacks.

Chapter 3

System Overview

The main question addressed in this work is summarized as follows: *how can we leverage home user traffic patterns to detect attacks within few minutes of their onset?* To address this question, we develop models that make use of measurements taken from home routers that we use to develop algorithms that distinguish baseline home user activity and DDoS attacks.

The key requirements of the proposed methodology are: (a) **simplicity:** the methodology should facilitate straightforward extension, generalization, and scalability; (b) **locality:** it should enable detection of attacks close to the source, i.e., at the home gateway or the ISP; (c) **timeliness:** the methodology should detect attacks swiftly, typically within a few minutes of their initiation; (d) **minimalism:** it should rely on a minimal amount of information from home routers to prevent interference with their performance and ensure the preservation of user privacy.

To meet the above requirements, our methodology is based solely on a time series of *upstream and downstream traffic* (bytes and packets) transmitted through home routers, differing from previous work that relies on detailed information (e.g., packet traces) for similar tasks. The home user traffic data needed by our system can be gathered by ISPs using simple measurement infrastructures like in [59, 60].

Our system comprises two layers: a local layer at the home user level and a spatio-temporal correlation layer at the ISP level. The set of homes monitored by the system is denoted by $\mathcal{H}$. We assume that a subset of homes contains an infected device. These homes are denoted as *infected homes* and constitute the subset $\mathcal{B} \subset \mathcal{H}$. The infected devices (*bots*) are part of a *botnet*, and act as intermittent attackers responding to a Command and Control (CnC) server in a synchronized way. Figure 3.1 shows the proposed architecture. In the first layer, detection is based on a lightweight machine learning (ML) classifier running on each home router. With byte/packet counts samples collected every minute, the classifier uses simple statistics (e.g., standard deviation) calculated over a time window of size τ. We adopt in this work a *sliding window* of $\tau = 5$ minutes. The 5-minute window slides at minute

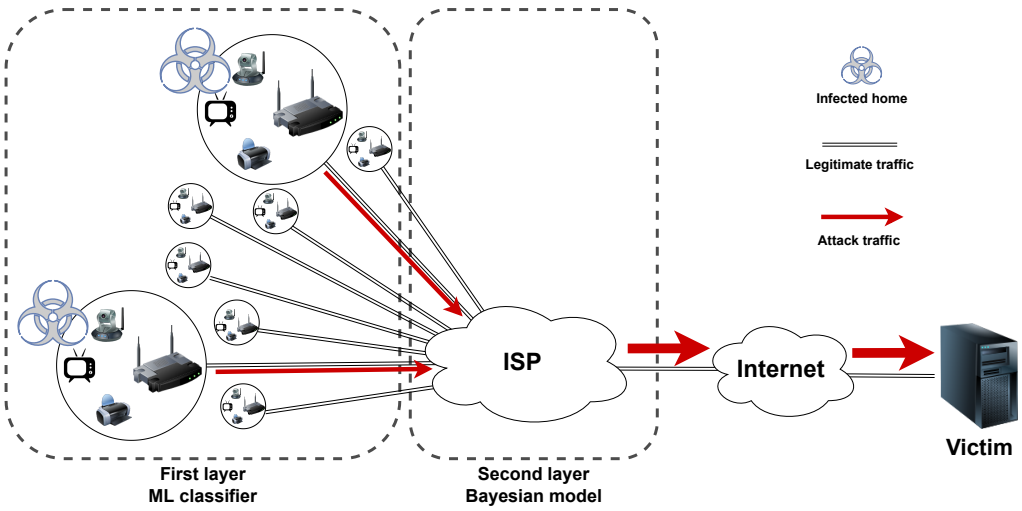


Figure 3.1: Architecture of the proposed two-layer DDoS detection system.

intervals, and statistics are calculated using only samples from the current window. Consequently, we obtain a new set of statistics at one-minute intervals. As we show in Section 5.1, a sliding window of 5 minutes contains enough information to detect attacks with high accuracy while keeping the detection time low.

Since DDoS attacks are synchronized in nature, the second system layer explores correlation among results from distinct homes during a given period of time. The concept is straightforward: if a single home router detects an attack within a specific time window, it might be a false alarm. However, if many routers simultaneously report an attack, the likelihood of it being a false alarm decreases significantly.

Intuitively, as the number of individual alarms increases, it becomes more likely that a synchronized DDoS attack is occurring. Conversely, a smaller number of positive results supports the hypothesis that no attack is in progress. In light of this, we develop a Bayesian hierarchical model to capture the expected behavior of a botnet-based DDoS attack. By using the Bayes factor, we assess the strength of evidence for each hypothesis, thereby increasing the detection probability and reducing the false alarm rate, as demonstrated in Section 5.2. Furthermore, we infer the posterior distribution of latent variables (such as the unknown number of bots) through analytical expressions. As demonstrated in Section 3.2.4, the model is also flexible and can learn from past events as labeled data becomes available.

The proposed two-layer setup provides more flexibility when compared to the detection exclusively at home routers or at edge server. Attacks can be mitigated at the home router or at the ISP depending on performance requirements (e.g., false alarm rate, attack detection probability, time to detect). In either way, we show that the vast majority of the attacks can be detected, making it harder for DDoS attacks to be successful. To our knowledge, no previous work considered such approach besides our model presented in [61] and later applied by Streit at al. in [48].

This chapter is structured as follows. First, we present in Section 3.1 the proposed local detection layer. In Section 3.2, we provide details on our spatio-temporal correlation layer. Finally, Section 3.3 discuss some implementation details.

3.1 Local detection layer

Our first layer aims to continuously detect, for individual home users, whether a DDoS attack occurred during a given sliding window. In particular, our primary goal is to evaluate the feasibility of a simple and reliable classifier using solely byte and packet counts. For this purpose, the classifiers use features derived from upload and download traffic samples collected from home routers during a sliding window of size τ. The algorithm runs continuously, processing each window in real-time as it becomes available. In this local layer, the windows are treated independently, without accounting for temporal correlations among windows of the same home router or spatial correlations across different home routers.

The local classifier generates a classification result every minute based on the data available during the interval defined by the current window. In other words, for window w_i at time t_i and using the data available in the interval $[t_i, t_{i-\tau}]$, the local classifier determines if the interval covered by w_i ($[t_i, t_{i-\tau}]$) contains attack traffic. If so, it sends an alarm at t_i. After testing a few window sizes, we opted in this work for a sliding window of size $\tau = 5$ minutes. We note that the hyperparameter τ can be easily changed, especially if a substantial shift in the average DDoS attack length is observed.

Figure 3.2 illustrates the behavior of the classifier for a residence. The figure shows three attacks starting at time t_i, t_j, and t_k, with durations of $\gamma = 2$, $\gamma = 6$ and $\gamma = 1$ time slots, respectively. It displays only the sliding windows that include at least a single slot with an attack. Window w_i contains data from the corresponding local residence collected from time slots $[t_i, t_{i-\tau}]$, where $\tau = 5$. In the figure, window w_i includes only a single time slot with traffic from the attack starting at t_i. We illustrate a scenario where the classifier at time t_i fails to detect any attack within this window, represented in the figure by an empty circle.

After t_i, the first window that detects an attack is w_{t_i+2} and sends an "alarm" (at t_{t_i+2}) to the control center (the second layer). A red circle represents the alarm. As shown in the figure, during the first attack lasting two slots, the local classifier issues only two alarms for all $\tau + \gamma - 1 = 6$ windows that encompass at least one slot with attack traffic. The behavior in subsequent attacks is similar, and the local classifier sends four and zero warnings for the last two attacks.

We collect four passive network metrics at every slot: upstream bps rate, upstream pps rate, the ratio between upstream and downstream bps rates, and the

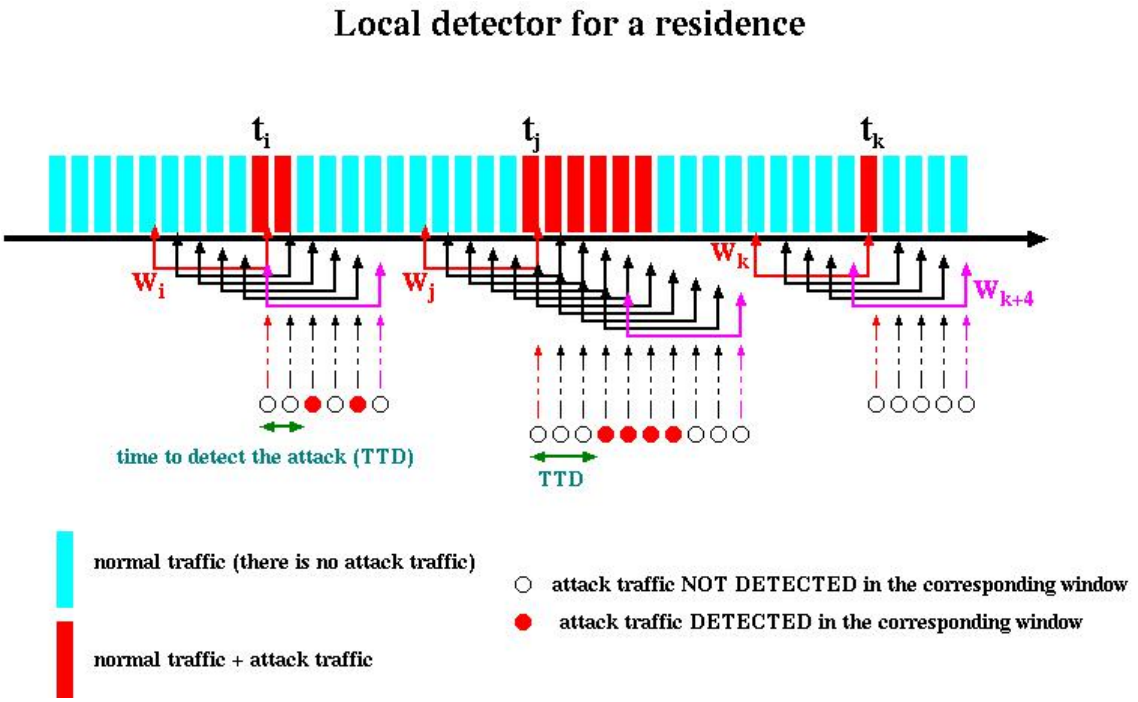


Figure 3.2: Illustration of the behavior of the classifier for a residence (local classifier).

ratio between upstream and downstream pps rates. The "raw" downstream rates only added noise to the dataset, worsening our classifier's results. Therefore, download traffic is only considered in the up / down rates. This is not surprising, since (as later discussed in Section 4.2) the distribution of downstream traffic rates during attacks is almost indistinguishable from the distribution of legitimate traffic. Also, we avoid relying on active network metrics (e.g., latency, packet loss, throughput), since they are less generalizable.

For each window, we compute three descriptors for each network metric: standard deviation, maximum, and the difference between the maximum and minimum (other statistics, such as mean, median, and minimum, were also tested but showed negligible improvement). This results in a total of 12 features for each window (see Table 3.1). During a DDoS attack, there is a significant increase in upstream traffic. Therefore, we define a set of features based on upstream traffic, as well as the relationship between upstream and downstream traffic flows. These features are used as input to the classifier.

We train the models using a labeled DDoS dataset that combines real baseline home user traffic with attack traffic generated using actual malware source code, as described in Section 4.1. The trained model is then evaluated under various scenarios, including the presence of mislabeled samples and the assumption of a smart attacker. Additionally, we report results using a dataset collected from a controlled experiment in which DDoS attacks were generated by devices connected

Table 3.1: List of features used by the ML classifier (first layer).

Bits per second features	Packets per second features
max upstream bps rate	max upstream pps rate
max $\frac{\text{upstream bps rate}}{\text{downstream bps rate}}$	max $\frac{\text{upstream pps rate}}{\text{downstream pps rate}}$
max-min upstream bps rate	max-min upstream pps rate
max-min $\frac{\text{upstream bps rate}}{\text{downstream bps rate}}$	max-min $\frac{\text{upstream pps rate}}{\text{downstream pps rate}}$
std dev upstream bps rate	std dev upstream pps rate
std dev $\frac{\text{upstream bps rate}}{\text{downstream bps rate}}$	std dev $\frac{\text{upstream pps rate}}{\text{downstream pps rate}}$

to home routers in residences (Section 5.3).

We considered five different candidate ML models: Logistic Regression, Decision Trees, Random Forests, Gaussian Naive Bayes and Multilayer Perceptron. The best model was selected through 5-fold cross-validation and then evaluated using test datasets. Hyper-parameters were set to the default values given by `scikit-learn`[1].

3.2 Spatio-temporal correlation layer

The second detection layer employs a Bayesian hierarchical model to correlate the results from distinct homes over the same time period. Let $N = |\mathcal{H}|$ represent the number of monitored homes, each containing a home router running the ML classifier (local detection layer). Each residential router classifier analyzes the upstream/downstream traffic every minute and reports the results to a central server (Figure 3.1). The classifier's output can either be *negative* (indicating regular home user traffic) or *positive* (indicating malicious traffic, i.e., a DDoS attack). Each home router may report a *false positive* (in the absence of attack traffic) with probability θ_F. The probability of reporting a positive result when there is an attack is θ_D (i.e., the *true positive* rate, also known as recall or sensitivity). We assume that $\theta_D > 0.5$ and $\theta_F < 0.5$, as any practical classifier should perform better than random guessing.

Figure 3.3 illustrates the behavior of the second-layer detector. The second-layer receives information from all residences (e.g, home-routers) in which the local classifier is installed. The figure illustrates the output of three residences (L_m, L_n and L_o). The figure shows that different local classifiers may detect the attack at different time instants. It also illustrates that a false alarm occurred (local detector L_o). Using the information received from the local classifiers, the second layer classifier decides, at every slot, if an attack occurs or not.

Many homes may not contain an infected device (bot). We assume that only a fraction θ_b of the N homes are infected and therefore participate in attacks. We denote by x_i the total number of home routers that report positive at time t_i, while

[1] `http://scikit-learn.org`

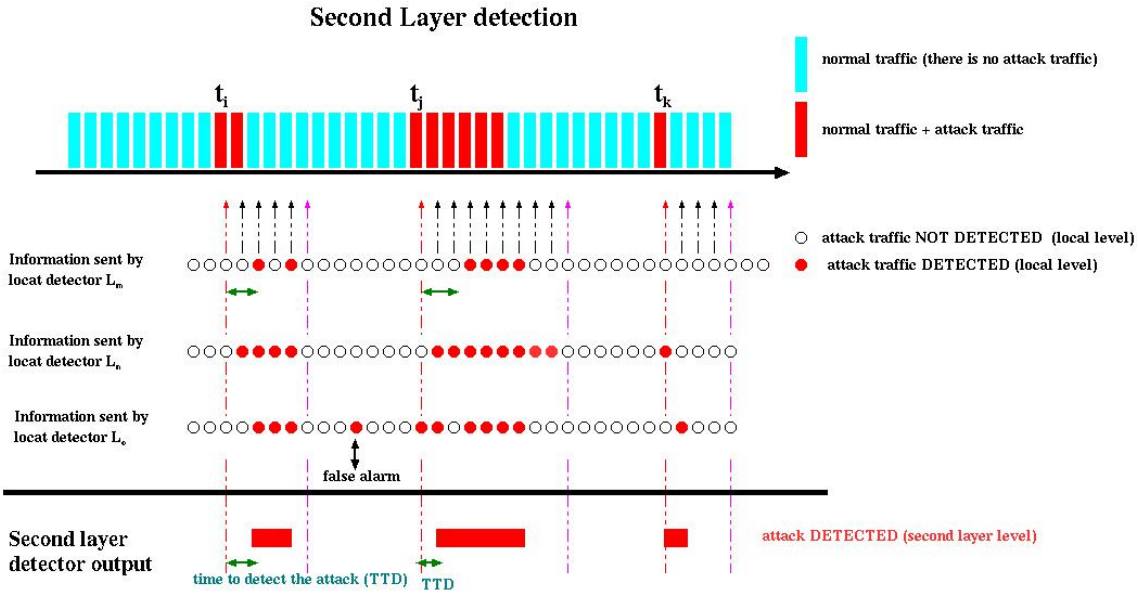


Figure 3.3: Illustration of the behavior of the second-layer detector. It receives one bit of information from local classifiers.

the number of negative results is simply $N - x_i$. Our problem then is to decide between two hypotheses based on the observed value x_i:

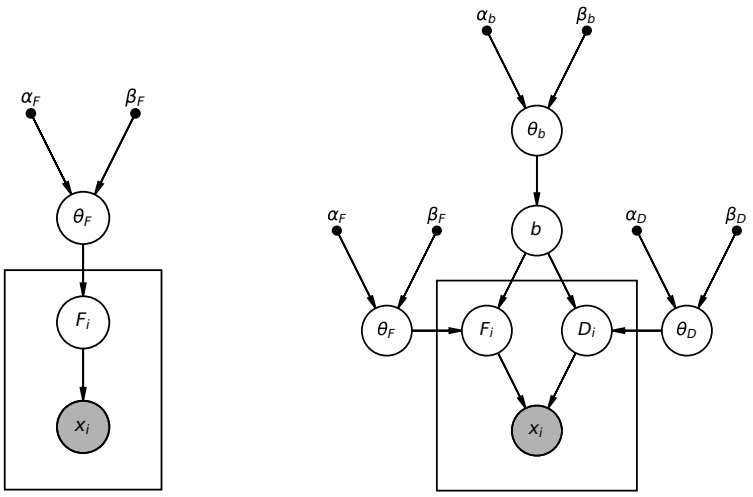
- H_0 (*null hypothesis*) — There was no DDoS attack in the last five-minute window; any positive results reported by home routers are *false positives*.
- H_1 (*alternative hypothesis*) — There was a DDoS attack in the last five-minute window; positive indicators are due to either *true positives* (from homes participating in the attack) or *false positives* (from homes not participating in the attack).

Figure 3.4a illustrates the probabilistic graphical model (PGM), representation of the system under the null hypothesis (H_0), where no DDoS attack is occurring, and Figure 3.4b illustrates the PGM under the alternative hypothesis (H_1), where a DDoS attack is occurring. The latent variables are represented by white nodes and the observed variable x_i is represented by a gray node. The dots indicate the model hyperparameters. The proposed models, now called the *beta-binomial attack models*, follow a beta-binomial structure, capturing the relationships between observed and latent variables defined in Table 3.2.

In the model of Figure 3.4a, x_i represents the number of home routers that report positive at time t_i, assumed to be false positives, and θ_F denotes the probability of a false positive at a home router. In the model of Figure 3.4b, x_i represents the number of home routers reporting positive at time t_i, which includes both true positives (from routers involved in the attack) and false positives (from routers not involved in the attack). θ_D denotes the probability of a true positive (i.e., a home

Table 3.2: Table of Notation.

		Observed variables
N	$\triangleq$	Number of homes / home routers.
x_i	$\triangleq$	Number of home routers that report positive at time t_i.
		Latent variables
θ_b	$\triangleq$	Probability of a (infected) home participating in the attack.
b	$\triangleq$	Number of (infected) homes participating in the attack.
θ_F	$\triangleq$	Probability of a false positive at a home router.
F_i	$\triangleq$	Number of false positives at time t_i.
θ_D	$\triangleq$	Probability of detection (true positive) at a home router.
D_i	$\triangleq$	Number of detections (true positives) at time t_i.
		Hyperparameters
α_b, β_b	$\triangleq$	Parameters of the prior distribution over θ_b.
α_F, β_F	$\triangleq$	Parameters of the prior distribution over θ_F.
α_D, β_D	$\triangleq$	Parameters of the prior distribution over θ_D.
α'_b, β'_b	$\triangleq$	Parameters of the updated prior distribution over θ_b.
α'_F, β'_F	$\triangleq$	Parameters of the updated prior distribution over θ_F.
α'_D, β'_D	$\triangleq$	Parameters of the updated prior distribution over θ_D.
		Hypothesis
H_0	$\triangleq$	No DDoS attack (null hypothesis).
H_1	$\triangleq$	DDoS attack (alternative hypothesis).



(a) Null hypothesis H_0 (no attack). (b) Alternative hypothesis H_1 (attack).

Figure 3.4: Probabilistic Graphical Model representation under each hypothesis (*beta-binomial attack model*).

participating in the attack), θ_F represents the probability of a false positive, and θ_b the probability of an infected home participating in the attack. (More details about the models are in Sections 3.2.1 and 3.2.2.)

