Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

The Attacker IP Prioritizer: An IoT Optimized Blacklisting Algorithm

Bachelor thesis

Thomas O'Hara, B.L.A

Study program: Electrical Engineering and Computer Science
Field of study: Computer Science
Supervisor: Ing. Sebastián García, Ph.D.

Prague, May 2021

BACHELOR'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **O'Hara Thomas** Personal ID number: **480003**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Electrical Power Engineering**

Study program: **Electrical Engineering and Computer Science**

II. Bachelor's thesis details

Bachelor's thesis title in English:

The Attacker IP Prioritizer : An IoT Optimized Blacklisting Algorithm

Bachelor's thesis title in Czech:

The Attacker IP Prioritizer : An IoT Optimized Blacklisting Algorithm

Guidelines:

IP address-based blacklists are an integral part of most firewall, IDS and security systems for any kind of Internet-connected device. Even modern Threat Intelligence feeds are based on IP addresses, domains and URLs. Therefore,the majority of our protection systems, such as in DNS servers and webbrowsers depend on blacklists. However, there has not been yet a goodevaluation about how effective these blacklists are, or how they can beoptimized for different environments.
With the ever constant growth of 5G and the bypassing of traditional firewalls with direct Internet connections, it is becoming more and more difficultto protect IoT devices using traditional blacklisting methods. Many blacklists inthe community are created by adding the IP addresses of attackers into ageneral feed, with the IP addresses usually coming from the data collected fromone or many honeypots. This idea is assumed to work well, but it has two maindrawbacks for IoT environments. First, although systems with greater storageand large computational resources may afford to store and parse an evergrowing blacklist, small Internet of Things (IoT) devices have limited computational resources and may not hold large blacklists in memory. Second,IP addresses attacking today can be associated with normal services in thefuture, especially in cloud environments. Moreover, the nature of IoT malware shows that attacking IP addresses mostly attack for a short amount of time (afew hours or days), questioning the value of blocking IP addresses for extendedperiods without verification.
In this Thesis, I will propose an algorithm to optimize the creation ofblacklists and an evaluation method in order to help understand their issues.First, I will design a new algorithm for creating blacklists that is optimized for theprotection of IoT devices, called the Attacker IP Prioritizer (AIP). Second, I willpresent an idea for a standardized methodology for evaluating the efficacy of blacklists.
The blacklist algorithm will be designed to optimize for certainperformance metrics common in IoT scenarios. AIP will create a routinelyupdated scoring system trained on network captures gathered from real IoThoneypot networks. For each source IP, AIP extracts a set of performancemetrics, among which are:
- Total number of connections
- Average number of connections per day
- Total number of bytes transferred
- Average bytes per connection
- Total number of packets transferred
- Average number of packets per connection
- Total connection times
- Average connection times per connection

These performance metrics will then be used to create models that usethe metrics to generate a score for each malicious IP. This sore will then beused to decide if that IP should be blocked or not.
One such model I propose will be designed to assign higher scores toIPs that are consistent in their attacks for a long time. In this model, IPs thatattack a meaningful amount of times every day will be assigned higher scores,thus making this blacklist more likely to contain IPs from devices that are higherin botnet hierarchies, such as victim bots, compromised computers.
Another model will be designed to assign higher scores to newer IPsthat attack a lot, thus prioritizing intense short term attackers. IPs in this blacklistwill be more likely to be end-user infections spreading to other IoT, as well andinfected web servers and fast changing cloud deployments. It achieves this byassigning greater importance to high total metric counts, and ages each IPbased simply on how long it has been since that IP was first added to thehistorical dataset.
The evaluation methodology will consist of training each blacklist withdata from the past and evaluating how accurate the protection will be. Thetraining and evaluation will be done in an iterative way, using each successiveday to update the blacklists, and each 'tomorrow' date to evaluate them.

Bibliography / sources:

1.Sûnnet Beskerming. 2007. Time to blacklist blacklists.http://www.beskerming.com/commentary/2007/07/01/196/Time_to_Blacklist_Blacklist
2.Baris Coskun. 2017. (Un) wisdom of Crowds: Accurately SpottingMalicious IPClusters Using Not-So-Accurate IP Blacklists.IEEETransactions on Information Forensics and Security 12, 6 (2017)https://ieeexplore.ieee.org/abstract/document/7839928/
3.Sivaramakrishnan Ramanthan, Jelena Mirkovic, and Minlan Yu. 2018.BlacklistsAssemble: Aggregating Blacklists for Accuracy. TechnicalReport. Technical ReportISI-TR-730. Information Sciences Institute.https://steel.isi.edu/members/sivaram/papers/blag_technical_report.pdf
4.Ramanathan, S., Mirkovic, J., & Yu, M. (2020, January). BLAG:Improving the Accuracy of Blacklists. In 27th Annual Network andDistributed System Security Symposium, NDSS (pp. 23-26).
5.Michael Bailey Sushant Sinha and Farnam Jahanian. 2008.Shades ofgrey: On the effectiveness of reputation-based "blacklists". 2008 3rdInternational Conference On Malicious and Unwanted Software(MALWARE). IEEE,https://ieeexplore.ieee.org/abstract/document/4690858

Name and workplace of bachelor's thesis supervisor:

Ing. Sebastián García, Ph.D., Artificial Intelligence Center, FEE

Name and workplace of second bachelor's thesis supervisor or consultant:

Date of bachelor's thesis assignment: **25.01.2021** Deadline for bachelor thesis submission: **21.05.2021**

Assignment valid until: **30.09.2022**

Ing. Sebastián García, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the bachelor's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Declaration

I hereby declare I have written this bachelor thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2021

.....
Thomas O'Hara, B.L.A

Abstract

IP address-based blacklists are the most important part of firewalls, security systems and Threat Intelligence feeds. However, there is no comprehensive and verified evaluation of blacklists to determine if they are effective, how efficient they are, or how they forget data. Given the reliance in Threat Intelligence feeds, it is critical for blacklists to be optimally generated and evaluated.

With the constant growth of 5G and IPv6 technologies, IoT devices have two unique problems: direct connection to the Internet and resources constrains. Therefore, traditional long blacklists may not fit in IoT devices, and may take time to process. Moreover, IP addresses attacking today may be associated with benign services later, marking a need to update blacklists.

This thesis proposes an algorithm to optimize the creation of blacklists as well as an evaluation method targeted at better quantifying a blacklists efficacy over time. Our Attacker IP Prioritizer, or AIP, framework is designed to optimize for certain performance metrics common in IoT scenarios.

The AIP framework includes three models that generate prioritized blacklists using a threat score for each malicious IP. This score is then used to decide if that IP should be blocked or not. Two of these models were designed to produce a score by combining features with a time-based aging function that decreases or increase that score. A third model uses a Machine Learning (ML) model that predicts if each particular IP is going to attack in the future.

The evaluation methodology of AIP consists of building each blacklist with data from the past and evaluating how accurate the protection is in the future. The training and evaluation were done in an iterative way, using each successive day to update the blacklists, and each 'tomorrow' date to evaluate them. The performance metrics used were percentage of bytes blocked, total duration of attacks blocked in the future, percentage of flows blocked in the future, and total IP addresses blocked in the future.

We compared AIP against four major blacklists that are provided for free as threat intelligence feeds in the Internet. For this we created an Index Metric that computes the effectiveness of each IP address in a blacklists to protect from future threats. The best AIP model achieved an Index metric of 0.0068%, which is 22x times better than the rest of the threat intelligence feeds on the Internet. We conclude that the AIP models and the evaluation methodology can help improve the protection of memory-constrained devices by maximizing the impact of blacklists.

Keywords: IoT, Honeypots, Blacklists, Threat Intelligence, Random Forest, Machine Learning, Intrusion Prevention

Acknowledgements

I would first like to thank my thesis advisor Ing. Sebastián García, Ph.D. from the FEE, Czech Technical University. I would like to thank AvastLab and the AIC group of the CVUT University in Prague for their support. Also I would like to thank Veronica Valeros for the use of her Hornet 15 dataset. Special thanks to María José Erquiaga, Maria Rigaki, Ondřej Lukáš, and Elnaz Babayeva for their support and reviews.

Thomas O'Hara

List of Tables

List of Figures

Contents

Chapter 1

Introduction

Blacklists have always been an essential part of security, being used in Antivirus, Intrusion Detection Systems (IDS), and threat intelligence feeds. They are easy to generate, have some level of accuracy, and are easy to implement. IoT devices, in particular, would benefit from them as a defense mechanism given the amount and diversity of attacks they receive and given the future direct exposure to the Internet given by 5G networks. Despite the known constraints of IoT devices, there are no methodologies that we know of for efficiently creating such blacklists. There are also no standard methodologies for evaluating and comparing which blacklists are more effective and perform better. Especially in the IoT world, devices not only have become a primary target for attackers that turn them into malicious bots [30], but they are also often resource-constrained, not providing enough memory and processing capacity to protect themselves [4, 5]. As far as we know, there are no comparison methodologies for blacklists, even for traditional computers and systems.

IoT blacklists are needed because they may be the first line of defense between the attacks that occur every day in these devices. There are free blacklists currently available, and many are sold as part of the threat intelligence business, but there is no comprehensive analysis of how good they are, how many errors they make, and how many resources they use. This may imply that blacklists may not work as well as we believe [2]. Some methods were proposed to improve their performance, mainly making use of aggregation of blacklists [20] and IP clustering techniques [6], while others focus more on better-utilizing traffic data [23]. However, these methods do not compare blacklists and do not focus on improving the performance while reducing the size of the blacklist.

This thesis presents an alternative framework for generating blacklists of IP addresses, called Attacker IP Prioritizer (AIP). AIP consists of three models that ingest network data and output blacklists. Two of the three models are based on a linear combination of normalized data used to create a score for each IP. The final list of IPs in the blacklist depends on this score. These models aim at maximizing the amount of malicious traffic blocked while minimizing the size of the blacklist. In this way, the blacklists are easily deployable in devices with small memory, such as smart-doorbells [1]. The third model implemented in the AIP framework is a Random Forest machine learning model. This method uses a concatenated database of labeled past data to train the model and then predicts which IPs seen in the past are going to be seen in the next time window of operation and should be blocked. These predicted IPs are used to create a blacklist.

In order to create our blacklists and evaluate our methods, we created a specific dataset of real IoT attacks. This dataset was created by capturing traffic in the Aposemat IoT laboratory [10], using real IoT devices for five months, namely December 2020 until April

2021. These devices were exclusively used as honeypots, and the traffic was filtered such that every connection to them can be considered an attack.

The evaluation of our models was done by comparing them to 4 open-source commonly used blacklists. This evaluation followed a specific methodology. First, a time frame was decided for the comparison, from December 2020 to April 2021, and each day we downloaded the third-party blacklists. Notice that this means that the comparison was in real-time since there is no way to download third-party blacklists that have already been updated with new data. Each day we used their latest version. Also, notice that we do not know the internal process each blacklist used to update. Second, the AIP models were updated using different criteria for each of these days, and blacklists were generated using these models. Third, each day we used the blacklists from the AIP framework and the downloaded feeds to *block* those IPs in the next day and calculated how many attacks were blocked by each blacklist. A special blacklist that is a memory of all the IPs that attacked in the previous days was used as a baseline.

According to the evaluation metrics, our comparison showed that the AIP framework performed as well or better than the downloaded open-source threat intelligence feeds. The linear models performed as well as other blacklists comparable in size, blocking on average 37% of incoming malicious IP addresses while being several orders of magnitude smaller. The Random Forest models performed much better than blacklists comparable in size, blocking on average up to 22% of the incoming malicious IP addresses. We calculated a performance index for each blacklist to compare ones differing in size. This index was what percent of the malicious traffic could be blocked by a single IP on average per blacklist. We found that the Random Forest blacklist could block 0.0068% of malicious attacking IPs per IP in its list. This made it 13x better than the AIP linear models at 0.0005% and 22x times better than IPsum, which was the best performing downloaded blacklist, at 0.0003%. The Random Forest model proved to be the best model for generating blacklists for IoT devices.

The contributions of this thesis are:

- The AIP framework , which consists of three models. One that prioritizes consistent attackers, one that prioritizes new attackers, and one that predicts future attacks using all features.
- The inclusion of an aging method inside our models.
- An IoT dataset of honeypots attacks.
- An online, public and free feed of our three blacklists for the community to download.
- A methodology for comparing and evaluating blacklist.
- A comparison between mainstream blacklists and AIP.

The rest of this thesis is organized as follows: Chapter 2 describes the previous work, Chapter 3 describes the three AIP framework models, Chapter 4 describes the data, Chapter 5 describes the experiments performed, Chapter 6 is a discussion and analysis the results of the experiments and Chapter 7 concludes this thesis.

Chapter 2

Previous Work

There have been several proposals to improve the effectiveness of blacklists. The method called BLAG [20] is a system that generates blacklists based on the aggregation of other blacklists. BLAG rates IP addresses from other blacklists using a sorting mechanism and then adds the most aggressive IP addresses to the BLAG blacklist. This method calculates a reputation for each source blacklist based on its accuracy and then aggregates the best ones. The accuracy of each blacklist is computed using the network traffic of the user, which should have an IDS. It then generates scores for each IP based on its history and the success rate of the blacklist of origin. BLAG uses recommendation algorithms, thresholds, and IP prefix expansion. The most successful IP addresses emerging from this system are blacklisted. The experiments showed that BLAG achieved high specificity (86%) and high recall (27%–61%).

Another method proposed to generate blacklists through clustering algorithms [6]. This method increased the efficacy of blacklists by targeting malicious clusters of IPs. This method identified clusters of related IP addresses and then sorted them into malicious or not malicious clusters by comparing the clusters to blacklists. The authors showed that using this method, they were able to identify malicious clusters of IPs by checking the identified clusters against different blacklist sources. This paper shows that even a blacklist with a very low recall can identify malicious clusters with a high success rate.

An alternative method [23] consisted of a system that focuses more on local traffic than on high-level blacklist comparisons. The method was proposed for blacklisting spam IP addresses and is based on dynamic thresholding and speculative aggregation. Dynamic thresholding adds IP addresses to a blacklist only if their reputation drops below a certain threshold. The reputation is established by comparing traffic from a real email server to a honeypot server. The other important method employed was speculative aggregation, meaning to predict whether unknown IPs were malicious by classifying the network of origin. This was done by aggregating the spam honeypot data together from any given network and comparing the ratios of malicious IPs to benign IPs and benign email to spam email. A decision is made based on if a given network is classified as malicious or not. If a network is classified as malicious, IPs from that network receive a higher danger reputation. This reputation is updated with new information and data. Similarly, as in the BLAG paper, both methods use thresholds and add network prefixes to block yet unknown IPs.

Regarding the evaluation of blacklists, one research study [14] performed a thorough investigation of 15 blacklists to assess their efficacy. They concluded that the union of 15 public blacklists included less than 20% of malicious domains and most AV vendor black-

lists failed to protect against malware that utilizes Domain Generation Algorithms [15]. This is an important finding that describes the reality of blacklists and is the first study to do this comprehensively.

These previous methods for blacklisting IP addresses are effective and provide helpful insight into the state of the art of blacklisting. However, they have limitations. The first two methods above are solely based on the aggregation of other blacklists. For a blacklist to better protect a specific type of device, that blacklist should be trained using data collected on that type of device. If malicious IoT traffic is not used to generate blacklists, IoT devices are not adequately defended. Both methods address the problem of ineffective IP blacklists by creating models that work with existing blacklists instead of addressing the generation of blacklists directly from captured traffic. Our proposed method is the first that provides the framework for any user to build their blacklists only based on a traffic capture of the device being protected.