One way of deciding between two hypotheses is to compare how well each one explains the observed data. After observing x_i positive results from N home routers at time t_i, our intuition tells us to decide in favor of the hypothesis under which x_i is more likely. We expect x_i to be greater under H_1 (attack) when compared to H_0 (no attack). However, we may observe a large x_i by chance even when there is no attack. Therefore, given the set of hyperparameters $\boldsymbol{\phi} = \{\alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D, \beta_D\}$, we are going to decide based on the ratio between the model evidences, defined as

$$\mathrm{BF}_{10}(x_i, \boldsymbol{\phi}) \triangleq \frac{p(x_i \mid \boldsymbol{\phi}, H_1)}{p(x_i \mid \boldsymbol{\phi}, H_0)}. \tag{3.1}$$

This ratio, known as the *Bayes Factor*, measures the relative credibility of the models (hypotheses) H_0 and H_1. By convention, we assume there is substantial evidence for hypothesis H_1 if $\mathrm{BF}_{10}(x_i, \boldsymbol{\phi}) > 3$ and for hypothesis H_0 if $\mathrm{BF}_{10}(x_i, \boldsymbol{\phi}) < 1/3$ [1]. We summarize in Table 3.3 different evidence categories based on the value of the Bayes factor. We note that, unlike in classical hypothesis tests, we can have evidence in favor of the null hypothesis, in favor of the alternative hypothesis or negligible evidence that favors no hypothesis (values close to 1).

While frequentist models usually rely on point estimates for distribution parameters (e.g., using Maximum Likelihood Estimation or Expectation Maximization), in the Bayesian framework we assume prior distributions for the model parameters and marginalize the likelihood by integrating over the parameter space. If hypoth-

Table 3.3: Evidence categories for Bayes factors (adapted from [1])

Bayes factor BF_{10}	**Interpretation**
> 100	Decisive evidence for H_1
$30 - 100$	Very strong evidence for H_1
$10 - 30$	Strong evidence for H_1
$3 - 10$	Substantial evidence for H_1
$1 - 3$	Anecdotal evidence for H_1
1	No evidence
$1/3 - 1$	Anecdotal evidence for H_0
$1/10 - 1/3$	Substantial evidence for H_0
$1/30 - 1/10$	Strong evidence for H_0
$1/100 - 1/30$	Very strong evidence for H_0
$< 1/100$	Decisive evidence for H_0

esis H_j, $j = 0, 1$, have a likelihood function $p(x_i \mid \boldsymbol{\theta}, H_j)$ and the parameters $\boldsymbol{\theta}$ follow a prior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{\phi}, H_j)$, the hypothesis' *model evidence* (or *marginal likelihood*) is given by

$$p(x_i \mid \boldsymbol{\phi}, H_j) = \int p(x_i \mid \boldsymbol{\theta}, H_j)\ p(\boldsymbol{\theta} \mid \boldsymbol{\phi}, H_j)\ d\boldsymbol{\theta} \tag{3.2}$$

With this approach, we naturally deal with overfitting by avoiding point estimates and penalizing more complex models [62, Chap.7]. For example, consider parameter b, the number of infected homes. In a frequentist model, we would need to either (1) assume it's known; or (2) estimate it based on (probably scarce) collected data. On the other hand, a Bayesian framework provides a third alternative: assuming that b is a random variable whose (posterior) distribution depends on both *prior* knowledge and available data.

3.2.1 Model evidence under null hypothesis

Under the null hypothesis H_0 (no attack in the given period), we assume that the N home routers independently generate a false positive with probability θ_F. The number of positives x_i given H_0 then follows a binomial distribution with parameters N and θ_F:

$$p(x_i \mid \theta_F, H_0) = \binom{N}{x_i} \theta_F^{x_i} (1 - \theta_F)^{N - x_i} \tag{3.3}$$

Even though it may be estimated through controlled experiments, the actual false positive rate (FPR) of the attack detector running on the home routers is unknown, and may even vary with time (e.g., due to changes in home user traffic patterns as reported in [49]). So, instead of using a point estimate (MLE), we assume that FPR is a random variable with prior distribution $p(\theta_F \mid \boldsymbol{\phi}, H_0)$. Since

the likelihood follows a binomial distribution, we adopt its *conjugate prior*, assuming $\theta_F \sim \text{Beta}(\alpha_F, \beta_F)$:

$$p(\theta_F \mid \alpha_F, \beta_F, H_0) = \frac{\theta_F^{\alpha_F - 1}(1-\theta_F)^{\beta_F - 1}}{B(\alpha_F, \beta_F)}, \tag{3.4}$$

where $B(a, b)$ is the beta function defined as $B(a, b) = \int_0^1 z^{a-1}(1-z)^{b-1}\, dz$. The *hyperparameters* α_F and β_F express our prior beliefs about θ_F. For example, they can be chosen based on controlled experiments. The hyperparameters can be interpreted as α_F previously observed false positives among $\alpha_F + \beta_F$ home routers. In general, larger values of α_F and β_F lead to a narrower distribution, while $\alpha_F = \beta_F = 1$ leads to a uniform prior over θ_F (equivalent to the absence of prior knowledge). In Chapter 3.2.4 we address the problem of adjusting hyperparameters based on prior knowledge.

Then, the model evidence under the hypothesis of no attack is given by

$$\begin{aligned} p(x_i \mid \alpha_F, \beta_F, H_0) &= \int_0^1 p(x_i \mid \theta_F, H_0)\; p(\theta_F \mid \alpha_F, \beta_F, H_0)\; d\theta_F \\ &= \binom{N}{x_i} \frac{B(x_i + \alpha_F, N - x_i + \beta_F)}{B(\alpha_F, \beta_F)}, \end{aligned} \tag{3.5}$$

known as the beta-binomial distribution.

3.2.2 Model evidence under alternative hypothesis

Let's suppose that, at time t_i, a DDoS attack is occurring. The observed number of positives x_i will account for both true positives (from homes that participate in the attack) and false positives (from homes that do not participate). Then, given a certain number of attackers, we can model the number of *false positives* and *true positives* separately. If b homes participate in the attack (from a total of N homes) and each home router detects an attack independently with probability θ_D (the classifier's true positive rate), the number of true positives D_i will follow a binomial distribution with parameters b and θ_D. Similarly, given the false positive probability θ_F the number of false positives F_i will follow a binomial distribution with parameters $N - b$ and θ_F. Since parameters θ_D and θ_F are actually unknown, we assume a beta prior distribution with parameters (resp.) α_D, β_D and α_F, β_F. Therefore, the total number of false positives F_i and true positives D_i given b attackers will both

follow a beta-binomial distribution (like in Equation 3.5):

$$\begin{aligned} F_i &\sim \text{Beta-Binomial}(N-b, \alpha_F, \beta_F), \quad \text{(False Positives)}, \\ D_i &\sim \text{Beta-Binomial}(b, \alpha_D, \beta_D), \qquad \text{(True Positives)}. \end{aligned} \tag{3.6}$$

The total number of positives x_i will be the sum of true positives D_i and false positives F_i at time t_i. The confusion matrix presented in Table 3.4 shows how latent variables F_i, D_i, b and parameter N are related.

Table 3.4: Confusion matrix summarizing model's notation.

		Predicted		
		Positive	Negative	Total
Actual	Positive	D_i	$b - D_i$	b
	Negative	F_i	$N - b - F_i$	$N - b$
	Total	x_i	$N - x_i$	N

We observe from the Probabilistic Graphical Model depicted in Figure 3.4b that F_i and D_i are conditionally independent given b. The PMF of the sum of F_i and D_i is the discrete convolution of $p(F_i \mid b, H_1)$ and $p(D_i \mid b, H_1)$:

$$\begin{aligned} &p(x_i \mid b, H_1) \\ &= \sum_{k=\gamma_i}^{\delta_i} p(F_i = x_i - k \mid b, H_1) p(D_i = k \mid b, H_1) \\ &= \sum_{k=\gamma_i}^{\delta_i} \left[\binom{N-b}{x_i-k} \frac{B(x_i - k + \alpha_F, N - b - (x_i - k) + \beta_F)}{B(\alpha_F, \beta_F)} \right. \\ &\left. \binom{b}{k} \frac{B(k + \alpha_D, b - k + \beta_D)}{B(\alpha_D, \beta_D)} \right], \end{aligned} \tag{3.7}$$

where $\gamma_i \triangleq \max\{0, x_i - (N-b)\}$ and $\delta_i \triangleq \min\{b, x_i\}$. (For $k < \gamma_i$ and $k > \delta_i$, the product $p(F_i = x_i - k \mid b, H_1) p(D_i = k \mid b, H_1)$ is zero.)

Then, the model evidence $p(x_i \mid H_1)$ can be obtained by:

$$p(x_i \mid H_1) = \sum_{b=0}^{N} p(x_i \mid b, H_1)\ p(b \mid H_1) \tag{3.8}$$

If each home participates in the attack with (unknown) probability θ_b, by assuming a prior distribution $\theta_b \sim \text{Beta}(\alpha_b, \beta_b)$ we have that the total number of attacking homes b is Beta-Binomial(N, α_b, β_b):

$$
\begin{aligned}
p(b \mid H_1) &= \int_0^1 p(b \mid \theta_b, H_1)\ p(\theta_b \mid H_1)\ d\theta_b \\
&= \binom{N}{b} \frac{B(b+\alpha_b, N-b+\beta_b)}{B(\alpha_b, \beta_b)}.
\end{aligned}
\tag{3.9}
$$

Finally, the *model evidence* $p(x_i \mid H_1)$ under the alternative hypothesis H_1 is

$$
\begin{aligned}
&p(x_i \mid H_1) \\
&= \sum_{b=0}^{N} p(x_i \mid b, H_1)\ p(b \mid H_1) \\
&= \sum_{b=0}^{N} \left[\binom{N}{b} \frac{B(b+\alpha_b, N-b+\beta_b)}{B(\alpha_b, \beta_b)} \right. \\
&\sum_{k=\gamma_i}^{\delta_i} \left[\binom{N-b}{x_i-k} \frac{B(x_i-k+\alpha_F, N-b-(x_i-k)+\beta_F)}{B(\alpha_F, \beta_F)} \right. \\
&\left. \left. \binom{b}{k} \frac{B(k+\alpha_D, b-k+\beta_D)}{B(\alpha_D, \beta_D)} \right] \right].
\end{aligned}
\tag{3.10}
$$

We note that the model evidence under the null hypothesis $p(x_i \mid H_0)$, (3.5), is a special case of $p(x_i \mid H_1)$ with $p(b = 0 \mid H_0) = 1$ and $p(D_i = 0 \mid H_0) = 1$, i.e., we are certain there were no attacking homes in the period. Then, given H_0, we have $x_i = F_i$ while given H_1 we have $x_i = F_i + D_i$.

Table 3.5 summarizes the distributions of latent variables b, F_i and D_i and observable variable x_i.

3.2.3 Bayes factor

After defining the model evidence under both hypotheses, we compute the corresponding Bayes factor BF_{10}:

$$
\begin{aligned}
&\mathrm{BF}_{10}(x_i, \alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D, \beta_D) \triangleq \frac{p(x_i \mid H_1)}{p(x_i \mid H_0)} \\
&= \sum_{b=0}^{N} \Bigg[\binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k=\gamma_i}^{\delta_i} \binom{N-b}{x_i - k} B(x_i - k + \alpha_F, N - b - (x_i - k) + \beta_F) \\
&\binom{b}{k} B(k + \alpha_D, b - k + \beta_D) \Bigg] \div \\
&\left[\binom{N}{x_i} B(x_i + \alpha_F, N - m + \beta_F) B(\alpha_b, \beta_b) B(\alpha_D, \beta_D) \right].
\end{aligned}
\tag{3.11}
$$

Therefore, after observing x_i positive results from N homes at a given period, we can use (3.11) to evaluate the evidence in favor of each hypothesis. In case of evidence in favor of an attack, the network operator can then take appropriate countermeasures depending on the strength of evidence (e.g, anecdotal, substantial, decisive).

Table 3.5: Distribution of latent and observable variables.

Variable	Likelihood	Prior	Marginal	Posterior
b	Binom(N, θ_b)	Beta(α_b, β_b)	Beta-Binom(N, α_b, β_b)	Beta(N, α'_b, β'_b)
F_i	Binom$(N - b, \theta_F)$	Beta(α_F, β_F)	Beta-Binom$(N - b, \alpha_F, \beta_F)$	Beta$(N - b, \alpha'_F, \beta'_F)$
D_i	Binom(b, θ_D)	Beta(α_D, β_D)	Beta-Binom(b, α_D, β_D)	Beta(b, α'_D, β'_D)
x_i	Equation 3.7	Beta-Binom(N, α_b, β_b)	Equation 3.10	Equation 3.12

When we decide that the hypothesis of attack is most likely correct, we can infer the number of infected homes (that participate in the attack). Since the classifier running at the home routers is not perfect, the observed number of positives x_i may be biased due to undetected attackers (false negatives) and/or false positives from homes that are not infected.

The posterior distribution of the number of attackers can be obtained from (3.7), (3.9), and (3.10):

$$
\begin{aligned}
p(b \mid x_i, H_1) &= \frac{p(x_i \mid b, H_1)\ p(b \mid H_1)}{p(x_i \mid H_1)} \\
&= \Bigg[\binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k=\gamma_i}^{\delta_i} \binom{N-b}{x_i - k} B(x_i - k + \alpha_F, N - b - (x_i - k) + \beta_F) \\
&\binom{b}{k} B(k + \alpha_D, b - k + \beta_D) \Bigg] \div \\
&\Bigg[\sum_{\hat{b}=0}^{N} \binom{N}{\hat{b}} B(\hat{b} + \alpha_b, N - \hat{b} + \beta_b) \\
&\sum_{k=0}^{\hat{b}} \binom{N-\hat{b}}{x_i - k} B(x_i - k + \alpha_F, N - \hat{b} - (x_i - k) + \beta_F) \\
&\binom{\hat{b}}{k} B(k + \alpha_D, \hat{b} - k + \beta_D) \Bigg].
\end{aligned}
\tag{3.12}
$$

3.2.4 Model updating

Given an observed number of positive results x_i, we show how to compute the evidence in favor of a DDoS attack at time t_i using the Bayes Factor, which is the ratio between the model evidences under two competing hypotheses (models). The model evidences depend not only on x_i, but also on the prior distributions of parameters (latent variables) θ_b, θ_F, and θ_D under H_0 and H_1. However, over time, knowledge about the occurrence of DDoS attacks can be acquired from past observations, allowing us to refine our understanding of whether an attack took place at a particular moment. In this section, we describe how the posterior predictive distribution can be used to update the model and incorporate beliefs derived from these past observations, improving the accuracy of future predictions.

Sampling during normal period

Suppose we observed x_n positive results at time t_n. If we believe that it is very unlikely that a DDoS attack occurred at t_n, we can use that information to update our model's hyperparameters. Since the number of attackers is (most likely) zero at t_n, $D_n = 0$ and $x_n = F_n$. The number of false positives F_n follows a beta-binomial distribution with parameters N, α_F and β_F (see Table 3.5). Given F_n, the number of false positives F_j at a later time t_j ($j > n$) also follows a beta-binomial distribution. This follows directly from the fact that the Beta distribution is the conjugate prior for the binomial distribution. Additionally, in this case, it is well known that the

parameters of the posterior predictive distribution $p(F_j \mid x_n)$ are given by

$$\alpha'_F = \alpha_F + x_n, \qquad \beta'_F = \beta_F + N - x_n. \tag{3.13}$$

Given sample x_n, the Bayes factor (3.11) for a new observation x_j with the updated model is

$$\begin{aligned} &\mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D, \beta_D \mid x_n) \\ &= \mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \alpha'_F, \beta'_F, \alpha_D, \beta_D). \end{aligned} \tag{3.14}$$

The hyperparameter update rule (3.13) can be applied recursively when more samples become available. In general, for a set $\mathcal{X}_n = \{x_1, x_2, \ldots, x_m\}$ of samples collected at times $t_1, t_2, \ldots, t_m$ with no attacks, the update rule is

$$\begin{aligned} \alpha'_F &= \alpha_F + \sum_{\forall x_i \in \mathcal{X}_n} x_i, \\ \beta'_F &= \beta_F + |\mathcal{X}_n| N - \sum_{\forall x_i \in \mathcal{X}_n} x_i. \end{aligned} \tag{3.15}$$

Following this method, the knowledge about θ_F (the false positive probability of the classifier running on the home routers) can be updated over time based on the observations during periods with (very likely) no attacking homes.

Sampling during attack period

Suppose now that x_a positive results were observed at time t_a, when a DDoS attack was in progress. How should x_a influence our beliefs when a new sample x_j ($j > a$) is observed? We showed in Section 3.2.2 that x_a is given by the sum of false positives F_a and true positives (detections) D_a at time t_a. Since we cannot easily distinguish between false and true positives (we do not know the actual number of attackers b), we are not able to devise a direct update rule for the hyperparameters α_F, β_F, α_D, β_D, α_b or β_b.

From the Probabilistic Graphical Model in Figure 3.4b, it can be seen that samples x_a and x_j are conditionally independent given latent variables b, θ_F and θ_D (hypothesis H_1). Therefore, if we consider that the number of attackers b is roughly the same at t_a and t_j [2], we can calculate the updated model evidence under H_1 from

[2]This is more probable when we are looking at shorter time spans and attacks originate from the same botnet. We can evaluate if this is a reasonable assumption by comparing the posterior distributions $p(b \mid x_a, H_1)$ and $p(b \mid x_j, H_1)$.

the Law of Total Probability:

$$
\begin{aligned}
&p(x_j \mid x_a, H_1) \\
&= \frac{1}{p(x_a \mid H_1) B(\alpha_b, \beta_b) B(\alpha_F, \beta_F) B(\alpha_D, \beta_D)} \\
&\Bigg[\sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k_j=\gamma_j}^{\delta_j} \sum_{k_a=\gamma_a}^{\delta_a} B(\alpha_F + x_j + x_a - k_j - k_a, \beta_F + \\
&2(N-b) + k_j + k_a - x_j - x_a) \\
&B(\alpha_D + k_j + k_a, \beta_D + 2b - k_j - k_a) \\
&\binom{N-b}{x_j - k_j} \binom{b}{k_j} \binom{N-b}{x_a - k_a} \binom{b}{k_a} \Bigg].
\end{aligned}
\tag{3.16}
$$

Similarly, x_a and x_j are conditionally independent given θ_F (hypothesis H_0). Therefore, for the hypothesis of no attack at t_j, we have

$$
\begin{aligned}
&p(x_j \mid x_a, H_0) \\
&= \frac{1}{p(x_a \mid H_1) B(\alpha_b, \beta_b) B(\alpha_F, \beta_F) B(\alpha_D, \beta_D)} \\
&\Bigg[\binom{N}{x_j} \sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k_a=\gamma_a}^{\delta_a} B(\alpha_F + x_j + x_a - k_a, \beta_F + 2N - b + k_a - x_j - x_a) \\
&B(\alpha_D + k_a, \beta_D + b - k_a) \binom{N-b}{x_a - k_a} \binom{b}{k_a} \Bigg].
\end{aligned}
\tag{3.17}
$$

Finally, the updated Bayes factor for x_j given sample x_a follows directly from (3.16) and (3.17):

$$
\mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D, \beta_D \mid x_a) = \frac{p(x_j \mid x_a, H_1)}{p(x_j \mid x_a, H_0)}. \tag{3.18}
$$

Comparing (3.16) to (3.10) and (3.17) to (3.5), we see that the posterior predictive distribution $p(x_j \mid x_a, H_j)$ and the marginal likelihood $p(x_a \mid H_j)$, $j = 0, 1$, follow different distributions for both hypotheses. As a result, given a set of multiple samples $\mathcal{X}_a = x_1, x_2, \ldots, x_m$ collected during attack times $t_1, t_2, \ldots, t_m$, there is no recursive update rule like (3.15):

$$
p(x_j \mid \mathcal{X}_a, H_1) = \sum_{b=0}^{N} \int_0^1 \int_0^1 p(x_j \mid b, \theta_F, \theta_D, H_1)\; p(b, \theta_F, \theta_D \mid \mathcal{X}_a, H_1)\; d\theta_D\; d\theta_F
$$

$$= \sum_{b=0}^{N} \int_0^1 \int_0^1 p(x_j \mid b, \theta_F, f_D, H_1) \frac{p(\mathcal{X}_a \mid b, \theta_F, \theta_D, H_1)\ p(b, \theta_F, \theta_D)}{p(\mathcal{X}_a \mid H_1)}\ d\theta_D\ d\theta_F$$

$$= \sum_{b=0}^{N} \int_0^1 \int_0^1 \frac{p(x_j, \mathcal{X}_a \mid b, \theta_F, \theta_D, H_1)\ p(b)\ p(\theta_F)\ p(\theta_D)}{p(\mathcal{X}_a \mid H_1)}\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D) p(x_j, \mathcal{X}_a \mid b, \theta_F, \theta_D, H_1)\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D) \prod_{i=1}^{j} p(x_i \mid b, \theta_F, \theta_D, H_1)\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D)$$

$$\prod_{i=1}^{j} \sum_{k=\gamma_i}^{\delta_i} p(F_i = x_i - k \mid b, \theta_F, H_1) p(D_i = k \mid b, \theta_D, H_1)\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D)$$

$$\prod_{i=1}^{j} \sum_{k=\gamma_i}^{\delta_i} \binom{N-b}{x_i-k} \theta_F^{x_i-k} (1-\theta_F)^{N-b-x_i+k} \binom{b}{k} \theta_D^{k} (1-\theta_D)^{b-k}\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D)$$

$$\sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} \binom{N-b}{x_1-k_1} \cdots \binom{N-b}{x_j-k_j} \theta_F^{x_1-k_1} \ldots \theta_F^{x_j-k_j}$$

$$(1-\theta_F)^{N-b-x_1+k_1} \ldots (1-\theta_F)^{N-b-x_j+k_j} \binom{b}{k_1} \cdots \binom{b}{k_j}$$

$$\theta_D^{k_1} \ldots \theta_D^{k_j} (1-\theta_D)^{b-k_1} \ldots (1-\theta_D)^{b-k_j}\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \int_0^1 p(\theta_F) \int_0^1 p(\theta_D)$$

$$\sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} \prod_{i=1}^{j} \binom{N-b}{x_i-k_i} \binom{b}{k_i}$$

$$\theta_F^{\sum_s x_s - k_s} (1-\theta_F)^{j(N-b)+\sum_s k_s - x_s}$$

$$\theta_D^{\sum_s k_s} (1-\theta_D)^{jb-\sum_s k_s}\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} \prod_{i=1}^{j} \binom{N-b}{x_i-k_i} \binom{b}{k_i}$$

$$\int_0^1 p(\theta_F)\ \theta_F^{\sum_s x_s - k_s} (1-\theta_F)^{j(N-b)+\sum_s k_s - x_s}$$

$$\int_0^1 p(\theta_D)\ \theta_D^{\sum_s k_s} (1-\theta_D)^{jb-\sum_s k_s}\ d\theta_D\ d\theta_F$$

$$= \frac{1}{p(\mathcal{X}_a \mid H_1)} \sum_{b=0}^{N} p(b) \sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} \prod_{i=1}^{j} \binom{N-b}{x_i-k_i} \binom{b}{k_i}$$

$$\int_0^1 \frac{\theta_F^{\alpha_F-1}(1-\theta_F)^{\beta_F-1}}{B(\alpha_F, \beta_F)}\ \theta_F^{\sum_s x_s - k_s} (1-\theta_F)^{j(N-b)+\sum_s k_s - x_s}$$

$$\int_0^1 \frac{\theta_D^{\alpha_D-1}(1-\theta_D)^{\beta_D-1}}{B(\alpha_D, \beta_D)}\ \theta_D^{\sum_s k_s} (1-\theta_D)^{jb-\sum_s k_s}\ d\theta_D\ d\theta_F$$

$$
\begin{aligned}
&= \frac{1}{p(\mathcal{X}_a \mid H_1) B(\alpha_F, \beta_F) B(\alpha_D, \beta_D)} \sum_{b=0}^{N} p(b) \sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} \prod_{i=1}^{j} \binom{N-b}{x_i - k_i} \binom{b}{k_i} \\
&\int_0^1 \theta_F^{\alpha_F - 1 + \sum_s x_s - k_s} (1-\theta_F)^{\beta_F - 1 + j(N-b) + \sum_s k_s - x_s} \\
&\int_0^1 \theta_D^{\alpha_D - 1 + \sum_s k_s} (1-\theta_D)^{\beta_D - 1 + jb - \sum_s k_s} \, d\theta_D \, d\theta_F \\
&= \frac{1}{p(\mathcal{X}_a \mid H_1) B(\alpha_F, \beta_F) B(\alpha_D, \beta_D)} \sum_{b=0}^{N} p(b) \\
&\sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} B(\alpha_F + \sum_s x_s - k_s, \beta_F + j(N-b) + \sum_s k_s - x_s) \\
&B(\alpha_D + \sum_s k_s, \beta_D + jb - \sum_s k_s) \prod_{i=1}^{j} \binom{N-b}{x_i - k_i} \binom{b}{k_i} \\
&= \frac{1}{p(\mathcal{X}_a \mid H_1) B(\alpha_b, \beta_b) B(\alpha_F, \beta_F) B(\alpha_D, \beta_D)} \sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k_1=\gamma_1}^{\delta_1} \cdots \sum_{k_j=\gamma_j}^{\delta_j} B(\alpha_F + \sum_s x_s - k_s, \beta_F + j(N-b) + \sum_s k_s - x_s) \\
&B(\alpha_D + \sum_s k_s, \beta_D + jb - \sum_s k_s) \prod_{i=1}^{j} \binom{N-b}{x_i - k_i} \binom{b}{k_i}
\end{aligned} \tag{3.19}
$$

Detailed sampling during an attack

In the case where sampling is done during an attack, we show that it is not possible to directly update the model's hyperparameters by observing only variable x_a. To update the hyperparameters, it is necessary to also observe latent variables b, F_i and D_i during the attack. Lets assume this information is available for a subset of $\tilde{N} < N$ homes. For example, an ISP could give instrumented devices that act like honeypots to some of it's clients. If an attack happens, we can infer, among the $\tilde{N}$ homes, the number of attackers $\tilde{b}$. From $\tilde{b}$ and the result of each DDoS detector at the home routers, we are able to compute the number of false positives $\tilde{F}$ (positive results from the $\tilde{N} - \tilde{b}$ homes that did not participate in the attack) and the number of true positives $\tilde{D}$ (positive results among the $\tilde{b}$ attackers). Since b, F_i and D_i follow a beta-binomial distribution, the corresponding posterior predictive distributions given $\tilde{b}$, $\tilde{F}$ and $\tilde{D}$ is also beta-binomial. Similar to (3.13), the model update rule is given by

$$
\begin{aligned}
&\alpha_b' = \alpha_b + \tilde{b}, \quad \beta_b' = \beta_b + \tilde{N} - \tilde{b} \\
&\alpha_F' = \alpha_F + \tilde{F}, \quad \beta_F' = \beta_F + \tilde{N} - \tilde{b} - \tilde{F} \\
&\alpha_D' = \alpha_D + \tilde{D}, \quad \beta_D' = \beta_D + \tilde{b} - \tilde{D}.
\end{aligned} \tag{3.20}
$$

The Bayes factor for a new sample x_j is

$$
\begin{aligned}
&\mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D, \beta_D \mid \tilde{N}, \tilde{b}, \tilde{F}, \tilde{D}) \\
&= \mathrm{BF}_{10}(x_j, \alpha_b', \beta_b', \alpha_F', \beta_F', \alpha_D', \beta_D')
\end{aligned} \tag{3.21}
$$

As in (3.15), the update rule (3.20) can be applied recursively when more data is available.

3.2.5 Known classifier performance (binomial attack model)

After a period of observation of the classifier results and subsequent model updates, the uncertainty about the performance of the classifier running on the home routers (i.e., the True Positive Rate (TPR) and False Positive Rate (FPR)) will decrease considerably. As more data becomes available and the parameters α and β of the beta prior for TPR and FPR grow large, the beta distribution becomes more sharply peaked, converging to a point estimate. In this case, the Bayesian model can be simplified by treating the TPR and FPR as fixed values rather than uncertain variables.

The beta-binomial distribution with integer α and β can be used to model a simple Pólya urn: the urn starts with α red balls and β black balls; when a ball is drawn from the urn, it is replaced in the urn together with a new ball of the same color. After k random draws from the urn, the number of red balls drawn follows a beta-binomial distribution with parameters k, α and β.

Intuitively, when $\alpha \gg k$ and $\beta \gg k$ (the number of balls inside the urn before drawings is much larger than the number of balls drawn), the effect of adding one ball at each of the k draws is negligible. As a consequence, the process is very similar to drawing with replacement. It is easy to show that the number of red balls drawn after k random draws approaches a binomial distribution with parameters k and $\frac{\alpha}{\alpha+\beta}$. The result holds true when α and β are positive reals.

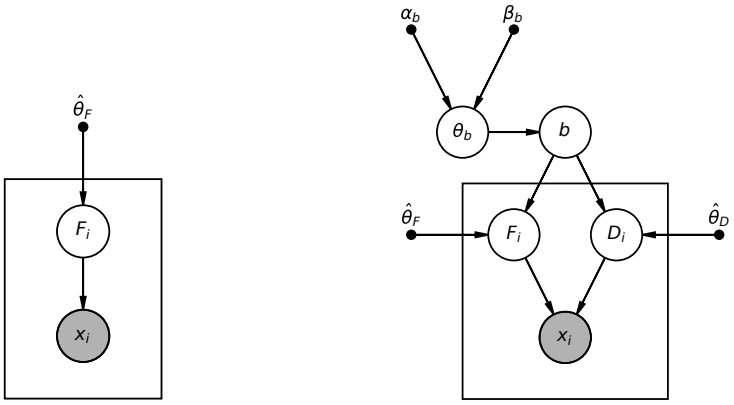
Let $\kappa_F = \alpha_F + \beta_F$ and $\kappa_D = \alpha_D + \beta_D$ be the *concentration* of the prior distribution over, respectively, θ_F and θ_D. The beta distribution becomes narrower as its concentration increases [63, Chap.6]. Therefore, for sufficiently large κ_F and κ_D, latent variables θ_F and θ_D can be modeled, instead, as deterministic parameters. In this case, it is easy to see that the distribution of F_i and D_i given b attackers (3.6) will follow a binomial distribution with parameters, respectively, $\hat{\theta}_F = \frac{\alpha_F}{\alpha_F+\beta_F}$ and $\hat{\theta}_D = \frac{\alpha_D}{\alpha_D+\beta_D}$:

$$\begin{aligned} F_i &\sim \text{Binomial}(N-b, \hat{\theta}_F), \quad \text{(False Positives)}, \\ D_i &\sim \text{Binomial}(b, \hat{\theta}_D), \quad \text{(True Positives)}. \end{aligned} \tag{3.22}$$

We call this model the *binomial attack model*. We replace the latent variables θ_F and θ_D of the beta-binomial attack model (Figure 3.4) with $\hat{\theta}_F$ and $\hat{\theta}_D$ as shown in the Probabilistic Graphical Model (PGM) of Figure 3.5. The observed number of positives x_i at t_i are now conditionally independent given only one latent variable (b). In the beta-binomial model, samples were conditionally independent given three latent variables: b, θ_F and θ_D. We will show that this assumption leads to some simplifications.

Given b attackers, the number of positives x_i is given by the sum of two (conditionally) independent binomial random variables F_i and D_i with parameters θ_F and θ_D, respectively. Therefore, x_i follows a J-binomial distribution [64] with $J = 2$. As before (3.7), the PMF of the total number of positives x_i is given by the discrete convolution of $p(F_i \mid b, H_1)$ and $p(D_i \mid b, H_1)$ [3]. The Bayes

[3] Alternatively, an approximate solution can be obtained via saddlepoint approximation [65].



(a) Hypothesis H_0 (no attack). (b) Hypothesis H_1 (attack).

Figure 3.5: Probabilistic Graphical Model representation under each hypothesis assuming known $\hat{\theta}_F$ and $\hat{\theta}_D$ (*binomial attack model*).

factor is

$$\begin{aligned}
&\mathrm{BF}_{10}(x_i, \alpha_b, \beta_b, \hat{\theta}_F, \hat{\theta}_D) \\
&= \Bigg[\sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k=\gamma_i}^{\delta_i} \binom{N-b}{x_i - k} \hat{\theta}_F^{x_i - k} (1 - \hat{\theta}_F)^{N - b - (x_i - k)} \\
&\binom{b}{k} \hat{\theta}_D^{k} (1 - \hat{\theta}_D)^{b-k} \Bigg] \div \\
&\left[\binom{N}{x_i} \hat{\theta}_F^{x_i} (1 - \hat{\theta}_F)^{N - x_i} B(\alpha_b, \beta_b) \right].
\end{aligned} \tag{3.23}$$

The posterior distribution of the number of attackers given x_i positive results (equivalent to Equation 3.12) is

$$\begin{aligned}
&p(b \mid x_i, H_1) \\
&= \Bigg[\binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k=\gamma_i}^{\delta_i} \binom{N-b}{x_i - k} \hat{\theta}_F^{x_i - k} (1 - \hat{\theta}_F)^{N - b - (x_i - k)} \\
&\binom{b}{k} \hat{\theta}_D^{k} (1 - \hat{\theta}_D)^{b-k} \Bigg] \div \\
&\Bigg[\sum_{\hat{b}=0}^{N} \binom{N}{\hat{b}} B(\hat{b} + \alpha_b, N - \hat{b} + \beta_b) \\
&\sum_{k=0}^{\hat{b}} \binom{N-\hat{b}}{x_i - k} \hat{\theta}_F^{x_i - k} (1 - \hat{\theta}_F)^{N - \hat{b} - (x_i - k)} \\
&\binom{\hat{b}}{k} \hat{\theta}_D^{k} (1 - \hat{\theta}_D)^{\hat{b}-k} \Bigg].
\end{aligned} \tag{3.24}$$

Since the false positive probability is now given by hyperparameter $\hat{\theta}_F$, the model evidence

under H_0 is not affected by past samples, i.e., $p(x_j \mid x_i, H_0) = p(x_j \mid H_0)$ for $i < j$. However, if we know that an attack is occurring at time t_a and x_a positive results are observed, that information can be used to adjust our beliefs under hypothesis H_1. The Bayes factor for a new sample x_j is

$$\mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \hat{\theta}_F, \hat{\theta}_D \mid x_a) = \frac{p(x_j \mid x_a, H_1)}{p(x_j \mid H_0)}, \tag{3.25}$$

where the posterior predictive distribution for H_1 is given by

$$\begin{aligned}
&p(x_j \mid x_a, H_1) \\
&= \frac{1}{p(x_a \mid H_1) B(\alpha_b, \beta_b)} \\
&\Bigg[\sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b) \\
&\sum_{k_j=\gamma_j}^{\delta_j} \sum_{k_a=\gamma_a}^{\delta_a} \hat{\theta}_F^{x_j + x_a - k_j - k_a} (1 - \hat{\theta}_F)^{2(N-b) + k_j + k_a - x_j - x_a} \\
&\hat{\theta}_D^{k_j + k_a} (1 - \hat{\theta}_D)^{2b - k_j - k_a} \binom{N-b}{x_j - k_j} \binom{b}{k_j} \binom{N-b}{x_a - k_a} \binom{b}{k_a} \Bigg].
\end{aligned} \tag{3.26}$$

Finally, when a subset of $\tilde{N} < N$ homes is monitored so that the exact number of attackers $\tilde{b}$ (out of $\tilde{N}$) at a given time can be observed, we can use the simple update rule

$$\alpha_b' = \alpha_b + \tilde{b}, \qquad \beta_b' = \beta_b + \tilde{N} - \tilde{b}. \tag{3.27}$$

Update rule (3.27) leads to

$$\mathrm{BF}_{10}(x_j, \alpha_b, \beta_b, \hat{\theta}_F, \hat{\theta}_D \mid \tilde{N}, \tilde{b}) = \mathrm{BF}_{10}(x_j, \alpha_b', \beta_b', \hat{\theta}_F, \hat{\theta}_D). \tag{3.28}$$

3.2.6 Poisson approximation

Our model can be further simplified when the number of homes N is sufficiently large. In this scenario, the binomial distribution of F_i and D_i can be well approximated by the Poisson distribution. Then, we can use the property that the sum of two independent Poisson random variables with parameters λ_1 and λ_2 follows a Poisson distribution with parameter $\lambda = \lambda_1 + \lambda_2$ (we may avoid the convolution operation used to compute the PMF of $x_i = F_i + D_i$ given b). Using this approximation, we have

$$\begin{aligned}
F_i &\sim \mathrm{Poisson}((N - b)\ \hat{\theta}_F) && \text{(False Positives)} \\
D_i &\sim \mathrm{Poisson}(b\ \hat{\theta}_D) && \text{(True Positives)} \\
x_i &\sim \mathrm{Poisson}((N - b)\ \hat{\theta}_F + b\ \hat{\theta}_D) && \text{(Predicted Positives)}
\end{aligned} \tag{3.29}$$

We denote this as the *Poisson attack model*. The Poisson model shares the binomial attack model's PGM structure (Figure 3.5), with different distributions for F_i, D_i and x_i.

The Bayes factor is given by

$$\begin{aligned}
&\mathrm{BF}_{10}(x_i, \alpha_b, \beta_b, \hat{\theta}_F, \hat{\theta}_D) \\
&= \frac{\sum_{b=0}^{N} \binom{N}{b} B(b + \alpha_b, N - b + \beta_b)\ \mu^{x_i} e^{b(\hat{\theta}_F - \hat{\theta}_D)}}{N^{x_i}\ \hat{\theta}_F^{x_i}\ B(\alpha_b, \beta_b)},
\end{aligned} \tag{3.30}$$

where $\mu \triangleq (N-b)\ \hat{\theta}_F + b\ \hat{\theta}_D$.

Comparing (3.30) to (3.11) and (3.23), it is clear that the complexity of computing the Bayes factor reduces from $\mathcal{O}(N^2)$ to $\mathcal{O}(N)$.

The posterior distribution of the number of attackers b given x_i is

$$\begin{aligned} & p(b \mid x_i, H_1) \\ & = \frac{\binom{N}{b} B(b+\alpha_b, N-b+\beta_b)\ \mu^{x_i} e^{b(\hat{\theta}_F - \hat{\theta}_D)}}{\sum_{\hat{b}=0}^{N} \binom{N}{\hat{b}} B(\hat{b}+\alpha_b, N-\hat{b}+\beta_b)\ \mu^{x_i} e^{\hat{b}(\hat{\theta}_F - \hat{\theta}_D)}}. \end{aligned} \tag{3.31}$$

Finally, given that x_a positives where observed during an attack at t_a, the posterior predictive distribution for the hypothesis of attack is

$$\begin{aligned} & p(x_j \mid x_a, H_1) \\ & = \frac{1}{p(x_a \mid H_1) B(\alpha_b, \beta_b)} \\ & \left[\sum_{b=0}^{N} \binom{N}{b} B(b+\alpha_b, N-b+\beta_b) \frac{\mu^{x_j + x_a} e^{-2\mu}}{x_j! x_a!} \right]. \end{aligned} \tag{3.32}$$

The model update rule for a known number of attackers $\tilde{b}$ among $\tilde{N} < N$ homes is given by Equation 3.27.

Usually, the Poisson distribution is a good approximation for the binomial distribution when the variance $n\ p\ (1-p)$ is large. This should not happen in our scenario since the products $\hat{\theta}_F\ (1-\hat{\theta}_F)$ and $\hat{\theta}_D\ (1-\hat{\theta}_D)$ are both supposed to be low for an attack detector that is reasonably good. We show in Section 5 that the Poisson model can be used in place of the binomial model without a noticeable difference when N is reasonably large.

3.2.7 Known probability of participating in the attack

When the concentration of the prior distribution over θ_b, denoted as $kappa_b = \alpha_b + \beta_b$, is sufficiently large, we can assume the probability of a home participating in the attack is not a latent variable but a known parameter. This assumption, along with the assumption of known θ_F and θ_D, leads to the model we proposed in [61] and was later applied by Streit et al. [48]. Although simpler, this model may have some limitations. Good estimates of the expected number of bots participating in future attacks are very hard to obtain. Antonakakis et al. [66] showed not only that the size of Mirai botnets varied considerably over time, but also that some ASes where much more affected than others. The model presented here deals with the uncertainty about the parameter θ_b introducing a prior distribution.

3.3 Implementation

We briefly discuss the practical challenges associated with implementing the proposed system including both layers.

3.3.1 Local detection layer

For each residence, the first layer requires its home router to extract features from the upload and download traffic passing through the router. Using a pre-trained Random Forest classifier, the

router determines whether any device within the home network is participating in an attack. The byte and packet counters required to obtain the expected number of uploaded and downloaded bytes every minute are easily accessible in Unix systems (e.g., through the virtual file located at `/proc/net/dev`). For the Random Forest classifier, the necessary features consist of simple statistics (such as maximum, minimum, and standard deviation) calculated over a sliding window of five samples, each representing one minute.

Embedding the trained classifier is straightforward. Random Forest classifiers average the predictions from multiple Decision Trees [67], essentially simple sequences of `if-else` statements. The `sklearn-porter` tool [68] can be used to convert a trained `scikit-learn` model into C source code, making it suitable for efficient use in embedded systems with limited resources.

The OpenWrt system provides some out-of-the-box tools that can be used to implement countermeasures and minimize collateral damage when an attack is detected. Some examples include:

- Triggering SNMP traps to notify the network manager with the `mini-snmpd` daemon
- Dynamic IP blocking via `ipset` or `banIP`
- MAC address filtering via OpenWrt's `firewall` tool
- Rate limiting using OpenWrt's Traffic Control (`tc`).