The dynamic spam blacklist method [23] seems to be closer to solving the core issue of the problem by working with the generation of blacklists directly from traffic. Although moving in the right direction, this method did not include a way to lower the importance of IPs that grew older. In other words, it does not answer the questions: what happens to a malicious IP that is very active attacking for some time and then stops attacking? Does it get removed from the blacklist, or does it stay? What happens as more time goes on? Many methods lose efficiency as time goes on [25]. Our method addresses these limitations.

Chapter 3

AIP Framework

3.1 General Methodology

To address the problems mentioned above with current blacklisting approaches, the first part of this thesis proposes the Attacker IP Prioritizer framework (AIP). AIP is a framework that includes several algorithms to create blacklists and one comparison methodology. This chapter describes the three different models implemented in AIP, which inputs are traffic data and which outputs are blacklists. Two of the three models in AIP are based on a linear combination of data, and the third model is a Random Forest Classifier.

All AIP algorithms work by consuming daily data about actual attacks, computing how to process this data in different ways, and outputting a list of IPs to block the next day. Each algorithm prioritizes different aspects of the attacks, given that users may be interested in different types of attacks.

3.2 Processing of Input Data

The algorithms in AIP use data coming from several honeypots for training and evaluation. This dataset is described in Chapter 4. This section describes how the data is processed and used. In the context of this thesis, we refer to a honeypot as a device that is set up with no real interaction or usage planned. Since the collected data always comes from a device with no specified use, all the computers connecting to it are invariably attacking.

Note that this definition of an attack is true even for a regular computer connected in a network concerning its closed ports. If port 23/TCP is closed, and another computer connects to it, that connection can be considered an attack.

The data is processed in batches of 24 hours. Every 24 hours, we extract a set of features used for training each of the three algorithms. These are the guiding principles that are used to decide which features should be extracted from the data and used to help decide which IPs should be more or less important for blocking in a blacklist:

- **Aggressiveness**. The more attacks received by a particular IP, the more important it is to block it. Each attack is represented by a flow in the network, containing data such as source port, source IP address, the start time of the flow, and more.
- **Attack Duration**: The amount of time that an IP spends exchanging packets during an attack gives us information about how successful the attack may have been and how persistent the attacker was. This can be quantified by summing and averaging the length of the connections extracted from the network captures.

Table 3.1 Features extracted from the network traffic from the flows of each attacking IP

Features	Definition
`IPv4 Address`	Source IP address of the attacker
`Number of Flows`	Total connections from the given IPv4 address
`Total Connections Duration`	Sum of the connections duration
`Average Connections Duration`	Average duration for the connections
`Total Bytes`	Sum of all bytes in all connections
`Average Bytes`	Average bytes per connection
`Total Packets`	Sum of packets in all connections
`Average Packets`	Average packets per connection
`Time of First connection`	Time of the first connection (set only once)
`Time of Last connection`	Time of the last connection (updated)

- **Attack Size**: Attackers that are sending more data to the honeypots should be considered more dangerous because it suggests an active transfer, exfiltration, successful connection, or that a Command and Control server is actively conversing with the device. This feature can be quantified by the number of bytes and the number of packets that are transferred.

- **Re-occurrence**. By measuring the exact time when an IP attacks, it is possible to measure how many times and how often an IP attacks. IPs that attack for the first time recently should be considered as new attackers and dangerous, as well as IPs that attack consistently for a long time.

With these principles in mind, the features in Table 3.1 are extracted from the raw network traffic captures every 24 hours for each IP that attacks the honeypots. The basic unit of measurement is a flow (or connection), and we consider quantities in both directions inside the flow.

We extract every 24 hours a set of features for each IP that attacks the honeypots. These features are described in Table 3.1. The final processed dataset consists of this group of features per IP, per day. This dataset is then fed to the three AIP models for updating and processing.

3.3 Weighted Linear Combination Algorithms

The first two algorithms of AIP, called *Prioritize Consistent* and *Prioritize New*, use a weighted linear combination of the features previously described. Both algorithms share a similar structure: (i) database update, (ii) compute the features, (iii) use a linear algorithm, (iv) train the weights of the linear algorithm, (v) compute a score, (vi) age the score to forget IPs, (vii) use a threshold to decide which IP are included in the final blacklists. The main differences are how the training of the weights is done, and the type of aging function applied.

Database Creation and Update The core of the linear models in the AIP framework is a database. This database per IP address remembers all the information ever produced by each IP address and is updated with the new information coming every day. We refer

to this database as *main* database. Therefore, the first step of the update algorithm is dealing with this main database. During this phase, the goal is to convert the daily features computed and described in Section 3.2 into a database of features per IP address. The main database contains a list of all the IP addresses that have ever been encountered by the current instance of AIP, along with their corresponding features. This database is essential because it is used as a form of memory for the linear models. During the database update phase, AIP receives data from the last 24 hours for a certain number of IP addresses. For each of these IP addresses, its past known features in the main database is updated using the new data. The update is done by summing the historical totals with the new totals and recalculating the running averages. The data update algorithm is shown in Algorithm 1.

Algorithm 1: AIP algorithm to update the main database with new daily data

```
Result: Updated database
for IP in New_Data do
    if IP is in the database then
        active-days += 1;
        total-ip-flows += new-ip-flows;
        average-ip-flows = total-ip-flows/active-days;
        total-ip-duration += new-ip-duration;
        average-flow-duration = total-ip-duration/total-ip-flows;
        total-ip-packets += new-ip-packets;
        average-flow-packets = total-ip-packets/total-ip-flows;
        total-ip-bytes += new-ip-bytes;
        average-flow-bytes = total-ip-bytes/total-ip-flows;
        time-of-last-flow = new-time-of-last-flow *;
    else
        active-days += 1;
        total-ip-flows = new-ip-flows;
        average-ip-flows = total-ip-flows;
        total-ip-duration = new-ip-duration;
        average-flow-duration = total-ip-duration/total-ip-flows;
        total-ip-packets = new-ip-packets;
        average-flow-packets = total-ip-packets/total-ip-flows;
        total-ip-bytes = new-ip-bytes;
        average-flow-bytes = total-ip-bytes/total-ip-flows;
        time-of-last-flow = new-time-of-last-flow ;
        if First time IP appears then
            time-of-first-flow = new-time-of-first-flow
        end
        ;
    end
end
```

Thus, after the update using the new data, there are eight features with corresponding values for each IP in the database. These values are used later for training the models and scoring the IPs.

Once it is updated with the new data, this main database is used to train the two

Table 3.2 Extracted Features per IP for Training

Feature	Time Frame of Feature
`Number of Flows`	All Time
`Average Flows`	Average Per Day
`Total Connection Duration`	All Time
`Average Connection Duration`	Average Per Flow
`Total Bytes`	All Time
`Average Bytes`	Average Per Flow
`Total Packets`	All Time
`Average packets`	Average Per Flow

linear models: the Prioritize Consistent Model and the Prioritize New Model. These two models compute a score for each IP in the historical database based on the aggregated data for that IP. This score is meant to designate the importance for each particular IP to be blocked. The higher the score, the greater the importance of blocking.

Having the IPs rated by a score according to their importance is a very useful feature because it allows for the easy scaling of the blacklist according to the application's needs. When a blacklist is implemented in a device as part of an IP-based solution, it is loaded and stored in the device's memory. Thus, the memory capabilities of the device can be a bottleneck for how large the blacklist can be. In larger systems such as servers and industrial firewalls, this is not a concern since these devices tend to have plentiful memory. However, in IoT devices and other smaller devices such as routers, memory can be a limitation. Every IP that is implemented in a resource-restricted system needs to be as effective as possible. The scoring-based method for a blacklist allows the end-user to choose the number of IP addresses that can fit the system for the most significant effect. On the other hand, if the blacklist is being implemented in a high-end system where memory is not an issue, the number of IPs can be easily scaled to achieve the best result possible.