Our evidence shows that implementing the first layer in the home router does not impact its performance. We installed this layer using off-the-shelf home routers running OpenWrt (the TP-Link models WDR3500, WDR3600, and WDR4300). The additional CPU and memory load on the home routers was minimal.

3.3.2 Spatio-temporal correlation layer

The second layer utilizes the information provided by the residential routers' classifiers, that may consist of just a single bit of information. More data could be sent in order to run the ML classifier on the edge server instead. However, since attacks generate a lot of upstream traffic, this would increase the likelihood of delaying an alarm or losing data from affected routers.

For N routers, the worst-case complexity for calculating the Bayes factor is $\mathcal{O}(N^2)$ for the beta-binomial and binomial models, and $\mathcal{O}(N)$ for the Poisson model. In addition to being computationally efficient (avoiding the need for sampling methods such as MCMC), Bayes factor (BF) values can be precomputed on the server for each value of x_i (ranging from 0 positive results to N). The server only needs to update the stored BF values if new evidence prompts an update to the prior parameters (see Section 3.2.4).

Multiple commercial and open-source solutions are available to mitigate the impact of an ongoing attack detected at the network edge. Some strategies include: traffic filtering, blackhole routing, rate limiting, and scrubbing centers. As discussed in Section 3.2.3, the network operator may implement different countermeasures depending on the strength of evidence provided by the Bayes factor.

Chapter 4

Datasets

This section introduces the datasets used to evaluate the proposed two-layer DDoS detection system. We start by describing in Section 4.1 how two labeled DDoS datasets were built, combining real home user traffic data with Mirai and BASHLITE traffic. We use this dataset to adjust our model parameters. Section 4.2 presents the experiment we conducted with a selected group of volunteers to obtain a dataset containing *real DDoS attacks*, the "volunteers dataset".

4.1 Labeled DDoS datasets

One of the main obstacles for building high-performance DDoS attack detection systems is the scarcity of labeled DDoS datasets [21]. The available alternatives are very limited: some are old (like the CAIDA 2007 dataset [23] and the NSL-KDD dataset [24]), others rely solely on synthetic data (like the CIC-DDoS2019 [26], UNSW-NB [25] and Bot-IoT [27] datasets). To our knowledge, the best DDoS datasets available are the ones that use testbeds with emulated traffic from selected IoT devices: 9 IoT devices in the N-BaIoT dataset [28], 3 in the IoT-23 dataset [30], 2 in McDermott et al.'s dataset [29], 26 in the ACI IoT Network Traffic Dataset 2023 [31], and 4 in the LATAM-DDoS-IoT Dataset [22]. The recent CIC-BCCC-NRC TabularIoTAttack-2024 dataset [32] provides a collection of 9 preprocessed datasets from multiple authors, with 83 features extracted from PCAP trace files. Results obtained from such scenarios may not generalize well, since attack detectability may vary a lot for different IoT models [69].

These datasets span a variety of devices, and network activity was generated in a controlled lab environment using various device combinations that emulated a smart home environment. In contrast, our dataset includes traffic data from over four thousand residences and contains more than ten thousand identifiable devices. The devices in our dataset consist of different brands of smartphones, smart TVs, printers, cameras, laptops, tablets, smartwatches, video game consoles, home theater systems, iPods, and Raspberry Pi devices.

Obtaining labels for DDoS attacks is challenging, primarily because it's difficult to determine when these attacks occur. Additionally, real data from home users is scarce, partly due to privacy concerns. Our work addresses these challenges by proposing a novel approach. We combine real home user traffic data, the baseline for regular traffic, with DDoS attack traffic generated in a controlled environment using authentic malware source code. This method enables us to create a dataset with traffic from thousands of IoT and non-IoT devices.

4.1.1 Baseline traffic dataset

We briefly introduce our baseline traffic dataset and then present a longitudinal analysis derived from the traces to appreciate the general characteristics of the home users being considered.

From a mid-size ISP, we obtained samples of the *total upload and download traffic* flowing through home routers of 4,870 residences from 14 different cities. No information from either the header or the payload of packets was collected. The samples were obtained during a period of 20 days, from May 1 through May 20, 2021. The dataset includes byte and packet counts for upload and download traffic on the LAN interface, capturing both wired and wireless traffic within individual homes, sampled at minute intervals. Data was gathered from off-the-shelf wireless home routers, which serve as gateways to the Internet provider as shown in Figure 4.1. These home routers run OpenWrt, each including open-source software to collect and send information to a server.

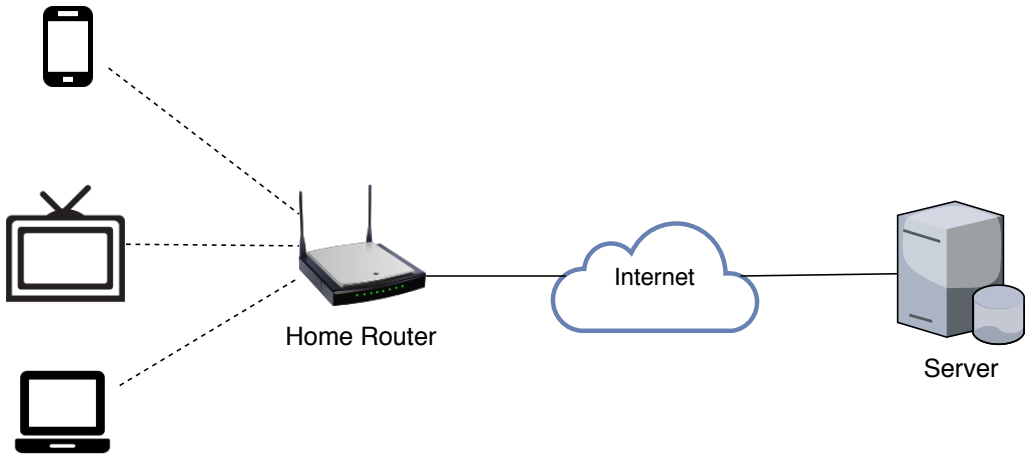


Figure 4.1: Measurement Infrastructure

Distribution of devices

We make no assumptions about the types of devices connected to individual home networks. In fact, we had no *prior* knowledge about network usage or the types of devices adopted by home users and we did not apply any pre-filtering of devices. In addition, unlike previous work [28, 29, 32, 39–43], our methodology is completely agnostic to the nature of the devices and does not rely on their specifics.

To determine whether our dataset includes residences with a wide variety of distinct devices, we sampled approximately 12% of the $4,870$ homes in our measurement campaign to examine the MAC addresses and hostnames from DHCP tables. We found that $10,136$ distinct devices were connected at least once to the sampled home networks, averaging more than 17 unique devices per home. Additionally, half of these homes had at least 13 distinct devices. Our analysis of the MAC addresses and DHCP tables showed a variety of devices connected to the home networks, including smartphones, tablets, IP cameras, smart TVs, laptops, desktops, smartwatches, printers, video game consoles, etc.

Figure 4.2 presents a histogram of the total number of distinct devices that connected at least once to each home network. Most home routers (approximately 75%) observed 20 or less devices during the 20 days period. However, we observe a heavy tail of 6 homes (approx. 1%) with over

a hundred devices. Judging by the elevated number of "*iPhone*", "*Galaxy*" and "*Android*" devices at these networks, these are probably high traffic places.

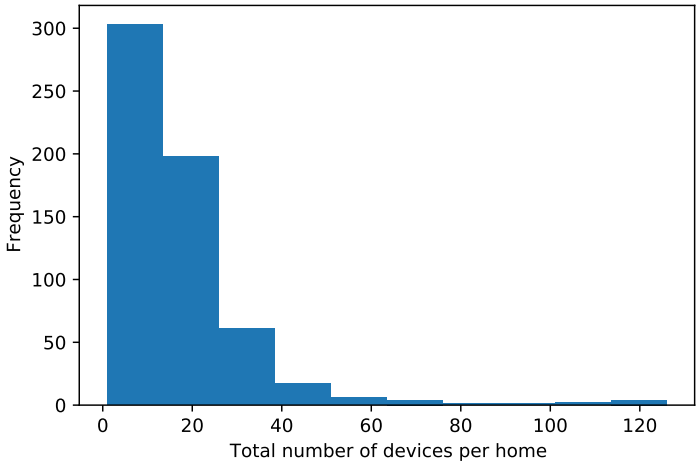


Figure 4.2: Histogram of the number of devices per home.

Distribution of upstream and downstream traffic

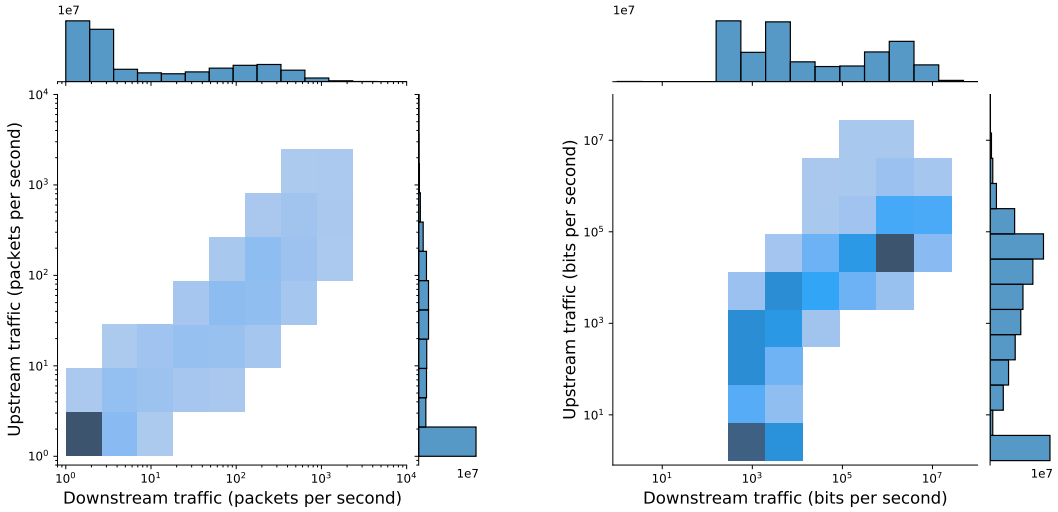
We collected approximately 116 million traffic samples (byte and packet counts) from 4,870 home routers. The average traffic in bits per second and packets per second is computed by taking the difference between the values of, respectively, the byte and packet counters at two consecutive measurements and dividing by the elapsed time.

Figure 4.3 shows the distribution of the upstream and downstream bits / packets per second rates. We can see that most packet rate samples for legitimate home user traffic are low (less than 10 pps). Both joint distributions show some correlation between the metrics. Figure 4.3a in particular indicates a high correlation between upstream and downstream packet rates. Our results (Chapter 5) suggest that the ratio between the upstream and downstream traffic rates is an important feature for the detection of DDoS attacks. At Figure 4.3b, we also see that downstream bits per second rates are usually higher than upstream rates.

4.1.2 Botnet attack experiments

Experimental setup

We setup a controlled environment at our laboratory to generate attack traffic traces using a Raspberry Pi and a TP-Link WDR3600 wireless router running OpenWrt. The Raspberry Pi runs **real malware** source code and is connected to the wireless router via Wi-Fi. The Raspberry Pi launches DDoS attacks while the OpenWrt router records the value of the network interface's bytes and packets counters at one second intervals. In this isolated network, we avoid concurrent traffic from other devices and measure almost exclusively malware traffic at the wireless (WLAN) interface. We note that we do not block the interference from neighboring Wi-Fi networks. Furthermore, by moving the devices between runs, we are able to better emulate Wi-Fi rates observed at real networks.



(a) Packets per second.

(b) Bits per second.

Figure 4.3: Distribution (joint and marginal) of upstream and downstream traffic samples.

Malware source code

We consider two of the most common IoT botnet malwares: Mirai and BASHLITE. From their publicly available source code, we obtained the functions responsible for the attack vectors of interest, removing all source code related to scanning for victims and communicating with the CnC (Command and Control) server.

In Listing 1, we show an excerpt from Mirai's source code used in UDP flood attacks. This code written in C is relatively simple: a `while (TRUE)` loop keeps sending back-to-back UDP packets to the IP addresses of each victim. The infinite loop continues until the parent process interrupts the attack with a `SIGKILL` signal. Despite the algorithm's simplicity, we show in this work that detecting Mirai and BASHLITE attacks is a nontrivial task.

By extracting the functions that implement the attack vectors, we developed a software module to conduct Mirai and BASHLITE DDoS attacks against a given target for a configurable time. The software module can launch the volumetric and state exhaustion attacks shown in Table 4.1.

Table 4.1: Attack vectors used in the experiments

Attack	Type	Source Code
UDP flood	Volumetric	Mirai, BASHLITE
UDP PLAIN flood	Volumetric	Mirai
TCP SYN flood	State exhaustion	Mirai, BASHLITE
TCP ACK flood	State exhaustion	Mirai, BASHLITE

Attack dataset

As reported in [70], most DDoS attacks last for a relatively short time, e.g. a few minutes. Additionally, according to the Mirai CnC source code, the attack duration for a single attack command is capped at 1 hour[1]. The results of Section 5.1.3 indicate that the vast majority of all attacks

[1]Due to the malware limitations, longer DDoS attacks (e.g., that last for an entire day) are probably implemented by issuing multiple attack commands with short duration.

Listing 1 Mirai source code excerpt taken from the `attack_udp_generic` function defined in the `attack_udp.c` file.

```
while (TRUE) {
    for (i = 0; i < targs_len; i++) {
        char *pkt = pkts[i];
        struct iphdr *iph = (struct iphdr *)pkt;
        struct udphdr *udph = (struct udphdr *)(iph + 1);
        char *data = (char *)(udph + 1);

        // For prefix attacks
        if (targs[i].netmask < 32)
            iph->daddr = htonl(ntohl(targs[i].addr) +
                (((uint32_t)rand_next()) >> targs[i].netmask));
        if (source_ip == 0xffffffff)
            iph->saddr = rand_next();
        if (ip_ident == 0xffff)
            iph->id = (uint16_t)rand_next();
        if (sport == 0xffff)
            udph->source = rand_next();
        if (dport == 0xffff)
            udph->dest = rand_next();
        // Randomize packet content?
        if (data_rand)
            rand_str(data, data_len);

        iph->check = 0;
        iph->check = checksum_generic((uint16_t *)iph,
                sizeof (struct iphdr));
        udph->check = 0;
        udph->check = checksum_tcpudp(
                iph, udph, udph->len,
                sizeof (struct udphdr) + data_len);
        targs[i].sock_addr.sin_port = udph->dest;
        sendto(fd, pkt, sizeof (struct iphdr) +
               sizeof (struct udphdr) + data_len,
               MSG_NOSIGNAL,
               (struct sockaddr *)&targs[i].sock_addr,
               sizeof (struct sockaddr_in));
    }
}
```

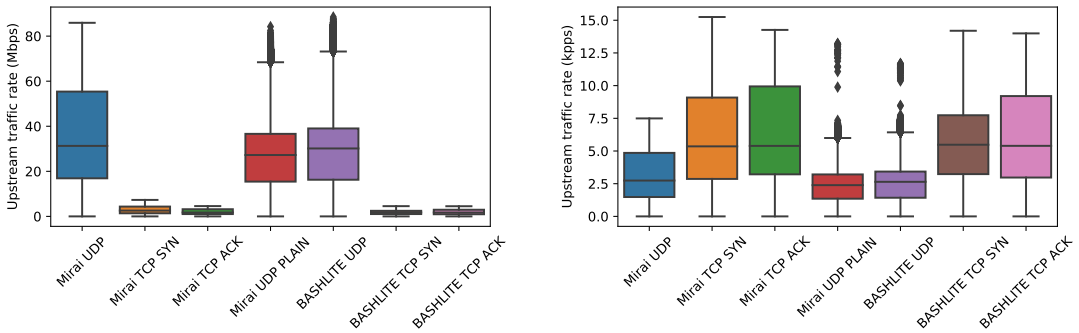
(99%) can be detected in less than two minutes from the time they are initiated. Therefore, if a short attack is detected in a few minutes, an identical longer attack will also be detected. Based on these observations, for each attack vector, we generate 300 attack samples whose duration follows a Gaussian distribution with mean $\mu_s = 120$ secs and standard deviation $\sigma_s = 10$ secs (short attacks), and 40 attack samples with mean $\mu_l = 600$ secs and standard deviation $\sigma_l = 100$ secs (long attacks). Each attack sample in the attack dataset is identified by three variables: the attack vector, the type of the attack (short or long), and the attack duration.

Our attack dataset contains seven attack vectors, four for Mirai and three for BASHLITE (Table 4.1). In particular, the UDP PLAIN attack is an optimized UDP flood attack implemented only by Mirai. Volumetric attacks (UDP) use packets with random payloads of 1400 bytes, while state exhaustion attacks (TCP) use empty packets with no payload. We consider attacks with different payload sizes in Section 5.1.4.

Figure 4.4 compares the upstream traffic for each attack vector. It is worth noting the differences among the generated attack traffic. While UDP attacks generate traffic (left plot) with a median of around 30 Mbps, other attack vectors generate traffic at rates with medians of around 2 Mbps. These lower rates are comparable to some home user upload rates (as shown in Section 4.1.1). In addition, TCP attacks that send packets with empty payload generate traffic with high packets per second (pps) rates.

One might infer that using simple threshold policies would suffice to detect attacks. However, as we show in Section 5.1.7, simple threshold policies produce unacceptably high false positive rates or low true positive rates (or both).

In summary, detecting DDoS attacks from only byte and packet counts is non-trivial and requires clever schemes. These include identifying patterns in baseline home user traffic and attack traffic, as well as exploring potential correlations between upload and download traffic.



(a) Upstream bits per second rates.

(b) Upstream packets per second rates.

Figure 4.4: Box plot of upstream traffic rates generated by each attack vector. As expected, attack vectors that send packets with large payloads generate higher bits per second (bps) rates and lower packets per second (pps) rates when compared to attacks that send empty packets with no payload.

4.1.3 Labeled dataset construction

Next, we describe how the baseline traffic dataset (Section 4.1.1) is combined with the attack dataset (Section 4.1.2) to build two labeled datasets. *Labeled dataset 1* contains 12 days of baseline traffic data (from May 1 through May 12, 2021), and is used for the evaluation of our first layer

(Section 5.1). For the evaluation of the second layer (Section 5.2), we build *labeled dataset 2* using the last 8 days of home user data (from May 13 through May 20, 2021). Our experiments show that the effect of attacks on downstream bps and pps rates is negligible (see Figures 4.6c and 4.6d). As the attacker's goal is typically to cause a denial of service at a target, it is not surprising that the upstream traffic of compromised nodes is more affected than their downstream traffic. Furthermore, the attacker can employ spoofed IP addresses during attacks, effectively nullifying response traffic received by the bots. Based on these observations, only upstream attack traffic is considered in the labeled dataset.

The procedure for building the *labeled datasets* is described next. The first step is to select the homes that will participate in the attacks. Let $\mathcal{H}$ denote the set of homes used in our study. From this set, we randomly choose $b = |\mathcal{B}|$ homes to "infect", where $\mathcal{B} \subset \mathcal{H}$ is the set of infected homes and $|\mathcal{B}|$ is the cardinality of set $\mathcal{B}$.

In the second step, the type of the attack (short or long), the attack vector, and the attack start time are defined. Let A be the total number of DDoS attacks that will be added to the baseline traffic dataset. For every attack i, $1 \leq i \leq A$, the type of the attack, d_i, is defined: i is defined as a short attack with probability p_s, and as a long attack with probability $(1 - p_s)$. The attack vector v_i used at the i-th attack is randomly chosen from the set of seven available DDoS attack vectors (see Table 4.1). The attack start time, a_i, is chosen from a uniform random variable $U(T_F, T_L)$, where T_F is the first minute of the baseline traffic dataset and T_L is the last minute of the dataset (Section 4.1.1). This step produces A attacks to be inserted in the baseline traffic dataset with attack i identified by the tuple (d_i, v_i, a_i), $d_i \in \{short, long\}$.

The final step is to select A attack samples from the attack dataset and add them to the baseline traffic dataset. For each attack i and infected home j, we randomly sample (with replacement) from the attack dataset an attack sample of type d_i and attack vector v_i to be inserted in the baseline traffic dataset at time a_i. Algorithm 1 summarizes the labeled dataset generation procedure.

Algorithm 1 Labeled DDoS dataset generation

```
Sample b infected homes from H
for i ← 1 to A do                                   ▷ Generate A attack instances
    if rand() < p_s then                            ▷ Probability for short attacks
        d_i ← "short"
    else
        d_i ← "long"
    end if
    v_i ← sample_vector()                           ▷ Random attack vector
    a_i ~ U(F_m, L_m)                               ▷ Uniform start time
end for
for all attack instances i ∈ [1, A] do
    for all infected homes j ∈ [1, b] do
        r_ij ← sample_attack(v_i, d_i)              ▷ Atk. trace w/ duration d_i, vector v_i
        insert_attack_traffic(j, r_ij, a_i)
    end for
end for
```

Figure 4.5 illustrates how the labeled datasets are built. In this example, $|\mathcal{H}| = 6, b = 3$ and $A = 2$. The home router traffic is represented by blue rectangles and the attack traffic is colored red. Home routers HR3, HR5 and HR6 were set as attack sources (infected homes) and attacks start at $a_1 = 1$ and $a_2 = 21$. The first attack is long, while the second is short.