3.3.1 Prioritize Consistent Algorithm

The Prioritize Consistent algorithm (from now on PC) is designed to give higher scores to IP addresses that consistently attack the network over a long period. IP addresses of this sort are less likely to be coming from IoT devices or cloud deployments since those tend to be infected and uninfected at a swift pace. These are more likely to be more permanent servers. There are two main parts of the PC algorithm; the scoring function and the aging modifier.

This algorithm starts by using the main database that was described in the first part of Section 3.3, which means that every day, the main database is updated, and the algorithm is ready to use the new data. The first step of the algorithm is to create a score for each IP address using the eight features that are listed in Table 3.2.

The calculation of the *score* for each IP address is presented in Equation 3.1, where $feature_i$ is the value of each data feature, n is the number of features, and w_i is the weight corresponding to the feature $feature_i$. Notice that the weights are the same for

all IP addresses.

$$score_{IP} = \sum_{i=1}^{n} w_i * feature_i \tag{3.1}$$

The score is the sum of the eight weighted features and defines each feature's importance in the final score. The sum of all the weights is 1. In the PC algorithm, these weights were defined to prioritize the consistency of the attacks exactly. The highest weights are assigned to the four average values to prioritize IPs in the list that attack consistently and transfer more data than others. The values for the weights in the PC algorithm are presented in table 3.3.

Table 3.3 Data Weights for PC Model

Features	Weight
`Number of Flows`	0.05
`Average Flows`	0.2
`Total Connection Times`	0.05
`Average Connection Times`	0.2
`Total Bytes`	0.05
`Average Bytes`	0.2
`Total Packets`	0.05
`Average packets`	0.2

Table 3.4 Data Weights for PN Model

Features	Weight
`Number of Flows`	0.2
`Average Flows`	0.05
`Total Connection Times`	0.2
`Average Connection Times`	0.05
`Total Bytes`	0.2
`Average Bytes`	0.05
`Total Packets`	0.2
`Average packets`	0.05

The score is a linear combination of the weighted features for each IP.

An investigation of how the eight features listed in Table 3.2 grew over time showed that the scoring methodology needed to be altered to account for the scale of values. An example of an IP entry in the main database is shown in Table 3.5 after it was updated for four months. As it can be seen, the values for the features, specifically the total number of bytes, are orders of magnitude larger than the others. Because the final score is a weighted sum of the values in this table, excessively large values can dominate the calculation outcome. An example calculation for the score for the example IP in Table 3.5 is shown in Equation 3.2.

$$score_{43.254.240.34} = \sum_{i=1}^{n} w_i * feature_i = 403,365.2603 \tag{3.2}$$

Table 3.5 is a breakdown of the different features that make up the score in Equation 3.2. As can be seen in the table, the total bytes feature compromises 97% of the final total score since its weighted value is 394,783. Thus, the rest of the values had almost no effect on the final score. We found that this was true for most IPs that stayed on the blacklist for an extended period. In order to solve this, we introduced a normalization methodology for the features.

For each IP in the main database, the features shown in Table 3.2 are normalized using Equation 3.3, where F_{value} is the feature value for each feature (example Total Bytes), F_{Min} is the smallest F_{value} for all IPs in the entire main database up to now, and F_{Max} is the largest F_{value} for any IP in the main database until now. We are aware that the maximum and minimum values may change every day; therefore, old scores may not

Table 3.5 Example IP from Main Database After 4 months

Entry	Value	PC Score
`IP address`	43.254.240.34	N/A
`Number of Flows`	3,675	183.75
`Average Flows`	5.42	1.09
`Total Connection Times`	96,310.37	4,815
`Average Connection Times`	26.206	5.24
`Total Bytes`	**7,895,676**	**394,783**
`Average Bytes`	2,148.48	429.7
`Total Packets`	62,855	3,142.8
`Average packets`	17.103	3.42

be directly comparable with new scores. However, our algorithms do not compare past scores. This normalization is also called Min-Max normalization.

$$Normalized_F = f(F_{Value}, F_{Min}, F_{Max}) = \frac{F_{Value} - F_{Min}}{F_{Max} - F_{Min}} \tag{3.3}$$

After the normalization of the features Equation 3.1 can be rewritten as Equation 3.4.

$$normalizedScore_{IP} = \sum_{i=1}^{n} w_i * Normalized_{F_i} \tag{3.4}$$

Aging of Attacks One of the essential concepts in Threat Intelligence and blacklists, in general, is when to *stop* blocking the IP addresses. This is a complicated problem for two reasons. First, if the IP of an attacker is taken out of the list, it is not blocked anymore, and new attacks may happen. Second, if the IP of a benign computer is never taken out of the list, it is continually blocked, and normal services can lose access. A good aging function is then proposed that is dynamic in how it ages the IP.

The aging function is the second core part of the AIP algorithm. Every time the main database is updated with new data, the last connection time for IP addresses is also updated. If that IP has not been seen to attack in the last update (usually one day), the time of its last connection is left unchanged. The time of the first connection is never updated.

The aging function for the PC algorithm is designed to consider the first time and the last time an IP is connected to estimate how long the IP has been attacking. The aging function is applied to the score of each IP after the score has been computed for that day.

Before aging the score of any IP address, the aging function checks if the IP should be aged or not. If the IP was seen in the last update (meaning the last 24 hours), its score is not modified. That means that the score is not aged in the PC algorithm if the IP attacks every day. This is consistent with the idea that as long as the IP attacks, it keeps its score *fresh*.

On the other hand, if the IP address has not been seen attacking in the last update, the function ages the score by multiplying the score by the aging function shown in Equation 3.5:

$$a\_pc(x, y) = 1 - \frac{x}{x + y} \tag{3.5}$$

$$y = time\_of\_last\_flow - time\_of\_first\_flow \tag{3.6}$$

In Equation 3.5, x is the amount of time since the IP address attacked last, and y is the total amount of time that the IP was found to be attacking, i.e., the last recorded event time minus the first event time (Equation 3.6). The output of this aging function is a decimal value between 0 and 1 multiplied by the score, thus decreasing it by some value.

Figure 3.1 illustrates the shape of the aging function as the values for x and y change. As shown in the graph, the highest values are given to IPs that attacked for longer periods since the last attack was the smallest. Thus, IPs that are the most consistent in their attacks are aged the least. The lowest scores are given to IPs with low y values and high x values, meaning they attacked for a short time, and the last attack occurred a long time ago. In this way, the score for a given IP is decreased over time depending on its attack times.

The output of the aging function is then multiplied by the initial score, hence decreasing the IPs score, making the pc_final_score in Equation 3.8 a combination of Equations 3.4 and 3.5:

$$pc\_final\_score = a\_pc * normalizedScore \tag{3.7}$$

Re-written as:

$$pc\_final\_score = (1 - \frac{x}{x+y}) * \sum_{i=1}^{n} (w_i * Normalized_{F_i}) \tag{3.8}$$

The last step of implementing the PC algorithm is to use a threshold to restrict the number of IP addresses in the final blacklist not to grow indefinitely. To do this, we implemented a threshold score of 0.002 so that only IPs that achieve a score higher than this are included in the final blacklist. The threshold was defined as an ad-hoc heuristic by experts by a careful analysis of the impact of the blacklist.

3.3.2 Prioritize New Algorithm

The Prioritize New algorithm (from now on, PN) is designed to give higher scores to IP addresses that are new and aggressively attacking the network over a short period. IP addresses of this sort are likely to be coming from IoT devices or cloud deployments since those tend to be infected and uninfected at a swift pace. Like the PC algorithm described in Section 3.3.1, there are two parts of the PN algorithm; the scoring function and the aging function.

This algorithm starts by using the main database that was described in the first part of Section 3.3, which means that every day, the main database is updated, and the algorithm is ready to use the new data. The first step of the algorithm is to create a score for each IP address using the eight features that are listed in Table 3.2.

The calculation of the *score* for each IP address is presented in Equation 3.1, where $feature_i$ is the value of each data feature, n is the number of features, and w_i is the weight corresponding to the feature $feature_i$. Notice that the weights are the same for all IP addresses.

The score is the sum of the eight weighted features and defines each feature's importance in the final score, illustrated in Equation 3.1. Like the PC algorithm, the features

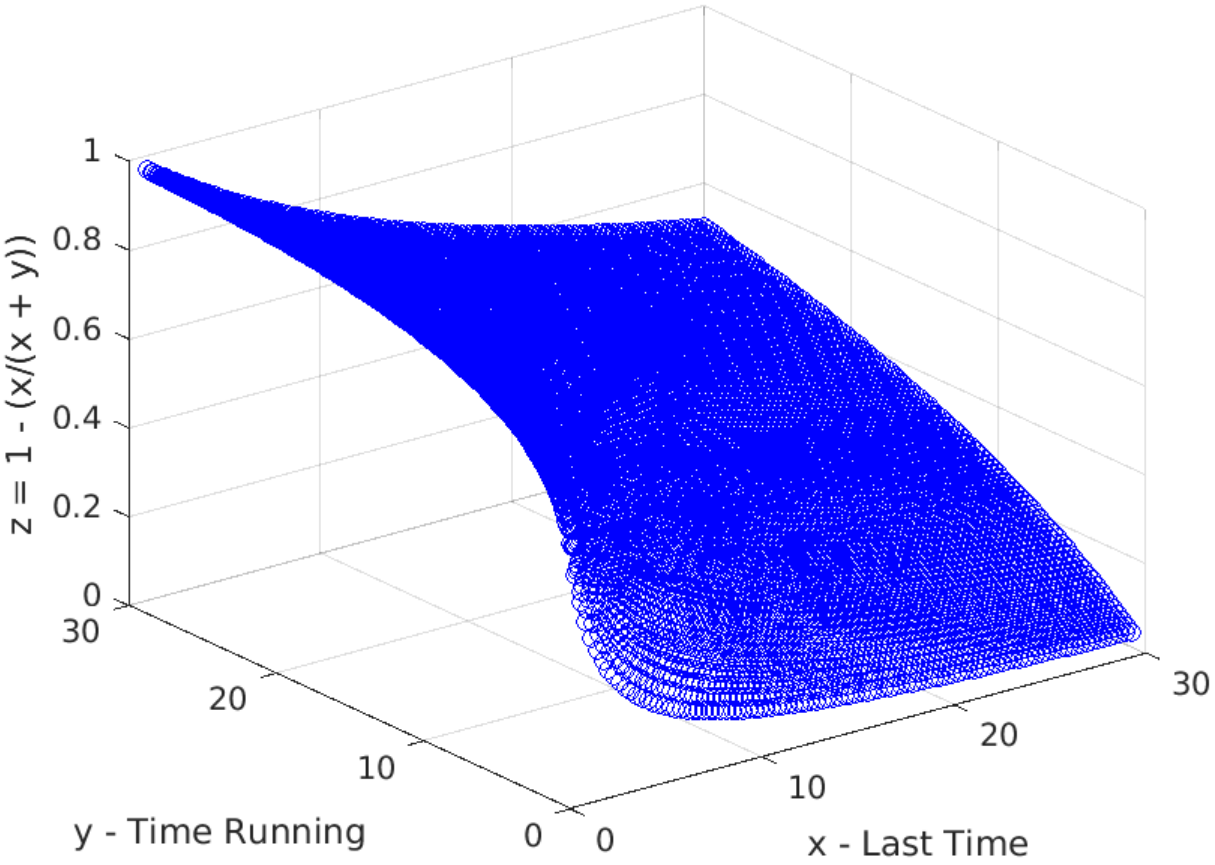

Figure 3.1 Graph of Aging Function for the Prioritize Consistent AIP algorithm

from the main database are normalized using Equation 3.3. The sum of all the feature weights is 1. In the PN algorithm, these weights are defined to prioritize IPs with the most significant recent attacks. Thus the highest weights are assigned to the four total values. This is because a new IP does not have a trustworthy average value since averages are normalized over time. Thus, the best way to compare a new attacker to another new attacker is with the features that represent totals. Thus the total features are assigned higher weights. The values for the weights in the PN algorithm are presented in table 3.4.

The aging function is the other main difference between the PN and PC linear algorithms. Before aging the score of any IP address, the aging function checks if the IP should be aged or not. If the IP was seen in the last update (meaning the last 24 hours), its score is not modified.

On the other hand, if the IP address has not been seen in the last update, the function ages the score using an aging function different from the one used for PC. Equation 3.9 shows the aging function for PN, where x is the amount of time since its last attack (in units defined by the user, but we use one day), and c is a constant to control the rate at which the score is decreased over time:

$$a\_pn(x) = \frac{c}{c + x} \tag{3.9}$$

Different values for a are illustrated in Figure 3.2.

As it can be seen from the Figure 3.2, the different values for a cause the rate of score aging to increase or decrease as time goes on without an attack. The lower the value of c, the faster the score decreases, and the higher the value of c, the slower it decreases.

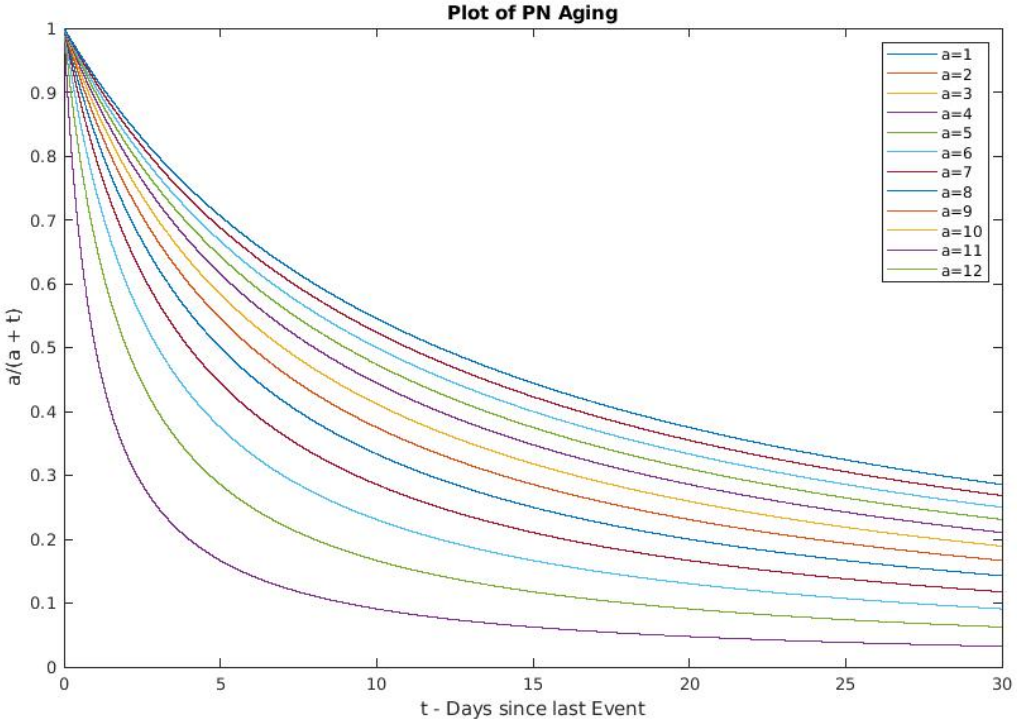

Figure 3.2 Graph of Aging Function for the Prioritize New algorithm

The value for c is constant across all IP addresses, and we used expert knowledge to fix its value for our experiments in $c = 2$. The output value of the aging function, shown in Equation 3.9, varies between 0 and 1, decreasing with the number of days since the last connection from that IP. The aging modifier is multiplied by the previously computed normalized score, making the $final\_score\_pn$, shown in Equation 3.10.

$$pn\_final\_score = (\frac{c}{c+x}) * \sum_{i=1}^{n}(w_i * Normalized_{F_i}) \tag{3.10}$$

The last step of implementing the PN algorithm is to use a threshold to restrict the number of IP addresses in the final blacklist to not grow to infinity. To do this, we implemented a threshold score of 0.002 so that only IPs that achieve a score higher than this are included in the final blacklist. Experts selected this score as an ad-hoc heuristic value based on the performance of the blacklist.