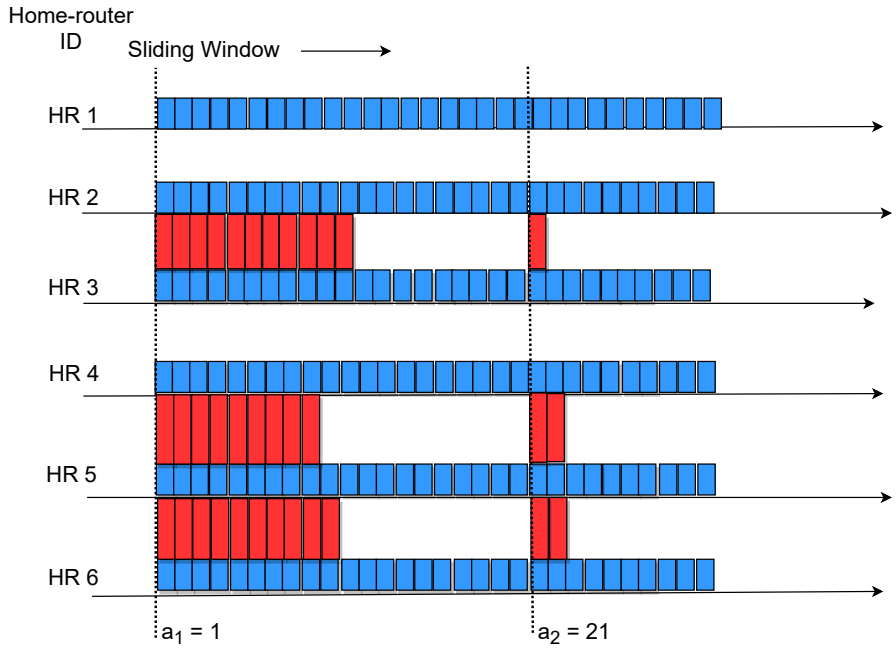


Figure 4.5: Dataset example. First attack is long and starts at $a_1 = 1$. Second attack is short and starts at $a_2 = 21$.

4.2 Volunteers dataset

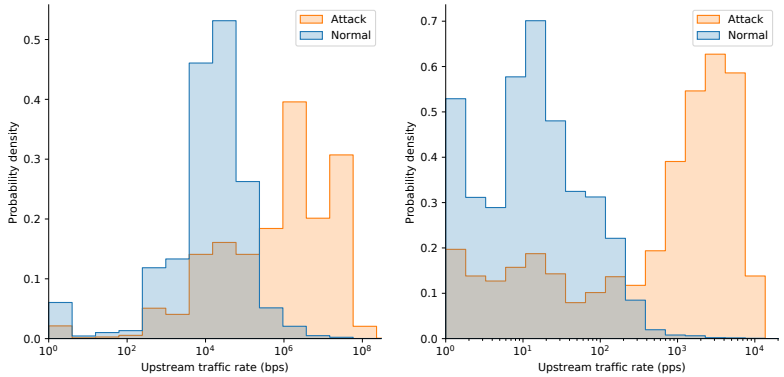
In the following, we describe an experiment to obtain a dataset that includes attacks from *devices connected to home routers in ordinary residences* rather than in a laboratory setup. We selected a group of ten volunteers. Each volunteer received an "infected" Raspberry Pi, running modified Mirai and BASHLITE source code. By removing certain software functions, we ensured that the modified malware could not infect other devices on the residential network.

All Raspberry Pi devices were programmed to launch DDoS attacks targeting a monitored server at random times. The intervals between attacks were generated according to an exponential distribution with a mean of 300 minutes (5 hours). To mitigate any significant impact on the user experience, intervals less than or equal to 5 minutes were excluded. During the experiment, volunteers were unaware of when each attack occurred, allowing them to maintain their daily routines without bias regarding the timing of the attacks. We varied the attack vectors during the experiment, using every attack vector and malware implementation listed in Table 4.1.

The volunteers were monitored for 31 days, from August 9 to September 9, 2021. We conducted a total of 111 DDoS attacks, averaging 3.6 attacks per day. Figure 4.6 shows the distribution of upstream and downstream traffic in the volunteers' home network during their daily activity and during attacks. Figures 4.6a and 4.6b show that upload traffic is usually higher during attacks. However, there is a reasonable overlap between the distributions, suggesting that simple threshold-based policies might generate a lot of false positives and/or false negatives. Furthermore, the distribution of the downstream traffic rates (Figures 4.6c and 4.6d) during attacks is almost indistinguishable from the distribution of legitimate traffic.

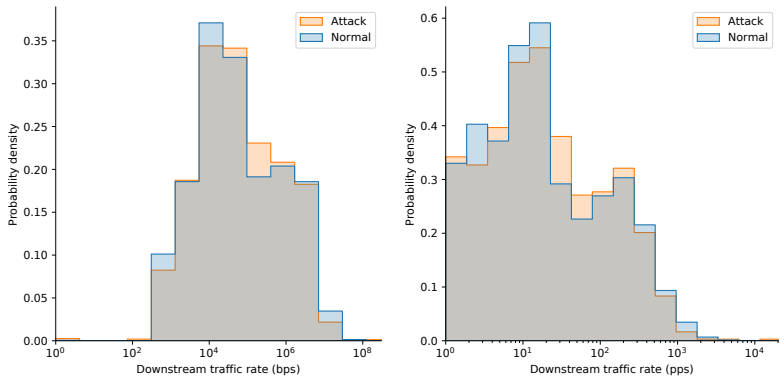
The volunteers' data was expanded by including traffic data from an additional 190 residences. A random sample of 190 residences was drawn from the set of 4,870 residences. The data from these 190 residences was collected from the same period as the volunteer experiment. Note that the scenario depicted by the dataset is that of two hundred residences, of which only a small fraction (5%) contains an infected device participating in sporadic DDoS attacks.

In Section 5.3, this dataset is used to evaluate the performance of our two-layer DDoS detection system. To our knowledge, no existing dataset in the literature shares the same characteristics as the one created in this study. Specifically, this dataset combines real traffic from home users with botnet-based DDoS attacks from a local device within the domestic network.



(a) Upstream bits per second rates.

(b) Upstream packets per second rates.



(c) Downstream bits per second rates.

(d) Downstream packets per second rates.

Figure 4.6: Density plots of upstream / downstream traffic rates at volunteers' homes comparing periods with and without attacks.

In Table 4.2, we compare the distribution of the traffic samples in each dataset using a few summary statistics. Due to the nature of the field experiment, we are unable to separate the normal and attack traffic in the *volunteers dataset*. We observe that the upload rates in the botnet lab experiment are significantly higher when compared to baseline home user traffic. However, when attack traffic is combined with normal traffic to create the *labeled dataset*, the traffic statistics remain nearly unchanged compared to those of the normal traffic. This is expected, as attacks are rare in the dataset, and the botnet represents only 5% of all residences. Furthermore, we observe that the distribution of traffic in the *labeled dataset* is similar to that of the *volunteers dataset*, suggesting that our results may generalize well between the two datasets.

Table 4.2: Summarized traffic statistics for each dataset.

Dataset	Statistic	Download rate		Upload rate	
		Bytes per sec. (kbps)	Packets per sec. (pps)	Bytes per sec. (kbps)	Packets per sec. (pps)
Baseline traffic (normal)	Mean	864	74	114	40
	Std.dev.	2463	213	1133	488
	25th pctl.	0.8	0.0	0.06	0.0
	Median	5.4	2.0	2.6	1.0
	75th pctl.	668	77	32	29
Botnet experiment (attack)	Mean	2.6	1.4	10876	3442
	Std.dev.	12	3.9	14788	2464
	25th pctl.	0.2	0.3	1238	1405
	Median	0.4	0.5	2470	3429
	75th pctl.	0.8	1.1	15822	4351
Labeled dataset (normal + attack)	Mean	864	74	176	62
	Std.dev.	2463	213	1888	716
	25th pctl.	0.8	0.0	0.1	0.0
	Median	5.4	2.0	2.6	1.0
	75th pctl.	668	77	32	29
Volunteers dataset (normal + attack)	Mean	902	98	182	60
	Std.dev.	5223	593	2672	444
	25th pctl.	9.8	4.0	5.7	3.1
	Median	37	14	17	13
	75th pctl.	481	79	48	36

Chapter 5

Results

Section 5.1 presents an analysis of the local DDoS detector running on home routers (first layer) and Section 5.2 evaluates the hierarchical Bayesian model (second layer) using the labeled dataset. Section 5.3 presents results for the volunteers dataset, the system's performance in a real environment.

Table 5.1 describes each dataset used for evaluating the system. The results of the first set of experiments were obtained using the *labeled datasets*. The second set of experiments used the *labeled datasets* for training and the *volunteers dataset* for testing.

Table 5.1: Datasets used to evaluate the system's performance.

	First Layer		Second Layer
	Training	Test	Test
First	*Labeled dataset 1*	*Labeled dataset 1*	*Labeled dataset 2*
Experiments	May 1 to 9, 2021	May 10 to 12, 2021	May 13 to 20, 2021
Second	*Labeled datasets 1 and 2*	*Volunteers dataset*	*Volunteers dataset*
Experiments	May 1 to 20, 2021	August 9 to September 9, 2021	August 9 to September 9, 2021

5.1 First layer — local detection

In this section, we present the results of the local detection. Section 5.1.1 describes the training and test sets and Section 5.1.2 compares F1 Scores across different machine learning (ML) models used to detect attacks from individual home routers. The best model from Section 5.1.2 is evaluated and interpreted in Section 5.1.3. We then evaluate the best model in various scenarios: (1) a "smart" attacker adjusts malware parameters to generate traffic at lower rates aiming to avoid attack detection (Section 5.1.4); (2) new, unforeseen malware, different from that used during training, is introduced (Section 5.1.5); (3) the training data contains mislabeled samples to account for uncertainty in the baseline traffic data (Section 5.1.6). Finally, in Section 5.1.7 we compare our classifier to simple threshold policies to verify whether or not the added complexity of ML models is indeed needed.

5.1.1 Training and test sets

For training and evaluating the local detection layer, we use the *labeled dataset 1* (first 12 days), leaving the *labeled dataset 2* (remaining 8 days) for the evaluation of the second detection layer (Section 5.2). The training and test datasets contain a mixture of the seven different attack vectors

from Mirai and BASHLITE (see Table 4.1). Attacks from both malware are used for training and testing purposes, with the exception of Section 5.1.5. We use 5-fold cross-validation for model selection with 75% of the data, corresponding to 9 out of 12 days considered. Once the best model is selected, the first 9 days of the dataset are used for training and the remaining 3 days for testing.

By the end of 2016, roughly 5% of Deutsche Telekom's customers were infected by Mirai [71]. A study from 2012 [72] estimates an even higher infection rate of approximately 20%. In addition, according to [6], the median of the interval between sequences of commands from a CnC server is approximately $2,600$ secs. This is approximately 32 attacks per day and 385 total attacks during a 12-day period. Consequently, we set the number of DDoS attacks to $A = 385$. Since, given evidence from [71], 5% of customers were infected and our 12-day training dataset contains data from $4,764$ homes, we set the number of infected homes in the *labeled dataset 1* to $b = 238$ (approximately 5% of the homes). Our results show that parameters A and b have little impact on the classifier's performance since all samples are treated independently. Motivated by the study in [70], the majority of attacks are short, and we consider that 95% of the attacks last an average of 2 minutes ($p_s = 0.95$).

5.1.2 Model selection

We evaluate five ML models: Logistic Regression, Decision Tree, Random Forest, Gaussian Naive Bayes, and Multilayer Perceptron. These models are chosen to favor simplicity, i.e., aiming to test if simple models can be used to achieve our objectives. The F1 score is the target performance metric for comparison, and we use a 5-fold cross-validation on the training set. Table 5.2 summarizes the results. The Random Forest model exhibits the best performance with an average F1 Score (among 5 folds) of 0.9729. Therefore, we use Random Forest to obtain the results in the sequel.

Table 5.2: F1 score using 5-fold cross validation.

Model	F1 score
Logistic Regression	0.8856
Decision Tree	0.9520
Gaussian Naive Bayes	0.7508
Random Forest	0.9729
Multilayer Perceptron	0.9417

5.1.3 Model evaluation

Table 5.3: Summary of Random Forest results.

Metric	Baseline Section 5.1.3	Smart attacker Section 5.1.4	New variant Section 5.1.5	Mislabeled Section 5.1.6
Accuracy	0.9995	0.9996	0.9996	0.9993
F1 Score	0.9715	0.9691	0.9725	0.9614
TPR	0.9524	0.9504	0.9531	0.9325
FPR	$7.5 \cdot 10^{-5}$	$7.5 \cdot 10^{-5}$	$5.6 \cdot 10^{-5}$	$6.6 \cdot 10^{-5}$
Avg. alarms / day	0.11 per home	0.11 per home	0.08 per home	0.10 per home

We evaluate the efficacy of the local and second-layer detectors using metrics such as True Positive Rate (TPR) and False Positive Rate (FPR), among others. However, due to the sliding

Table 5.4: True Positive Rate for each attack.

Malware	Attack vector	TPR
Mirai	UDP flood	0.9451
Mirai	TCP SYN flood	0.9570
Mirai	TCP ACK flood	0.9748
Mirai	UDP PLAIN flood	0.9304
BASHLITE	UDP flood	0.9422
BASHLITE	TCP SYN flood	0.9585
BASHLITE	TCP ACK flood	0.9665

window nature of the first-layer classifiers, some clarification is needed. For a given attack vector v, the TPR is the number of sliding windows that contain attack traffic of type v that were correctly classified (true positives) divided by the total number of sliding windows that contain attacks of that type. Consequently, the TPR measures how effective a local classifier is at identifying an attack based on the interval covered by each window. Each local classifier utilizes only the data samples contained within that specific window. Since an attack can span multiple windows, we use a metric called "Attack Detection Probability" (ADP) to measure the overall effectiveness of the classifier in detecting attacks. The definition of ADP will be provided later.

Table 5.3 shows the results of the Random Forest model used in the first layer employing the test set. As mentioned in Chapter 3, each local classifier provides a classification result every minute, utilizing a data window of size $\tau = 5$, with each window containing 12 features. The Random Forest model achieves an accuracy of 0.9995, and the True Positive Rate (TPR, or recall) is 0.9524. On average, we observe 0.11 false alarms per home per day, with a False Positive Rate (FPR) of $7.5 \cdot 10^{-5}$, indicating a very low misclassification rate. We list the ML metrics in Table 5.3 at column "Baseline".

To evaluate the model's performance with respect to different attack vectors, we compute the TPR for each attack vector v and show the results in Table 5.4. We remind the reader that our model was trained with a mix of attack vectors. Therefore, the results reported in Table 5.4 indicate that if the attacker uses only a single vector, our detection mechanism still accurately identifies the attacks. We also note that some attacks, like the ones that use TCP (with higher packets per second rates), are easier to detect than others.

Attack detection probability (ADP)

An attack can occur across multiple consecutive sliding windows, allowing the detection of attack traffic in one or more of those windows that capture at least part of the attack. Consider an attack of duration γ minutes. Traffic from this attack will be present in $\tau + \gamma - 1$ consecutive windows, where τ is the size of the sliding window in minutes. We consider an attack to be detected by the local classifier if the model produces a true positive result in at least one of the windows that contain the traffic of that attack. We define the *attack detection probability* (ADP) as the number of attacks detected in at least one sliding window divided by the total number of attacks

While TPR measures the effectiveness of the local classifier in detecting an attack at each time slot using data from the corresponding sliding window, ADP evaluates the overall likelihood of detecting an attack at least once during its entire duration. TPR evaluates slot-by-slot classification accuracy, whereas ADP focuses on whether an attack is recognized at any instant during its occurrence, regardless of when it is detected. In the case of this work, TPR is applied to local residences, and ADP is used to assess the local classifiers and the second-layer detector. For the test set, the ADP is 0.9951. For comparison, the classifier's TPR is 0.9524.

It should be observed that TPR is expected to increase with γ and, in addition, TPR is smaller than or equal to ADP. Let p_i be the attack detection probability at t_i with corresponding window w_i. For a given window size τ, let E_r be the expected number of attack slots per window covering the attack. It is simple to show that E_r increases with γ. The probability of detecting an attack in window w_i depends on how many attack slots are present in that window. A reasonable assumption for machine learning classifiers is that p_i is an increasing function of E_r, since the more attack slots per window, the higher the probability of detecting an attack in that window. TPR is the expected value of p_i across all windows, and then it follows that TPR also increases with γ.

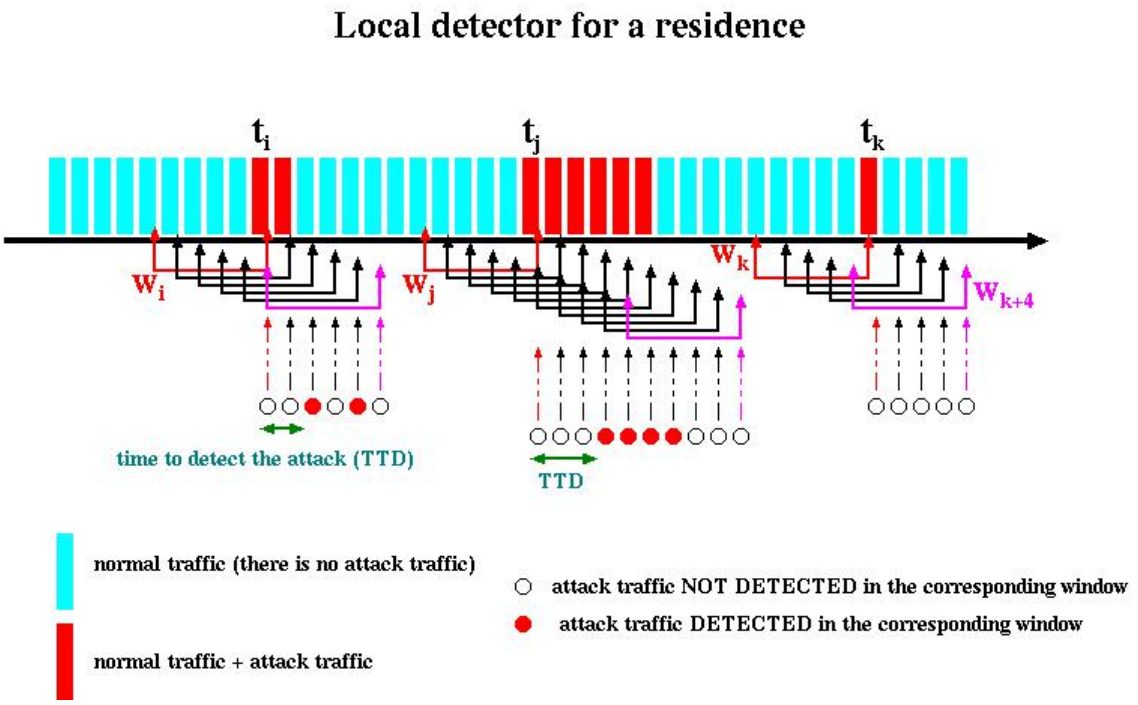


Figure 5.1: Illustration of the behavior of the classifier for a residence (local classifier). (Repeated from Figure 3.2 for convenience.)

Concerning ADP, an attack is considered *detected* if at least one of the windows that contain attack slots successfully detects it (see Figure 5.1). The probability that at least one of the $(\tau+\gamma-1)$ windows covering an attack starting at time a_i issues an attack warning is given by

$$\text{ADP}(\text{attack starting at } a_i) = 1 - \prod_{j=1}^{(\tau+\gamma-1)} (1 - p_{i+j-1}).$$

Then, clearly ADP $\geq$ TPR if $0 < p < 1$, that is, ADP benefits from multiple detection opportunities.

Next, we will examine whether the type of attack, either short or long, affects the ADP. We have a TPR of 0.9498 for short attacks, and for long attacks, we have a TPR of 0.9762. The attack detection probability for long attacks is also higher because there are more sliding windows encompassing one attack in a home. In particular, our classifier can detect all long attacks in our test set, i.e., the estimated ADP of long attacks is 1, while the attack detection probability of short attacks is 0.9949. Since long attacks are easier to detect, an intelligent attacker may use this information and prefer short attacks to increase the chances of the attack going undetected. However, short attacks may be less effective.