3.4 Random Forest Algorithm

The third AIP algorithm is built using the Random Forest algorithm (from now on RF) and has an entirely different design than the two previous linear models described in Sections 3.3.1 and 3.3.2.

A full explanation of the Random Forest Classifier is beyond the scope of this thesis, but a basic explanation of it is warranted. Random forest is an ensemble learning method for classification that is based on the usage of decision trees to make predictions on labeled datasets [26]. It is a supervised learning method. Random forests use Bootstrap aggregation [8] as it ensemble method, which is a method of binning the training data

by sampling a given training set uniformly and with replacement. It trains different trees on each of the training datasets using randomly chosen features to reduce the correlation between the different trees. The output of the decision trees is the aggregated values of the trees, and it is used to predict on a new set of data.

In order to explore other machine learning models and verify the correct functioning of the RF, we compared the Random Forest classifier to another common model, Logistic Regression [21]. After training the two models and then testing, we compared the two receivers operating characteristic (ROC) curves. These curves are shown in Figure 3.3 and 3.4. As shown in the comparison, the Random Forest classifier performed much better than the Logistic Regression model and had a higher AUC (area under the curve) value, the Random Forest Having $AUC = 0.8339$, and Logistic Regression having $AUC = 0.7232$.

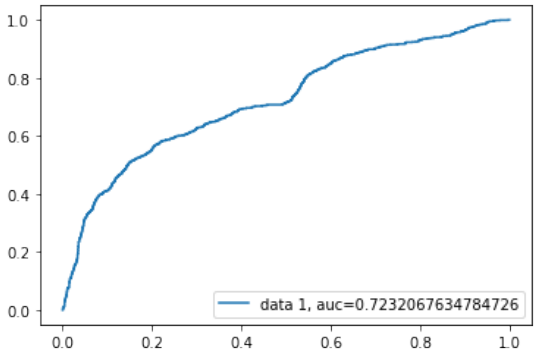

Figure 3.3 ROC curve for Logistic Regression

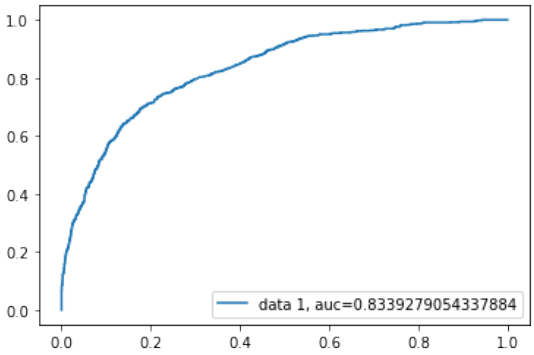

Figure 3.4 ROC curve for Random Forest

Our implementation of the classical RF algorithm uses a unique dataset. This was done by comparing each new day of network traffic, which is described at the beginning of this Chapter in Table 3.2, to the traffic collected on the next day. Each data flow in the data from day one is compared to the data in day two and labeled with 1 if the IP occurs and 0 if it does not. This is based on the concept that each dataset of IPs contains IP addresses that may or may not attack the network again during the 24 hours after the data is collected. Labeling the data flows in this way is meant to capture the features of the IPs that make the attacks and the features of the IPs that do not. It should be noted that because of this need to label the data, the RF model is continually training on data that is 24 hours old since the only way to know if a specific IP is going to attack is to wait and see if it does.

In order to build a more extensive dataset over time, each day, the data file from the day before is labeled using the most recent 24 hours of data and then concatenated with a historical dataset. This historical dataset of labeled data flows is then used to train the Random Forest classifier.

The algorithm for training and predicting using a Random Forest Classifier is shown in Algorithm 2. The numbers $[1, 4, 7, 10, 13, 16, 19, 22, 255]$ refer to different models of the

Table 3.6 Hyper-parameters of the Random Forest Algorithm. Trained with heuristic random search.

Parameter	Definition
`warm_start`	Reuse the solution of the previous call to fit
`oob_score`	Whether to use out-of-bag samples
`n_estimators`	Number of trees in the forest.
`min_samples_split`	Minimum number of samples required to split an internal node
`min_samples_leaf`	Minimum number of samples required to be at a leaf node
`min_impurity_decrease`	Determines if a node is going to be split
`max_features`	Number of features to consider
`max_depth`	Maximum depth of the tree
`criterion`	Function to measure the quality of a split
`bootstrap`	Whether bootstrap samples are used

Random Forest algorithm.

Algorithm 2: Random Forest blacklist generation

```
   Result: Random Forest Blacklist
 1 new_labeled_data = [];
 2 for IP flow in Unlabeled_data_from_48_to_24_hours_ago do
 3 |   if IP flow is in Data_from_last_24_hours then
 4 |   |   new_labeled_data += IP Flow + label 1;
 5 |   else
 6 |   |   new_labeled_data += IP Flow + label 0;
 7 |   end
 8 end
 9 main_database += new_labeled_data;
10 trained_classifier = RandomForestClassifier(main_database);
11 for X in [1, 4, 7, 10, 13, 16, 19, 22, 25] do
12 |   predictions = trained_classifier.predict(last X days of data);
13 |   create_blacklist_for_X_days_of_data(predictions);
14 end
```

The hyper-parameters shown in Table 3.6 need to be set to run a random forest classifier using the sklearn [3] library in Python [19]. In order to find the best parameters to use for the random forest classifier, we did a heuristic random search of a matrix of different parameter values using the built-in RandomizedSearchCV [3] function. Table 3.7 shows the values that were found to perform the best on the dataset that was tested on.

To generate a blacklist of IPs, the model needs to have a set of data to predict on. As shown in line 10 of Algorithm 2, the classifier is trained on the entire updated main database. Notice that we do not use cross-validation in the training process.

The classifier is then used to predict what IPs occur in the next 24 hours. The IPs that are predicted on are selected from the most recent section of the main database. The classifier labels each IP in the prediction dataset with 1 or 0 depending on the eight features in Table 3.2. The list of IPs labeled with 1 is then used to create the blacklist for the next day.

In order to test if there is a gain in performance for choosing a larger number of IPs to predict on, the Random Forest Classifier predicts on eight sections of data. As can be

Table 3.7 Values of the Hyper-parameters of the Random Forest after the heuristic random grid search

Parameter	Definition
warm_start	True
oob_score	False
n_estimators	150
min_samples_split	4
min_samples_leaf	3
min_impurity_decrease	0.0012
max_features	log2
max_depth	20
criterion	entropy
bootstrap	True

seen in lines 11-13 of Algorithm 2, the RF classifier is used to predict on the last day of collected data, the last 4 days, 7, 10, 13, 16, 19, 22, and 25 days of data. Each set of data is predicted on, and a separate blacklist is generated from the predictions.

Because each day new data is being added to the main dataset of labeled flows, the processing time for the Random Forest model grows at a linear rate, depending on the amount of data added each day. This is not very good, so to limit processing time, the dataset is capped at about one month. Thus, for the first 30 days of generating blacklists, the main dataset grows, and once it reaches 30 days, the first day of data collected is deleted from the dataset since it is the oldest data, and the newest data is added to the end of the dataset. In this way, the processing time is capped, and the most recent data is always used to update the dataset while only losing the oldest data. This decision was made because the training time of the model grew at a linear rate.