Time to detect (TTD)

The time to detect an attack is also an important metric to evaluate. Using the example of Figure 5.1, the time it took for the local classifier to detect the attack that started in t_i was two slots and three slots for the attack starting at t_j. As soon as the attack is detected, countermeasures can be taken at the local level, possibly isolating the local router if needed.

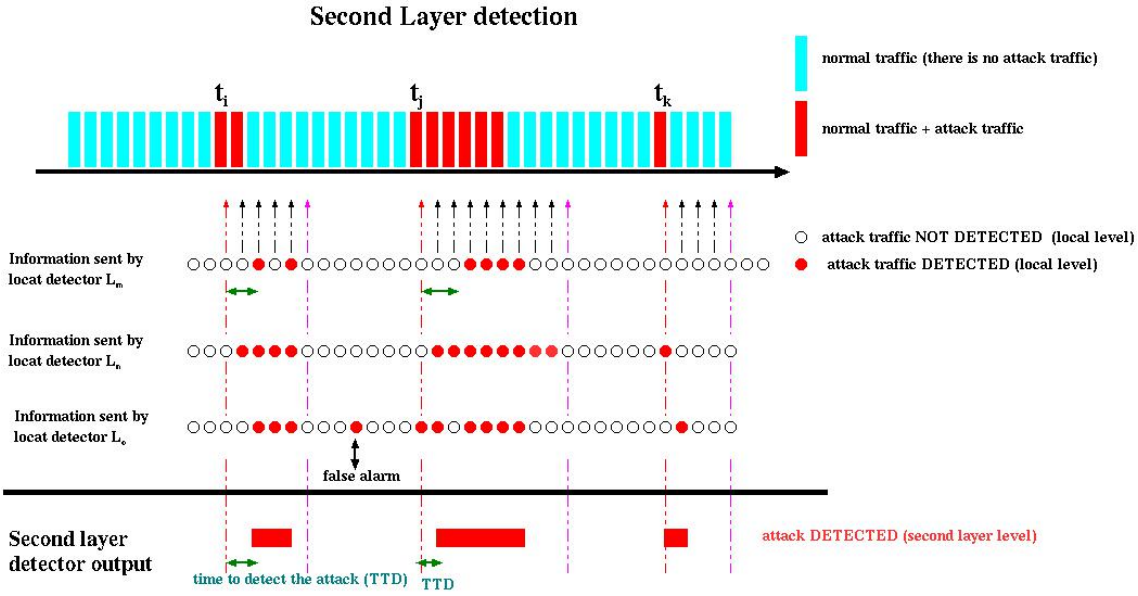


Figure 5.2: Illustration of the behavior of the second-layer detector. It receives one bit of information from local classifiers. (Repeated from Figure 3.3 for convenience.)

TTD can also be assessed at the second layer. Figure 5.2 illustrates that the second-layer detector may identify an attack more quickly or slowly than a specific local detector. Once the second layer detects a DDoS attack, further actions can be taken at the local level, as the confidence that a DDoS attack is indeed occurring improves.

In summary, short detection times are critical to adopt countermeasures and prevent subsequent attack attempts that may occur in short time intervals [6]. Among attacks detected by the classifier, 86.90% are detected within one minute, while 99.06% are detected within two minutes.

Feature evaluation

We use the importance metric based on the Gini index to evaluate the relative relevance of the twelve features used as inputs to the Random Forest model. Table 5.5 shows the results. The most important feature is the maximum of the *upstream packets per second (pps) rate* (Gini 0.298) followed by the difference between the maximum and minimum of the *upstream pps rate* (Gini 0.211). We note that the information collected from the download traffic is also important for detection. The fourth most important feature is the maximum ratio of upstream to downstream pps rates (Gini 0.111). Our data shows that legitimate home user traffic usually exhibits a high correlation between download and upload rates, differing from DDoS attacks. The top six features are derived from packets per second rates, but features based on bits per-second rates add useful information. We tested the classifier when only packet-rate-based features are used and compared the results with those obtained when both packet-rate and bitrate-related features are employed. In the first case, the F1 score is 0.9641, and in the second, it is 0.9715. The false positive rate is 10^{-4} with only packet-rate-based features and $7.5 \cdot 10^{-5}$ using all features.

Model interpretation. Random Forest models have limited interpretation capabilities beyond analyzing the importance of features. To circumvent its limitations, we use a simpler model – a Decision Tree (DT) – as a global surrogate [73, Chap.8]. A surrogate is a simple interpretable

Table 5.5: Feature importance of Random Forest model

Feature	Gini index
Up pps: Max	0.298
Up pps: Max - Min	0.211
Up pps: Std Dev	0.181
Up/down pps ratio: Max	0.111
Up/down pps ratio: Max - Min	0.075
Up/down pps ratio: Std Dev	0.036
Up bps: Max - Min	0.030
Up bps: Std Dev	0.030
Up bps: Max	0.023
Up/down bps ratio: Max - Min	0.003
Up/down bps ratio: Max	0.001
Up/down bps ratio: Std Dev	0.001

model used to reproduce the output of a more complex one. Decision Trees, for example, are often used to explain black box models due to their simplicity [74–78].

We build a global surrogate by training a DT with the training set but replace the true labels with the RF predictions. The maximum depth is limited to two to facilitate the interpretation. We believe the surrogate's performance (F1 score equal to 0.9518) is good enough to facilitate the interpretation of the RF's output but not to replace it in the classification step.

Figure 5.3 shows the resulting Decision Tree. The root node separates a large portion of the samples. For 99.8% of the sliding windows classified by the RF as *normal*, the standard deviation of the upstream pps rate is lower than or equal to 588.123 (left subtree). On the right-hand side, we see that most samples with higher upstream pps standard deviation are classified as *attack*. The second level of the tree shows that, after the standard deviation, the ratio between upstream and downstream traffic (either bytes or packets per second) will, in great part, determine whether a sliding window is classified as normal or not.

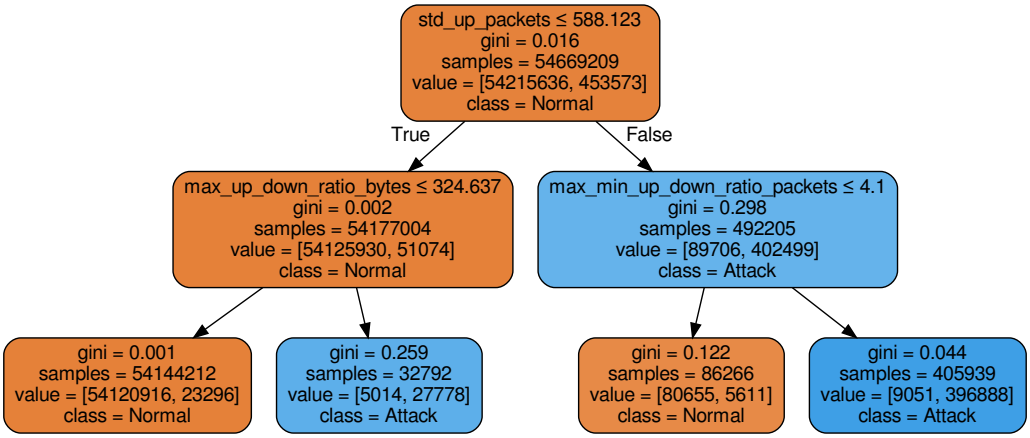


Figure 5.3: Global surrogate model: Decision Tree trained to reproduce the Random Forest's predictions.

In summary, there are three important features for classifying most windows: the standard

deviation of the upstream pps rate, the ratio between upstream and downstream traffic in bytes and the ratio between upstream and downstream traffic in packets per second. Windows classified as *attack* have higher values for the three features when compared to windows with normal traffic.

5.1.4 Smart attacker

Both Mirai and BASHLITE allow the attacker to set a custom payload size through commands issued by the CnC server. The sole exception is Mirai's TCP SYN flood attack, which is not configurable by default. The payload size parameter enables the attacker to adjust the attack's characteristics, trading off between a higher bits per second (bps) rate – by using larger payloads to maximize bandwidth consumption – and a higher packets per second (pps) rate – by using smaller payloads to overwhelm the target's packet-processing capacity. Therefore, a "smart" attacker could try to avoid detection by modifying this parameter.

We generate a new labeled dataset using the procedure described in Section 4.1 varying the packet payload sizes: 512B and 1400B for TCP SYN/ACK flood attacks; 0B and 512B for UDP flood attacks. We tested our model – which was trained with TCP attacks with empty payloads and UDP attacks with 1400B payloads – with this new dataset to assess its performance when detecting attacks with a different configuration than the one the model was trained with.

Table 5.6: TPR when different payload sizes are employed.

Malware	Attack	TPR
Mirai	UDP flood	0.9488
Mirai	TCP ACK flood	0.9653
Mirai	UDP PLAIN flood	0.9334
BASHLITE	UDP flood	0.9499
BASHLITE	TCP SYN flood	0.9598
BASHLITE	TCP ACK flood	0.9495

The results are close to those shown previously in Section 5.1.3, suggesting that the classifier is robust enough to detect attacks with different payload sizes than those used to train the model. There is a small decrease in F1 Score, from 0.9715 to 0.9691 (Table 5.3). Comparing Tables 5.6 and 5.4, we do not see a noticeable performance difference for each attack vector. An advantage of our methodology is that if the attack parameters change, we can easily generate a new dataset including attacks with the new parameters observed in the wild and retrain the classifier.

5.1.5 Detecting a new (unknown) attack: knowledge transfer

The Mirai malware, that was responsible for record-breaking attacks in 2016 and is still considered a threat [5, 9–11], evolved from BASHLITE [66]. Despite sharing some attack vectors (like TCP SYN/ACK flood and UDP flood), the source code from both malware is significantly different. To analyze the performance of our classifier when facing a new malware variant, we train our model solely with BASHLITE attacks and try to detect Mirai attacks on the test set.

By comparing the results with those obtained when training with all attack vectors (Section 5.1.3), we observe similar results for FPR ($5.6 \cdot 10^{-5}$) and TPR (0.9531). When considering attack vectors individually (Table 5.7), it is interesting to note the high true positive rate for UDP PLAIN flood attacks. Although not implemented by BASHLITE, these attacks seem to have a signature closer to BASHLITE's attack vectors. It turns out that the simplicity of the

proposed classifier is key not only for performance purposes but also to achieve the desired level of generalization.

Table 5.7: TPR for Mirai's attacks using BASHLITE model.

Attack	TPR
UDP flood	0.9474
TCP SYN flood	0.9434
TCP ACK flood	0.9775
UDP PLAIN flood	0.9440

5.1.6 Effect of mislabeled training data

The ISP did not detect any DDoS attacks during the data collection period. However, we cannot assure that our data does not contain attacks. Samples that were considered legitimate user traffic might actually contain attack traffic. In the following, we evaluate the impact of mislabeled data on the training set.

We randomly choose 5% of the attacks in the training set and change the labels from "attack traffic" to "normal traffic" on all corresponding sliding windows for all infected homes. Then, we train a Random Forest classifier with the new dataset and evaluate it using the test set with correct labels. While the FPR remains low ($6.6 \cdot 10^{-5}$), we notice a slight decrease in the classifier's TPR (from 0.9524 to 0.9325). Nonetheless, even when we add mislabeled samples to the training set, the classifier's ADP remains high (0.9943), with 97.9% of all attacks detected in up to two minutes.

5.1.7 Evaluating threshold policies

This section aims to evaluate the performance of a simple threshold policy in the first-level detector compared to the performance of the Random Forest model we trained. By "simple threshold policy", we refer to a simple strategy that employs a threshold ϕ on the upload throughput packet rate to determine whether attack traffic is present on a slot. Then, for each slot, a traffic rate above ϕ is classified by the local (first-level) detector as traffic that includes an attack.

We selected various threshold values ϕ based on the percentiles of the upstream packet rate during attacks and calculated the resulting F1 Score for each. The highest F1 Score obtained was F1 = 0.7478 (taking into account the class imbalance rate), with a corresponding TPR = 0.7276 and FPR = 9.3×10^{-4}. This was achieved when $\phi = \phi_o = 1,844$pps when ϕ_o corresponds to the 25th percentile of the traffic rate during attacks. These values are considerably worse than those obtained using the Random Forest classifier (F1 Score of 0.9715, TPR = 0.9524 and FPR = 7.5×10^{-5}).

Increasing the threshold from ϕ_o is expected to improve (reduce) the false positive rate (FPR) while worsening (reducing) the true positive rate (TPR). However, the exact magnitude of this change is unknown. For example, if $\phi = 3,895$ pps, corresponding to the median of the traffic rate during attacks, then FPR = 1.8×10^{-4}, which is approximately 19% of its value when $\phi = \phi_o$. However, this comes at the expense of the TPR. The new value is approximately 62% of that when $\phi = \phi_o$ (TPR = 0.4541), which is significantly low, implying F1 Score = 0.6066.

Overall, the Random Forest model consistently outperforms the single-threshold policy at any threshold, making it a better choice for the first-layer (local) detector, achieving both high detection rates and low false alarm rates.

Table 5.8: True Positive Rate (TPR) for each attack and scenario.

Malware	Attack vector	Baseline (Section 5.1.3)	Smart attacker (Section 5.1.4)	New variant (Section 5.1.5)	Mislabeled (Section 5.1.6)
Mirai	UDP flood	0.9451	0.9488	0.9474	0.9332
	TCP SYN flood	0.9570	-	0.9434	0.9425
	TCP ACK flood	0.9748	0.9653	0.9775	0.9661
	UDP PLAIN flood	0.9304	0.9334	0.9440	0.9058
BASHLITE	UDP flood	0.9422	0.9499	-	0.9371
	TCP SYN flood	0.9585	0.9598	-	0.9416
	TCP ACK flood	0.9665	0.9495	-	0.9074

Table 5.8 aggregates the results for all scenarios and attack vectors that were considered in Section 5.1. No (attack vector, scenario) combination led to a TPR lower than 0.90, showing the robustness of our approach for local detection.

5.2 Second layer — spatio-temporal correlation

In the previous section, we showed that 99.5% of all attacks in the *labeled dataset 1* were detected at the first layer (ML classifier) with a very low number of false positives. Fortunately, attacks "missed" by the first layer can be detected when we combine the outputs from multiple homes and add a second detection layer. In the following, we show how results can be improved by using our Bayesian model.

5.2.1 Dataset

As described in Section 5.1.1, we use the *labeled dataset 1* (first 12 days) to train and test the Machine Learning classifier. The *labeled dataset 2* (last 8 days) is left for the evaluation of the spatio-temporal correlation layer. We opted to be conservative when evaluating the second layer, decreasing the fraction of infected homes from 5% (*labeled dataset 1*) to 1% (*labeled dataset 2*). The rationale is that if the second layer has good performance when the number of infected devices is lower than expected, it should perform even better when a DDoS attack mobilizes a greater number of devices.

Both labeled datasets are built with data collected from more than $4,000$ homes configured to send measurements to a central server at one-minute intervals (Section 4.1). However, due to issues like loss of Internet connection, power failures, and periods of server maintenance, the number of home routers that send a measurement to the server varies over time. In certain minutes of the collection period, the number of measurements from distinct homes is much less than $4,000$. To address this missing data issue, we added a simple pre-processing step, leaving out of our evaluation the minutes when: (1) fewer than $4,000$ home routers sent a measurement; or (2) less than 50% of infected homes sent a measurement. In this process, we discard approximately 13% of measurements in the period. Section 5.3 shows that we can still obtain good results when the number of homes is less than $4,000$.

After filtering the *labeled dataset 2*, we apply the previously trained classifier. Define x_i as the total number of homes that reported positive (i.e., the classifier reported an attack) for the i-th 5-minute sliding window. Figure 5.4 shows a scatter plot comparing the real number of homes participating in an attack (b) and the total number of positive results (x_i) for each sliding window. When no DDoS attack is occurring (blue dots), we observe up to 6 homes reporting a false alarm

in the same sliding window ($x_i \leq 6$ and $b = 0$). During DDoS attacks (orange dots), the number of attacking homes b varies between 23 and 47. When $x_i < b$, the first layer failed to detect some of the attackers. In the case where $x_i > b$, there are false and true positives in the window i. With a perfect classifier on the first layer, all orange dots would be on the green line, i.e., $x_i = b$ for all windows with attacks [1].

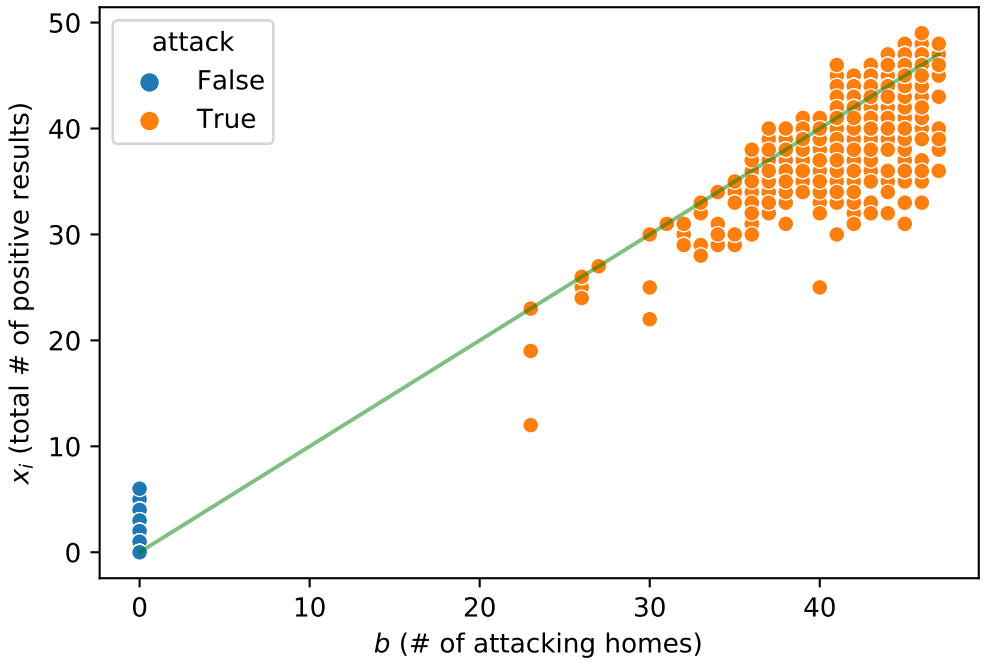


Figure 5.4: Scatter plot comparing the actual number of attacking homes b at each sliding window to the observed number of positive results x_i during periods with (orange) and without DDoS attacks (blue).

5.2.2 Choice of hyperparameters

The hyperparameters $\alpha_b, \beta_b, \alpha_F, \beta_F, \alpha_D$ and β_D in our Bayesian model determine the shape of the probability distribution of the latent variables θ_b, θ_F and θ_D before observing a sample x_i (notation in Table 3.2). A non-informative uniform prior is far from what we would expect in a real setting. First, based on the results for the *labeled dataset 1*, we can expect that the actual false positive probability θ_F will be close to 0, while the true positive probability θ_D will be closer to 1. Second, based on previous reports such as [71, 72], a probability of infection θ_b close to 1 (indicating that nearly all homes are infected) is very unlikely.

Therefore, we opt for the use of informative priors following the guidelines in [63, 79]. All priors begin with a vague Beta$(1, 1)$ "proto-prior" that is updated by a small set of representative dataset samples. The hyperparameters of the Beta prior distribution were calculated from the mode and concentration of the Beta distribution [63]. Based on the performance of the classifier (Section 5.1.3), we set the prior modes of θ_b, θ_F and θ_D to be equal to, respectively, 0.05, $7.5 \cdot 10^{-5}$

[1] We could also have $x_i = b$ for all attacks if the classifier happened to miss some bots and always raise an equivalent number of false alarms.

and 0.95. For the Poisson and binomial models, we set $\hat{\theta}_F = 7.5 \cdot 10^{-5}$ and $\hat{\theta}_D = 0.95$. The concentration of the Beta prior distribution – defined as the sum of the two shape parameters α and β – quantifies the uncertainty around its mode and can be interpreted as the equivalent number of prior samples in a Beta-binomial model. The concentration parameters (κ_b, κ_D, κ_F) are determined by selecting approximately 0.01% of the total number of samples in the training set. Specifically, κ_b and κ_D are set equal to the number of samples labeled as "attack", while κ_F is set equal to the number of samples labeled "normal". We obtained the following values for the concentration of θ_b, θ_F and θ_D, respectively, $\kappa_b = 52$, $\kappa_F = 5002$ and $\kappa_D = 52$. Consequently, we have $\alpha_b = 3.5$, $\beta_b = 48.5$, $\alpha_F = 1.375$, $\beta_F = 5000.625$ and $\alpha_D = 48.5$, $\beta_D = 3.5$. We emphasize that the mode of the prior distribution selected for parameter θ_b is five times larger than the actual fraction of infected homes in *labeled dataset 2*. This indicates that the model will, *a priori*, anticipate a higher number of infected homes.

5.2.3 Attack detection probability (ADP)

As discussed before, attack traffic will span over multiple sliding windows. A DDoS attack is successfully detected if there is *strong evidence* for at least one window in favor of H_1 (the observed data indicates a DDoS attack). Figure 5.5 shows the values of the variables b and x_i during a long BASHLITE UDP Flood attack in the *labeled dataset 2*. The number of attackers b vary over time due to the missing data. The number of homes raising an alarm for each window, x_i, is usually very close to b thanks to the good performance of the first layer. Right after the attack starts, we have $x_i = 34$, $b = 44$ and, according to the beta-binomial model, $\text{BF}_{10} = 6.5 \cdot 10^7$. This evidence is more than enough to detect this attack within a minute. One minute later, we have $x_i = b = 45$ simultaneous alarms and, thus, a stronger evidence in favor of the hypothesis of an attack with $\text{BF}_{10} = 2.4 \cdot 10^{11}$.

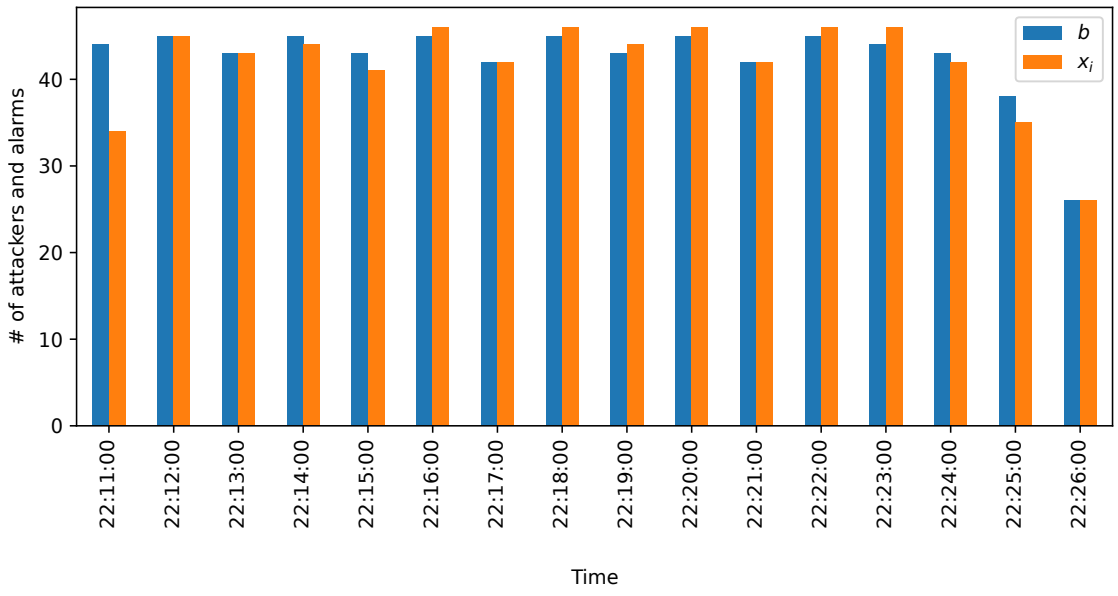


Figure 5.5: Number of attackers b (blue) and number of simultaneous alarms x_i (orange) during a long BASHLITE UDP Flood attack sample.

Over the eight days period, the three attack models (Poisson, binomial and beta-binomial) successfully detect all attacks in the dataset with the given hyperparameters. The models return *decisive evidence* in favor of H_1 ($\text{BF}_{10} \geq 100$ [1]) for all windows in which infected homes were

attacking. **All attacks in the dataset were detected in up to one minute**. This result suggests that our Bayesian model is robust, detecting attacks even when the real number of attackers in the *labeled dataset 2* (47 infected homes) is lower than the expected value of b given θ_b's prior (269 infected homes [2]).

5.2.4 False alarms

Our results show that the second layer improves the attack detection probability (ADP) and the time to detect an attack (TTD) while maintaining a low incidence of false alarms. Table 5.9 presents how samples without attack traffic are classified by each model according to the Bayes Factor. The vast majority of the samples are correctly categorized as "normal". In most cases, there is *decisive evidence* in favor of H_0, i.e., $\text{BF}_{10} < 1/100$. Only one "normal" sample is categorized as an attack by the Poisson model with strong evidence and by the binomial model with substantial evidence. When an alarm is triggered based on substantial evidence of an attack ($\text{BF}_{10} \geq 3$), the beta-binomial model produces no false alarms, while the Poisson and binomial models result in only one false alarm each.

Table 5.9: Categorization of "normal" samples in the *labeled dataset 2*. Evidence categories based on [1].

Hypothesis	Evidence	Bayes Factor Range	Poisson	Binomial	Beta-binomial
H_0	Decisive	$\text{BF}_{10} < 1/100$	7317	7322	7340
	Very strong	$1/100 < \text{BF}_{10} \leq 1/30$	20	15	0
	Strong	$1/30 < \text{BF}_{10} \leq 1/10$	0	0	0
	Substantial	$1/10 < \text{BF}_{10} \leq 1/3$	0	2	0
	Anecdotal	$1/3 < \text{BF}_{10} \leq 1$	2	0	0
H_1	Anecdotal	$1 \leq \text{BF}_{10} < 3$	0	0	0
	Substantial	$3 \leq \text{BF}_{10} < 10$	0	1	0
	Strong	$10 \leq \text{BF}_{10} < 30$	1	0	0
	Very strong	$30 \leq \text{BF}_{10} < 100$	0	0	0
	Decisive	$\text{BF}_{10} > 100$	0	0	0

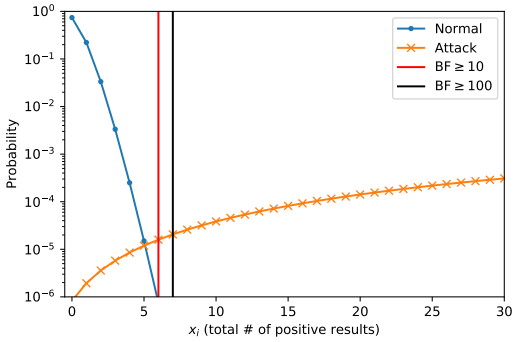
Adding a second DDoS detection layer is beneficial, increasing the ADP while decreasing the TTD. Moreover, the results show that the Bayesian model (and its variations) is robust when the prior distributions are informed by a small representative set (as recommended in [79]). The model detects all attacks, even when the number of infected homes, b, is approximately 17% of the expected value a priori.

5.2.5 Model evidence

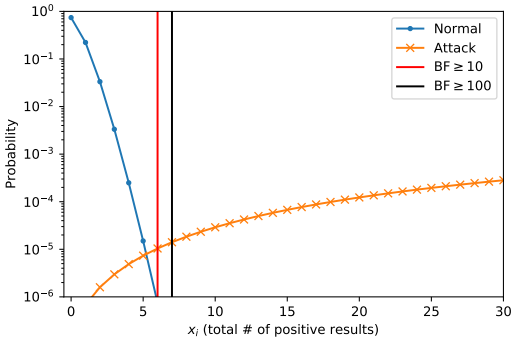
To better understand the second layer's performance, we now briefly discuss some numerical results.

Figure 5.6 presents the probability mass function of x_i under two hypotheses: (1) H_0 (no attack), represented in blue; and (2) H_1 (attack), represented in orange. The hyperparameters are set as described in Section 5.2.2, and each plot represents a different variation of the Bayesian model. The plots indicate a clear separation between the two distributions, where the probability of observing higher values of x_i is significantly greater under attack scenarios than during normal traffic conditions. The red and black lines in Figure 5.6 indicate critical Bayesian factor thresholds: (1) The red line represents $\text{BF}_{10} \geq 10$, indicating strong evidence in favor of H_1; and (2) The black line represents $\text{BF}_{10} > 100$, indicating decisive evidence for H_1.

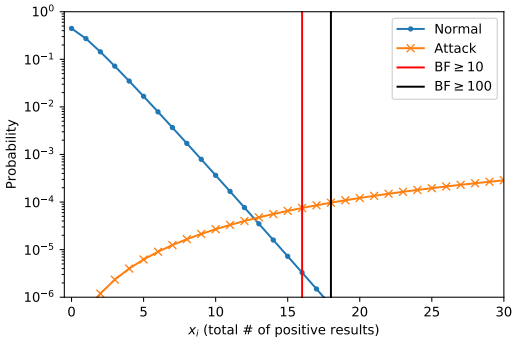
[2] For $N = 4000$, $\alpha_b = 3.5$ and $\beta_b = 48.5$, we have $\mathbf{E}[b] = N\frac{\alpha_b}{\alpha_b+\beta_b} \approx 269$.



(a) Poisson model.



(b) Binomial model.



(c) Beta-binomial model.

Figure 5.6: Model evidence (PMF of x_i) under hypotheses of attack (orange) and no attack (blue) according to each model.

In particular, the Poisson and binomial models require a small number of alarms to reach strong ($BF_{10} \geq 10$) and decisive ($BF_{10} > 100$) evidence thresholds. Furthermore, the transition from strong to decisive evidence occurs within a small range of x_i, meaning that only a few additional positive results are required for the models to significantly increase their confidence in attack detection.

Compared to Poisson and binomial models, the Beta-binomial model shows more conservative detection behavior, as indicated by the higher number of alarms required to reach strong and decisive evidence thresholds. This is because, in the Beta-binomial model, θ_F is treated as a latent variable with a prior distribution rather than a fixed parameter. Using a prior introduces additional uncertainty, requiring more observations to accumulate sufficient evidence. However, as discussed in Section 3.2.4, the prior distribution of θ_F can be updated using real data. This adaptive approach allows the model to refine its estimates over time, potentially reducing the number of alarms needed to reach strong or decisive evidence.

Additionally, the probability of false alarms under normal conditions is nearly zero for all models, indicating robustness against misclassification in benign scenarios.

5.3 DDoS attacks on volunteer residences

We now assess the performance of our system using the *volunteers dataset* described in Section 4.2. We start by training the first layer's Random Forest model using the *labeled datasets 1* and *2*, from May 1 through May 20, totaling ≈ 116 million (homes $\times$ sliding windows) samples.

We apply the machine learning classifier (corresponding to the first layer) to the 31-day volunteers dataset, which contains traffic data from 200 homes, 10 of which have an infected device. The Attack Detection Probability (ADP) for the first detection layer is ≈ 0.97 (vs. ≈ 0.995 on the *labeled dataset 1*). Among the successfully detected attacks, 73% were detected in up to 2 minutes. The Random Forest's false positive rate was approximately ten times greater than that observed in the *labeled dataset 1*: $6.8 \cdot 10^{-4}$ but is still very low.

Similar to previous findings, the results show improvement when incorporating the second layer. To handle missing data, we follow the same approach as described in Section 5.2.1. We discard sliding windows where either (1) fewer than 175 home routers submitted their classification reports or (2) less than 50% of the infected homes submitted a report. Approximately 3% of the reported data were discarded in this process. Hyperparameters are set as in Section 5.2.2, with priors' modes chosen according to the classifier's performance. Since the number of homes N in the volunteers dataset is approximately 20 times less than in the labeled datasets, we scale the concentration of the beta priors accordingly by dividing it by 20. Therefore, we have $\alpha_b = 1.125$, $\beta_b = 3.375$, $\alpha_F = 1.019$, $\beta_F = 250.981$, $\alpha_D = 3.375$ and $\beta_D = 1.125$ for the *volunteers dataset*.

We analyze the results when only "normal" samples (i.e., those without an attack) are considered in Table 5.10. The beta-binomial model does not raise any false alarms (that is, there are no "normal" samples for which $BF_{10} > 1$. In contrast, the Poisson and binomial models produce 288 false alarms with Bayes Factor in the range of *decisive evidence* in favor of H_1 ($BF_{10} > 100$). This results in a false positive rate (FPR) of $7.1 \cdot 10^{-3}$ and an average of 10.2 false alarms per day across a total of 200 homes.

Table 5.11 presents an analysis of all samples in the *volunteers dataset*. An attack is considered detected if there is at least *substantial evidence* supporting H_1 (i.e., $BF_{10} > 3$) in at least one sliding window for each attack. The average detection probability (ADP) for the beta-binomial model is 0.991, while it is 1 for both the Poisson and binomial models. The beta-binomial model successfully

Table 5.10: Categorization of "normal" samples in the *volunteers dataset*. Evidence categories based on [1].

Hypothesis	Evidence	Bayes Factor Range	Poisson	Binomial	Beta-binomial
H_0	Decisive	$BF_{10} < 1/100$	0	0	0
	Very strong	$1/100 < BF_{10} \leq 1/30$	35819	35819	35819
	Strong	$1/30 < BF_{10} \leq 1/10$	0	0	4629
	Substantial	$1/10 < BF_{10} \leq 1/3$	0	0	276
	Anecdotal	$1/3 < BF_{10} \leq 1$	4521	4353	12
H_1	Anecdotal	$1 \leq BF_{10} < 3$	108	276	0
	Substantial	$3 \leq BF_{10} < 10$	0	0	0
	Strong	$10 \leq BF_{10} < 30$	0	0	0
	Very strong	$30 \leq BF_{10} < 100$	0	0	0
	Decisive	$BF_{10} > 100$	288	288	0

detects 73.0% of all attacks within 2 minutes, with a maximum time-to-detection (TTD) of 5 minutes. Additionally, the beta-binomial model generated a total of 516 alarms, compared to 1,018 alarms raised by both the Poisson and binomial models. The Poisson and binomial models detected all attacks within two minutes at the expense of a high number of false alarms.

Table 5.11: Summary of results for the *volunteers dataset*.

Metric	Poisson		Binomial		Beta-binomial		
	Inform. Prior	Uniform Prior (θ_b)	Inform. Prior	Uniform Prior (θ_b)	Inform. Prior	Uniform Prior (θ_b)	Uniform Prior ($\theta_b, \theta_F, \theta_D$)
ADP	1	1	1	1	0.991	0.927	0
TTD $\leq$ 2 min	100%	100%	100%	100%	73.0%	58.2%	0%
FPR	$7.1 \cdot 10^{-3}$	$7.1 \cdot 10^{-3}$	$7.1 \cdot 10^{-3}$	$7.1 \cdot 10^{-3}$	0	0	0
Avg. false alarms / day	10.2	10.2	10.2	10.2	0	0	0

The high number of false alarms generated by the Poisson and binomial models can be attributed to the fixed values of θ_F and θ_D which are constant parameters determined based on the results obtained for the *labeled dataset 1* (refer to Figure 3.5). In the case of the beta-binomial model, θ_F and θ_D are latent variables and, therefore, have a prior distribution instead of a constant value. By setting $\hat{\theta}_F$ and $\hat{\theta}_D$ to the actual (unknown) FPR and TPR values for the *volunteers dataset*, the number of false alarms with $BF_{10} > 3$ (indicating at least *substantial evidence* in favor of H_1) decreases significantly from 288 to 12.

Next, we evaluate the impact of adopting non-informative uniform priors on our results. Uniform Beta$(1, 1)$ priors are commonly used to reflect a lack of *a priori* information about the probability parameter θ_b. This means that all values of θ_b between 0 and 1 are equally likely before observing any data.

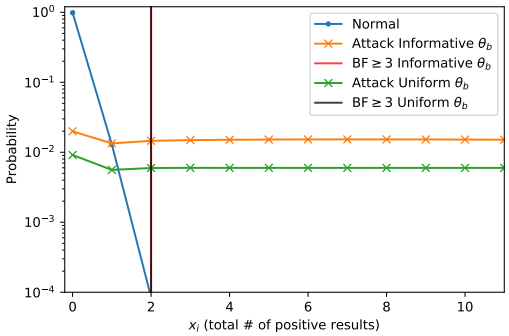
Similarly, when using a uniform θ_F prior, the classifier's false positive rate (FPR) can either be 0 (never raising a false alarm) or 1 (raising a false alarm whenever possible), with equal probability. Although this is an unreasonable assumption, we evaluate the impact of uniform priors in our dataset for the sake of completeness.

In Figure 5.7, we plot the model evidence for each model under both informative and uniform priors. Figures 5.7a and 5.7b illustrate that for the Poisson and binomial models, using a uniform prior for the θ_b parameter ($\alpha_b = \beta_b = 1$) does not alter the threshold value of x_i at which the Bayes Factor BF_{10} reaches 3. Consequently, key detection performance metrics such as ADP, TTD, and number of false alarms remain unchanged.

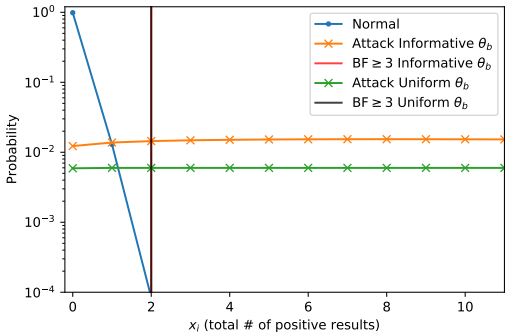
However, when a uniform θ_b prior is applied in the beta-binomial model while maintaining

informative priors for θ_F and θ_D (Figure 5.7c), increases the threshold for substantial evidence in favor of H_1 from $x_i = 5$ to $x_i = 7$. This adjustment makes the model more conservative, meaning it detects fewer attacks (lower ADP) and takes longer to identify them (longer TTD). Despite this stricter detection threshold, the number of false alarms remains at zero, suggesting that the choice of θ_b does not influence false alarm rates under these conditions.

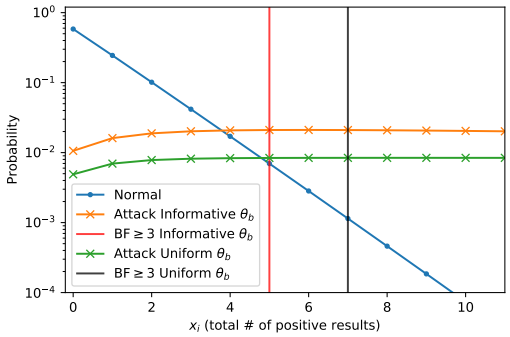
Finally, if uniform priors are assigned to θ_b, θ_F, and θ_D in the beta-binomial model (results omitted from Figure 5.7), no attack can be detected, as no value of x_i satisfies $\mathrm{BF}_{10} \geq 3$. A summary of all results is provided in Table 5.11.



(a) Poisson model.



(b) Binomial model.



(c) Beta-binomial model.

Figure 5.7: Model evidence (PMF of x_i) under hypotheses of no attack (blue) and attack (informative priors in orange and uniform priors in green) given $N = 175$.

When using the *volunteers dataset*, our two-layer DDoS detection system achieved the following results: (1) it detected 97% of all attacks near their origin, at the home routers; (2) it identified 99.1% if attacks using the beta-binomial model and 100% of the attacks using the binomial and Poisson models when accounting for spatio-temporal correlations; (3) it provided a reasonably short detection time for attacks, typically within 2 minutes; (4) the beta-binomial model raised no false alarms. Furthermore, the beta-binomial model demonstrated robust performance even when its hyperparameters were adjusted based on different data (*labeled dataset 1*). To the best of our knowledge, no previous work has employed such a comprehensive dataset to evaluate DDoS detection.

Chapter 6

Conclusion

As mentioned in Chapter 1, several studies have shown that IoT devices are being infected on a large scale due to insufficient protection mechanisms (e.g., weak authentication, outdated firmware, and misconfigurations) [11, 12]. Moreover, Mirai-based attacks remain prevalent, as reported by major cybersecurity companies. Studies such as [80] emphasize that Internet Service Providers (ISPs) play a crucial role in mitigating DDoS attacks, as most infected devices operate within their networks. These facts motivated our work to develop efficient detection mechanisms that can be implemented in residential routers. Additionally, our approach can be used to identify residences within an ISP's network that are participating in an attack, enabling the ISP to assist customers in enhancing the security of their local devices.

Most previously proposed attack detection methods rely on sensitive packet header information, such as source and destination IP addresses, protocols, and ports, to detect attacks. Furthermore, a notable shortage of DDoS datasets containing authentic traffic data exists. This scarcity motivated us to collect real residential traffic and design experiments to evaluate the feasibility of our proposed approach.

We propose a two-layer, lightweight, privacy-preserving system for DDoS detection that relies solely on byte and packet count data, metrics that are easily obtained even from inexpensive home routers. The first detection layer is implemented on these routers and uses a simple pre-trained ML classifier trained on real residential traffic collected from thousands of homes. The second layer can be implemented in a simple server and utilizes a Bayesian hierarchical model to correlate detection results across different residences using the ISP's services.

We partnered with an ISP to gather upload and download byte and packet counts from traffic flowing through residential routers in 4,870 homes across 14 cities. Over 20 days, these metrics, collected from off-the-shelf home routers, were used to construct a device-agnostic model. Attack vectors were generated using real source codes from Mirai and BASHLITE. (The generated *labeled datasets* have been made public and are available upon request.)