3.5 All IP Blacklist

We also created a baseline blacklist of what can be considered one of the *best* blacklists: a blacklist that remembers every attacker ever to connect to the honeypots. It is called the All-IP blacklist. Comparing to this blacklist is crucial because it is an upper limit of efficiency for a blacklist based on traffic from honeypot networks. Without using any forward prediction techniques, a model can only choose from IPs that attacked the network in the past when deciding what IPs to block. This All-IP blacklist can therefore be considered an upper limit of raw blocking without prediction. Users do not use this type of blacklist because it grows at a linear rate, making it almost impossible to implement in most hardware.

3.6 Evaluation Framework

One of the most important issues of current research on blacklists is the lack of a good evaluation. To overcome this problem, we developed a statistical-based methodology for evaluating the efficacy of the different models for blacklist generation. The methodology aims to measure how many attacks would be blocked *tomorrow* if the blacklists are

updated daily, up to today. The methodology is as follows:

- Start with an empty list of attacker's IPs and empty blacklists.
- Every 24 hours, take the attack data from the last 24 hours.
- Update each blacklist with the new data according to their specific method.
- Consider all the IPs in each blacklist as predictions to block in the following 24 hours for that blacklist.
- Receive the attacks of the following 24 hours.
- Evaluate the efficacy of the blacklist according to how much traffic it would have blocked in those following 24hs.

Figure 3.5 is an illustration of this methodology. On each step, the blacklist is retrained according to its specific generation methodology. Each blacklist here decides how to use the new data arriving. Some blacklists concatenate all IPs, and others take the ones attacking more and so on. However, it is clear that for the blacklists to stay effective, they need to update to keep up with the changing attackers. In the second step, the already retrained blacklist is used to predict which IPs should be blocked in the next time slot, therefore testing its efficiency.

To compare how the AIP blacklists perform against others, we downloaded four other open-source blacklists for the same four-month period at 24-hour intervals. These blacklists are IPsum [24], an open-source conglomerate of many blacklists into a single feed, FireHOL [18], a threat intelligence feed by The FireHOL Project, DigitalSide [9], a threat intelligence feed by DigitalSide Threat-Intel Group, and Emerging Threats [12], an open-source feed by Proofpoint Incorporated. Each of these threat intelligence feeds was evaluated using the same methodology described above, the results of which are presented in Chapter 5. For each blacklist, we downloaded a new file every day, and that is the list used for that day.

The first step in the evaluation is to choose a time slot window. This time slot represents the amount of time between updates for the blacklist in question. We decided to update it every 24 hours, as is explained in Chapter 3. Thus, in order to evaluate its efficiency, we generated four months of 24-hour sections of data using the methodology explained in Chapter 4.

For each new 24-hour data section, all blacklists in our comparison were updated and regenerated. In the case of the third-party blacklists, we use the blacklist published for that day.

Then, to ascertain its efficiency, the updated blacklists were tested with the malicious traffic from the following 24-hour time slot, as is illustrated in Step 1 of Figure 3.5. This allowed us to see how much of the malicious traffic the blacklist would have stopped if implemented in a firewall. Then, after that comparison was completed, the blacklists were again updated on the next 24-hour slot of data and then compared to the slot after that, as is illustrated in Steps 2 and 3.

The evaluation method was to compare how much of the attacks of *tomorrow* would have been blocked. However, how to assert the concept of *how much*?. For this, we used four metrics: the total amount of bytes blocked, total duration of the flows blocked, total unique IP blocked, and total flows blocked.

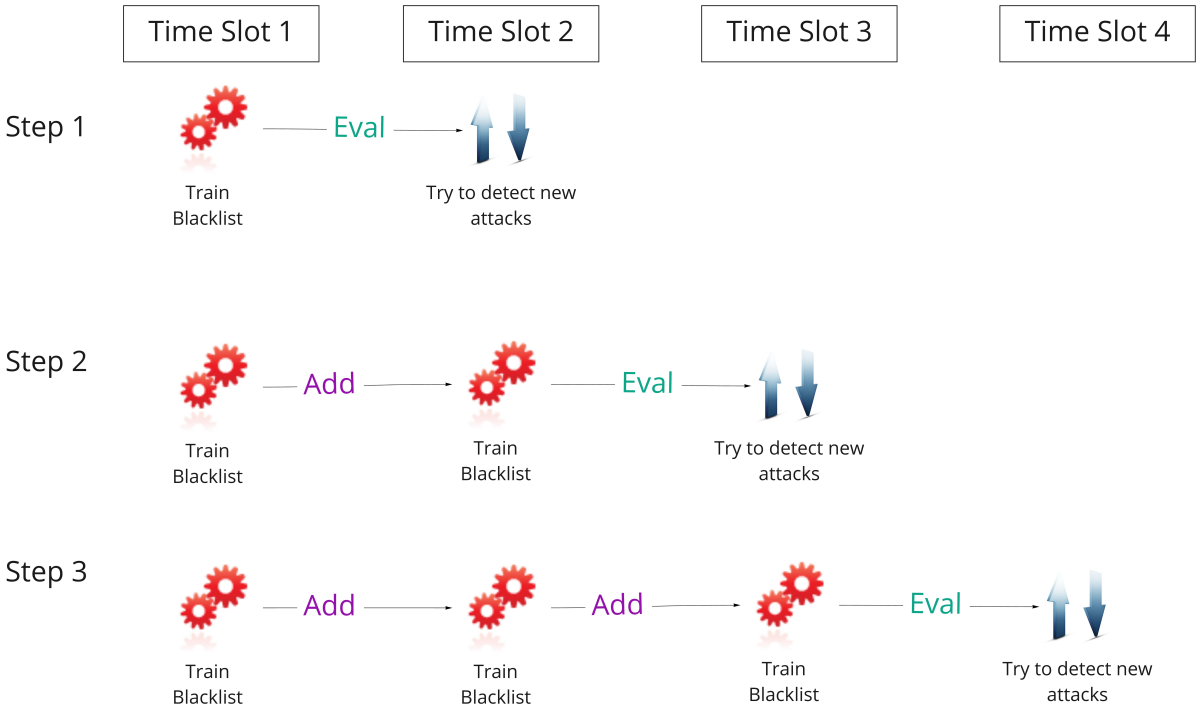

Figure 3.5 AIP Framework evaluation methodology

Table 3.8 Performance Metrics

Metric	Definition
`Total Bytes`	Total amount of data sent by all malicious IPs
`Total Duration`	Total amount of connection time by all malicious IPs
`Total Unique IPs`	Total number of IPs that attacked during time slot
`Total Flows`	Total number of individual attack attempts by IPs

These metrics are shown in Table 3.8. Total Bytes is meant to quantify how much data all the malicious IPs were responsible for sending during the given 24 hour period. Total duration is the amount of time the IPs spent connected to the honeypots. Total Unique Flows refers to the number of unique connection attempts made to the honeypots, successful or not. Lastly, the number of Unique IPs is self-explanatory.

Each IP on every blacklist is compared to the evaluation data and checked to see if it attacked or not. If it did, the amount of data associated with each metric is saved. We extract the bytes and the duration of the flow for each flow in the attack, then we sum the number of flows, bytes, and duration, and we sum 1 to the IPs blocked by the blacklist. Once all IPs in a blacklist have been checked, the percentages of the metrics are calculated by comparing them to the totals values of bytes, duration, flows, and IPs that attacked. Thus, the output for the evaluation of a given blacklist is the four metrics per day. Once this is complete, final averages can be calculated and graphed for use compared to other blacklists evaluated using the same methodology.

In this way, it is possible to create a standard way of comparing blacklists for their efficacy. Each of the four metrics gives valuable information for evaluating a given blacklist

for a specific task. For example, a blacklist meant to protect a network from a Distributed Denial of Service Attack (DDoS) [29] should perform well in blocking the bytes and events metrics since these type of attacks take up much bandwidth. This evaluation methodology is used in this thesis to evaluate the blacklists produced by the three models in the AIP framework.