We evaluated the performance of our approach using two datasets. The first, called the *labeled dataset*, was constructed by combining real residential traffic from the partner ISP, representing baseline regular activity, with DDoS attack traffic generated in a controlled laboratory environment using authentic malware source code. The second dataset, the *volunteers dataset*, includes traffic data (both regular and attack) collected from ten volunteered residences over 30 days. The volunteers hosted an infected Raspberry Pi device running authentic malware source code and were unaware of when the device would initiate an attack. Additionally, the *volunteers dataset* was expanded with traffic data from an additional 190 users from the partner ISP.

We show that the first detection layer was effective, while the second layer, which employs a Bayesian model, significantly enhanced detection performance. We also evaluated the time needed to detect an attack, showing that our approach reacts quickly. Another contribution of our work is the insight gained regarding how the routers' ML model classified samples as malicious or legitimate and identified the most important features for this classification.

In summary, our results indicate that simple statistics derived from byte and packet counts collected every minute are effective for achieving a high attack detection rate with a very low false positive rate. Our results also point towards a simple way to identify residences with infected devices within an ISP's network.

We published our current results in [61][81][82][83]. Also, other authors used our labeled dataset in their work [84][48][85][86].

References

[1] WETZELS, R., MATZKE, D., LEE, M. D., et al. "Statistical evidence in experimental psychology: An empirical comparison using 855 t tests", *Perspectives on Psychological Science*, v. 6, n. 3, pp. 291–298, 2011.

[2] DARKREADING. "DDoS Attacks Spiked, Became More Complex in 2020". 2020. Available at: <https://www.darkreading.com/attacks-breaches/ddos-attacks-spiked-became-more-complex-in-2020/d/d-id/1339814>. Accessed: 2021-07-02.

[3] TIDY, J. "Ukraine says it is fighting first 'hybrid war'". https://www.bbc.com/news/technology-60622977, 2022. Accessed: 2022-03-24.

[4] PAUL, K., MILMO, D. "Russia-backed hackers behind powerful new malware, UK and US say". https://www.theguardian.com/world/2022/feb/23/russia-hacking-malware-cyberattack-virus-ukraine, 2022. Accessed: 2022-03-24.

[5] CLOUDFLARE. "DDoS threat report for 2024 Q2". 2024. Available at: <https://radar.cloudflare.com/reports/ddos-2024-q2>. Accessed: 2024-11-6.

[6] MARZANO, A., ALEXANDER, D., FONSECA, O., et al. "The Evolution of Bashlite and Mirai IoT Botnets". In: *IEEE ISCC 2018*, 2018. doi: 10.1109/ISCC.2018.8538636.

[7] AKAMAI. *Q3 2016 State of the Internet - Security Report*. Technical report, Akamai, nov 2016.

[8] YOACHIMIK, O. "Cloudflare thwarts 17.2M rps DDoS attack — the largest ever reported". 2021. Available at: <https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported/>. Accessed: 2021-08-20.

[9] KASPERSKY. "IT threat evolution Q1 2021". 2021. Available at: <https://securelist.com/it-threat-evolution-q1-2021-non-mobile-statistics/102425/>. Accessed: 2021-07-02.

[10] NETSCOUT. "2H 2020 Threat Intelligence Report – DDoS in a Time of Pandemic". 2021. Available at: <https://www.netscout.com/threatreport/>. Accessed: 2021-07-02.

[11] AKAMAI. "Uncovering HinataBot: A Deep Dive into a Go-Based Threat". 2023. Available at: <https://www.akamai.com/blog/security-research/hinatabot-uncovering-new-golang-ddos-botnet>. Accessed: 2023-06-19.

[12] CLOUDFLARE. "DDoS Attack Trends for 2024 Q1". 2024. Available at: <https://radar.cloudflare.com/reports/ddos-2024-q1>. Accessed: 2024-11-6.

[13] KASPERSKY. “DDoS attacks in Q1 2019”. 2019. Available at: <https://securelist.com/ddos-report-q1-2019/90792/>. Accessed: 2021-07-02.

[14] NETLAB 360. “Mirai_ptea Botnet is Exploiting Undisclosed KGUARD DVR Vulnerability”. 2021. Available at: <https://blog.netlab.360.com/mirai_ptea-botnet-is-exploiting-undisclosed-kguard-dvr-vulnerability-en/>. Accessed: 2021-07-02.

[15] PALO ALTO NETWORKS. “New Mirai Variant Targets Zyxel Network-Attached Storage Devices”. 2020. Available at: <https://unit42.paloaltonetworks.com/new-mirai-variant-mukashi/>. Accessed: 2021-07-02.

[16] AKAMAI. *Satori Mirai Variant Alert*. Technical report, Akamai, Dec 2017.

[17] NETLAB 360. “New Threat: Matryosh Botnet Is Spreading”. 2021. Available at: <https://blog.netlab.360.com/matryosh-botnet-is-spreading-en/>. Accessed: 2021-07-02.

[18] KASPERSKY. “Mirai goes Enterprise”. 2019. Available at: <https://www.kaspersky.com/blog/mirai-enterprise/26032/>. Accessed: 2021-07-02.

[19] TUXCARE. “Mirai malware targets Linux servers and IoT devices”. 2023. Available at: <https://tuxcare.com/blog/miral-malware-targets-linux-servers-and-iot-devices/>. Accessed: 2024-03-07.

[20] HASAN, M. “State of IoT 2023: Number of connected IoT devices growing 16% to 16.7 billion globally”. 2023. Available at: <https://iot-analytics.com/number-connected-iot-devices/>. Accessed: 2023-11-29.

[21] RING, M., WUNDERLICH, S., SCHEURING, D., et al. “A Survey of Network-based Intrusion Detection Data Sets”, *Computers & Security*, 2019.

[22] ALMARAZ-RIVERA, J. G., PEREZ-DIAZ, J. A., CANTORAL-CEBALLOS, J. A., et al. “Toward the Protection of IoT Networks: Introducing the LATAM-DDoS-IoT Dataset”, *IEEE Access*, v. 10, pp. 106909–106920, 2022. doi: 10.1109/ACCESS.2022.3211513.

[23] “The CAIDA UCSD "DDoS Attack 2007" Dataset”. Accessed: 2021-08-16. Available at: <https://www.caida.org/catalog/datasets/ddos-20070804_dataset>.

[24] “NSL-KDD dataset”. Accessed: 2021-08-16. Available at: <https://www.unb.ca/cic/datasets/nsl.html>.

[25] MOUSTAFA, N., SLAY, J. “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)”. In: *2015 military communications and information systems conference (MilCIS)*, pp. 1–6. IEEE, 2015.

[26] SHARAFALDIN, I., LASHKARI, A. H., HAKAK, S., et al. “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy”. In: *2019 International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019. doi: 10.1109/CCST.2019.8888419.

[27] KORONIOTIS, N., MOUSTAFA, N., SITNIKOVA, E., et al. “Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset”, *Future Generation Computer Systems*, v. 100, pp. 779–796, 2019.

[28] MEIDAN, Y., BOHADANA, M., MATHOV, Y., et al. "N-BaIoT–Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders", *IEEE Pervasive Computing*, v. 17, n. 3, pp. 12–22, 2018. doi: 10.1109/MPRV.2018.03367731.

[29] MCDERMOTT, C. D., MAJDANI, F., PETROVSKI, A. V. "Botnet Detection in the Internet of Things using Deep Learning Approaches". In: *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2018. doi: 10.1109/IJCNN.2018.8489489.

[30] GARCIA, S., PARMISANO, A., ERQUIAGA, M. J. "IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set]". 2020. Available at: <`https://www.stratosphereips.org/datasets-iot23`>. Accessed: 2021-11-18.

[31] BASTIAN, N., BIERBRAUER, D., MCKENZIE, M., et al. "ACI IoT Network Traffic Dataset 2023". 2023. Available at: <`https://dx.doi.org/10.21227/qacj-3x32`>.

[32] SASI, T., LASHKARI, A. H., LU, R., et al. "An Efficient Self Attention-Based 1D-CNN-LSTM Network for IoT Attack Detection and Identification Using Network Traffic", *Journal of Information and Intelligence*, 2024. ISSN: 2949-7159. doi: https://doi.org/10.1016/j.jiixd.2024.09.001. Available at: <`https://www.sciencedirect.com/science/article/pii/S2949715924000763`>.

[33] SILVEIRA, F., DIOT, C., TAFT, N., et al. "ASTUTE: Detecting a different class of traffic anomalies", *ACM SIGCOMM CCR*, v. 41, n. 4, pp. 267–278, 2011.

[34] LIASKOS, C., KOTRONIS, V., DIMITROPOULOS, X. "A novel framework for modeling and mitigating distributed link flooding attacks". In: *INFOCOM 2016*, pp. 1–9. IEEE, 2016.

[35] MARÍN, G., CASAS, P., CAPDEHOURAT, G. "DeepMAL-deep learning models for malware traffic detection and classification". In: *Data Science – Analytics and Applications*, pp. 105–112. Springer Fachmedien Wiesbaden, 2021. doi: 10.1007/978-3-658-32182-6_16.

[36] NEVAT, I., DIVAKARAN, D. M., NAGARAJAN, S. G., et al. "Anomaly Detection and Attribution in Networks With Temporally Correlated Traffic", *IEEE/ACM Transactions on Networking*, v. 26, n. 1, pp. 131–144, 2018.

[37] PENA, E. H., CARVALHO, L. F., BARBON JR, S., et al. "Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment", *Information Sciences*, v. 420, pp. 313–328, 2017.

[38] WICHTLHUBER, M., STREHLE, E., KOPP, D., et al. "IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale". In: *Proceedings of the ACM SIGCOMM 2022 Conference*, p. 707–722, 2022. doi: 10.1145/3544216.3544268.

[39] DOSHI, R., APTHORPE, N., FEAMSTER, N. "Machine Learning DDoS Detection for Consumer Internet of Things Devices". In: *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 29–35, 2018. doi: 10.1109/SPW.2018.00013.

[40] BAHŞI, H., NÕMM, S., LA TORRE, F. B. "Dimensionality reduction for machine learning based IoT botnet detection". In: *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1857–1862. IEEE, 2018.

[41] NÕMM, S., BAHŞI, H. "Unsupervised anomaly based botnet detection in IoT networks". In: *2018 17th IEEE international conference on machine learning and applications (ICMLA)*, pp. 1048–1053. IEEE, 2018.

[42] WAN, Y., XU, K., XUE, G., et al. "IoTArgos: A Multi-Layer Security Monitoring System for Internet-of-Things in Smart Homes". In: *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pp. 874–883. IEEE, 2020.

[43] ANTHI, E., WILLIAMS, L., SŁOWIŃSKA, M., et al. "A supervised intrusion detection system for smart home IoT devices", *IEEE Internet of Things Journal*, v. 6, n. 5, pp. 9042–9053, 2019.

[44] SALMAN, O., ELHAJJ, I. H., CHEHAB, A., et al. "A machine learning based framework for IoT device identification and abnormal traffic detection", *Transactions on Emerging Telecommunications Technologies*, p. e3743, 2019.

[45] WANG, A., CHANG, W., CHEN, S., et al. "Delving into internet DDoS attacks by botnets: characterization and analysis", *IEEE/ACM Transactions on Networking*, v. 26, n. 6, pp. 2843–2855, 2018.

[46] JIA, Y., ZHONG, F., ALRAWAIS, A., et al. "FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks", *IEEE Internet of Things Journal*, v. 7, n. 10, pp. 9552–9562, 2020.

[47] SUDHEERA, K. L. K., DIVAKARAN, D. M., SINGH, R. P., et al. "ADEPT: Detection and Identification of Correlated Attack Stages in IoT Networks", *IEEE Internet of Things Journal*, v. 8, n. 8, pp. 6591–6607, 2021.

[48] STREIT, A., SANTOS, G. H., LEÃO, R. M., et al. "Network anomaly detection based on tensor decomposition", *Computer Networks*, v. 200, pp. 108503, 2021.

[49] FELDMANN, A., GASSER, O., LICHTBLAU, F., et al. "A Year in Lockdown: How the Waves of COVID-19 Impact Internet Traffic", *Commun. ACM*, v. 64, n. 7, pp. 101–108, Jun. 2021. ISSN: 0001-0782. doi: 10.1145/3465212. Available at: <`https://doi.org/10.1145/3465212`>.

[50] NAJAFIMEHR, M., ZARIFZADEH, S., MOSTAFAVI, S. "DDoS attacks and machine-learning-based detection methods: A survey and taxonomy", *Engineering Reports*, v. n/a, n. n/a, pp. e12697, 2023. doi: https://doi.org/10.1002/eng2.12697.

[51] ALI, I., AHMED, A. I. A., ALMOGREN, A., et al. "Systematic Literature Review on IoT-Based Botnet Attack", *IEEE Access*, v. 8, pp. 212220–212232, 2020. doi: 10.1109/ACCESS.2020.3039985.

[52] WANG, W., ZHU, M., ZENG, X., et al. "Malware traffic classification using convolutional neural network for representation learning". In: *2017 International Conference on Information Networking (ICOIN)*, pp. 712–717. IEEE, 2017.

[53] "MAWI Working Group Traffic Archive". 2014. Available at: <`http://mawi.wide.ad.jp/mawi`>. Accessed: 2022-01-22.

[54] DE ASSIS, M. V., JR., M. L. P. "Scorpius: sFlow Network Anomaly Simulator", *Journal of Computer Science*, v. 11, n. 4, pp. 662–674, Jul. 2015. doi: 10.3844/jcssp.2015.662.674. Available at: <`https://thescipub.com/abstract/jcssp.2015.662.674`>.

[55] SEDJELMACI, H., SENOUCI, S. M., TALEB, T. "An accurate security game for low-resource IoT devices", *IEEE Transactions on Vehicular Technology*, v. 66, n. 10, pp. 9381–9393, 2017.

[56] SUMMERVILLE, D. H., ZACH, K. M., CHEN, Y. "Ultra-lightweight deep packet anomaly detection for Internet of Things devices". In: *Performance Computing and Communications Conference*, pp. 1–8. IEEE, 2015.

[57] MEHDI, S. A., KHALID, J., KHAYAM, S. A. "Revisiting traffic anomaly detection using software defined networking". In: *International workshop on recent advances in intrusion detection*, pp. 161–180. Springer, 2011.

[58] WANG, B., LI, X., DE AGUIAR, L. P., et al. "Characterizing and Modeling Patching Practices of Industrial Control Systems", *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, v. 1, n. 1, pp. 18:1–18:23, 2017.

[59] SUNDARESAN, S., DE DONATO, W., FEAMSTER, N., et al. "Broadband Internet Performance: A View from the Gateway". In: *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, p. 134–145, New York, NY, USA, 2011. Association for Computing Machinery. ISBN: 9781450307970. doi: 10.1145/2018436.2018452. Available at: <https://doi.org/10.1145/2018436.2018452>.

[60] MENDES, D. X., SENGES, G. D. S., SANTOS, G. H. A. D., et al. "A Preliminary Performance Measurement Study of Residential Broadband Services in Brazil". In: *Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks*, LANCOMM '16, p. 16–18, New York, NY, USA, 2016. Association for Computing Machinery. ISBN: 9781450344265. doi: 10.1145/2940116.2940135. Available at: <https://doi.org/10.1145/2940116.2940135>.

[61] MENDONÇA, G., SANTOS, G. H., E SILVA, E. D. S., et al. "An Extremely Lightweight Approach for DDoS Detection at Home Gateways". In: *2019 IEEE International Conference on Big Data (Big Data)*, pp. 5012–5021. IEEE, 2019.

[62] GELMAN, A., CARLIN, J. B., STERN, H. S., et al. *Bayesian Data Analysis, Third Edition*. United Kingdom, Chapman and Hall/CRC, 2013.

[63] KRUSCHKE, J. K. *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan (2nd)*. San Diego, CA, Academic Press, 2015.

[64] BENNEYAN, J. C., BORGMAN, A. D. "A useful J-binomial type distribution for non-homogeneous dichotomous events". In: *IIE Annual Conference. Proceedings*, p. 1. Institute of Industrial and Systems Engineers (IISE), 2004.

[65] LIU, B., QUERTERMOUS, T. "Approximating the Sum of Independent Non-Identical Binomial Random Variables", *The R Journal*, v. 10, n. 1, pp. 472–483, 2018. doi: 10.32614/RJ-2018-011. Available at: <https://doi.org/10.32614/RJ-2018-011>.

[66] ANTONAKAKIS, M., APRIL, T., BAILEY, M., et al. "Understanding the Mirai botnet". In: *USENIX Security Symposium*, pp. 1092–1110, 2017.

[67] BREIMAN, L. "Random forests", *Machine learning*, v. 45, n. 1, pp. 5–32, 2001.

[68] MORAWIEC, D. "sklearn-porter", Transpile trained scikit-learn estimators to C, Java, JavaScript and others, Accessed: 2022-03-24. Available at: <https://github.com/nok/sklearn-porter>.

[69] MEIDAN, Y., BENATAR, D., BITTON, R., et al. "D-Score: An expert-based method for assessing the detectability of IoT-related cyber-attacks", *Computers & Security*, v. 126, pp. 103073, 2023. ISSN: 0167-4048. doi: https://doi.org/10.1016/j.cose.2022.

103073. Available at: <https://www.sciencedirect.com/science/article/pii/S0167404822004655>.

[70] BLENN, N., GHIËTTE, V., DOERR, C. "Quantifying the Spectrum of Denial-of-Service Attacks through Internet Backscatter". In: *Conference on Availability, Reliability and Security*, p. 21. ACM, 2017.

[71] AUCHARD, E. "German Internet outage was failed botnet attempt: report". 2016. Available at: <https://www.reuters.com/article/us-deutsche-telekom-outages-idUSKBN13N12K>. Accessed: 2021-07-02.

[72] OLLMANN, G. "Household Botnet Infections". 2012. Available at: <https://circleid.com/posts/20120326_household_botnet_infections>. Accessed: 2025-05-16.

[73] MOLNAR, C. *Interpretable machine learning*. Morrisville, NC, Lulu.com, 2020.

[74] CRAVEN, M., SHAVLIK, J. "Extracting tree-structured representations of trained networks", *Advances in neural information processing systems*, v. 8, pp. 24–30, 1995.

[75] BLANCO-JUSTICIA, A., DOMINGO-FERRER, J. "Machine learning explainability through comprehensible decision trees". In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pp. 15–26. Springer, 2019.

[76] POYIADZI, R., RENARD, X., LAUGEL, T., et al. "Understanding surrogate explanations: the interplay between complexity, fidelity and coverage", *arXiv preprint arXiv:2107.04309*, 2021.

[77] RENARD, X., WOLOSZKO, N., AIGRAIN, J., et al. "Concept tree: High-level representation of variables for more interpretable surrogate decision trees", *arXiv preprint arXiv:1906.01297*, 2019.

[78] JIA, S., LIN, P., LI, Z., et al. "Visualizing surrogate decision trees of convolutional neural networks", *Journal of Visualization*, v. 23, n. 1, pp. 141–156, 2020.

[79] KRUSCHKE, J. K. "Bayesian analysis reporting guidelines", *Nature human behaviour*, v. 5, n. 10, pp. 1282–1291, 2021.

[80] CETIN, O., GANAN, C. H., ALTENA, L., et al. "Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai". In: *26th Annual Network and Distributed System Security Symposium, NDSS 2019*, San Diego, United States, 2019. Internet Society.

[81] MENDONÇA, G., SANTOS, G. H., DE SOUZA, E., et al. "Uma abordagem leve para detecção de DDoS a partir de roteadores domésticos". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 834–847. SBC, 2019.

[82] MENDONÇA, G., SANTOS, G. H., DE SOUZA, E., et al. "Detecção de ataques DDoS usando correlação espaço-temporal bayesiana". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 224–237. SBC, 2022.

[83] MENDONÇA, G., LEÃO, R. M., EDMUNDO DE SOUZA, E. S., et al. "Lightweight DDoS attack detection using Bayesian space-time correlation", *IEEE Access*, 2025.

[84] STREIT, A. G., SANTOS, G. H., LEAO, R. M., et al. "Identificaçao de Anomalias em Redes de Dados baseada em Decomposiçao Tensorial". In: *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)*, pp. 952–965. SBC, 2020.

[85] RAMTIN, A. R., NAIN, P., MENASCHE, D. S., et al. “Fundamental scaling laws of covert DDoS attacks”, *Performance Evaluation*, v. 151, pp. 102236, 2021.

[86] RAMTIN, A., TOWSLEY, D., NAIN, P., et al. “Are covert DDoS attacks facing multi-feature detectors feasible?” *ACM SIGMETRICS Performance Evaluation Review*, v. 49, n. 2, pp. 33–35, 2022.