Chapter 4

Datasets

The dataset in this thesis was created by capturing traffic from a network of real IoT devices. These devices are used as dedicated honeypots, without human interaction; thus, all incoming connections are considered attacks, with some basic filtering techniques implemented to avoid artifacts of the capturing tools.

The capture server monitors each honeypot, and each connection, established or not, to the honeypot is logged as a flow in the dataset. This dataset is therefore comprised of all the attack flows on the IoT network.

4.1 Design of the IoT Laboratory

The Aposemat IoT laboratory [10] used to create the dataset is composed of real IoT devices, software honeypots, and IoT devices infected with malware. The real IoT devices include the devices listed in Table 4.1. The general design of the lab and capture methodology is illustrated in Figure 4.1. As can be seen, all devices are set up in a LAN connected to a Cisco switch, and the switch is connected to a router. In order to have all the devices connected directly to the Internet, as would be the case if they were connected via a 5G SIM card, all the devices are assigned external IP addresses, and all the traffic is bridged to the devices to and from the Internet. In order to capture all the traffic without altering it, the traffic is mirrored in the Cisco switch to a capture server, which generates network captures every 24 hours. This thus emulates the case where a modern smart home is connected directly to the Internet via a 5G or other connection, and thus we can capture and train AIP algorithms on the traffic.

4.2 Data Processing Techniques

Since the devices are connected directly to the Internet, and they have no actual human interaction, from a general point of view, most connections from outside the network to the honeypots can be considered attacks. Therefore, the raw network traffic is sorted so that only the incoming connections are included in the final dataset.

To be more precise, only connections that have an SYN flag [22] originating from outside the network are included. The traffic captures are processed using the Argus network flow system processing software [17]. Argus processes the pcap files [13] and converts them into data flows.

However, when sorting through the data flows and testing, it was discovered that about

Table 4.1 Devices used as honeypots

Device	Use
Amazon Echo Dot 1st gen	Honeypot
Amazon Echo Dot 2nd gen	Honeypot
Philips Hue Bridge and Devices	Honeypot
Synology NAS	Honeypot
Wyze Home Hub and Devices	Honeypot
Edimax IP Camera	Honeypot
HikVision IP Camera	Honeypot
Home&Life Router	Honeypot
ZyXel Router	Honeypot
Google Chrome-cast	Honeypot
Google Nest Home WiFi System	Honeypot and Access Point

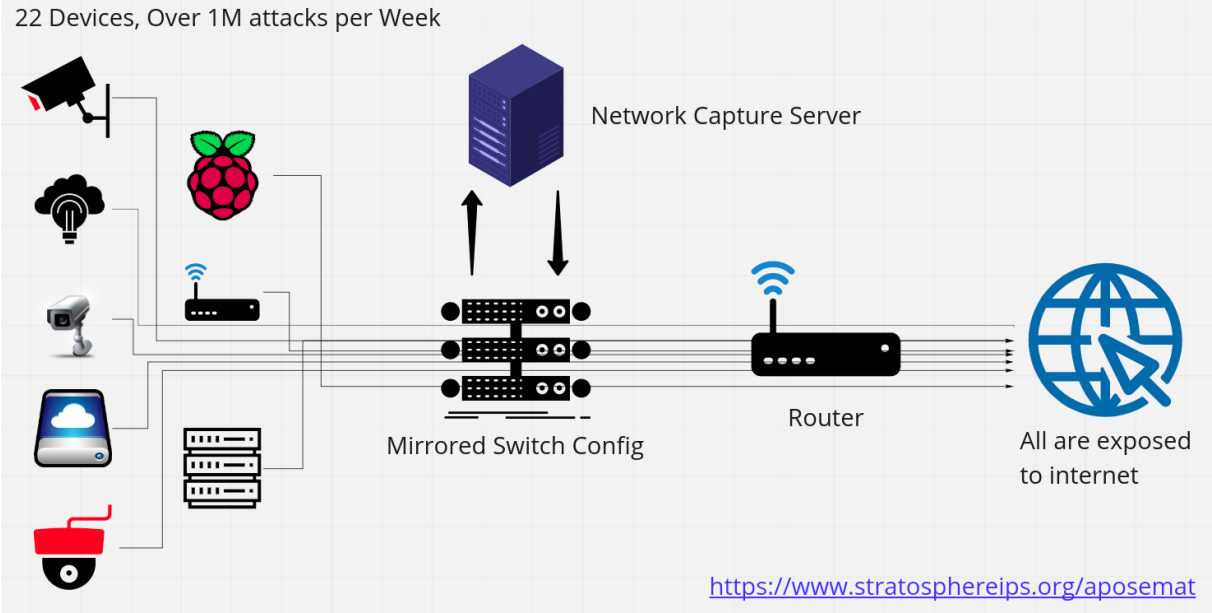

Figure 4.1 Aposemat IoT Lab capture method

10% of the final data were false positives. This was because the Argus tool maybe not see the initial SYN packets and therefore believe that a connection from a honeypot to a typical server on the Internet is a connection *from* that server. For the context of this thesis, we define a false positive as an IP address associated with a known non-malicious organization. We discovered that some IP addresses associated with large corporations such as Microsoft, Google, and Twitter would regularly initiate connections to the honeypots in the capture network.

The problem of the FP is in the Argus tool. Let H be a honeypot server and I a server on the Internet. Given a connection from the honeypot H to the real server I, it should not be considered in our database of attacks because it was originated in the honeypot. Argus, however, may lose some packets (like the original SYN packet) and therefore create a flow falsely originating from I to H. In order to recognize these false connections to the honeypot, we implemented the sorting by ports and ASN.

In order to deal with these FP connections, we implemented two more filters to the

Table 4.2 Common dynamic port ranges [7]

Operating System	Range
Free BSD	49152 to 65535
Debian	32768 to 60999
Windows	49152 to 65535
Solaris	32768 to 65535

traffic, a port sorter and an ASN (Autonomous System Number) database sorter. The port sorter analyzes the source ports in the connections and checks that the ports are assigned in a specific way. The motivation to sort by source port is that connections originating from a normal (not attacking) device are usually assigned a random port from the ephemeral port number range assigned to that device by its developers [27]. These ports are dynamically assigned when the device starts a connection to another device or server. Thus, when an attacking device connects to a honeypot device, it is assigned a fixed random port or a port from a non-common range to initiate the connection. In comparison, a connection from a legitimate operating system comes from an assigned port range. According to RFC 6335 [7], ports from the range 49152-65535 are reserved for dynamic user assignment, whereas System Ports are 0-1023 and user ports are 1024-49151. However, many developers do not hold this standard. Table 4.2 shows the ranges for major Operating systems.

Therefore, to restrict our data to only IP addresses that were initiating a connection to the honeypot and were not mistaken by Argus, we implemented a filter that only accepts connections from source ports greater than 32,000. This proved to be able to remove the bulk of the false positives from our data.

Even after sorting the connections by incoming connections only and from specific port ranges, there were still a few IP addresses that were false positives in the data. As a last precaution, we implemented an Autonomous System Number, or ASN, verification method. ASNs are 16-bit or 32-bit numbers assigned to a specific set of IP prefixes belonging to an organization used to maintain a single, clearly defined routing policy [11]. Thus, using an ASN database, a specific IP address can be checked to see if it is registered to a specific entity. The ASN verification method is based on the motivation that IP addresses of attackers do not usually come from well-known organizations. To implement this, we downloaded the free Geo-IP database of ASN information provided by MaxMind [16] and used it to check the registration information for each IP. We compiled a list of major service providers, and after each blacklist is generated, each IP in the blacklist is checked to see if they are registered to a major service provider, and they are removed if this is found to be true.

Therefore, all the network traffic is filtered using the above three stages, namely, including only incoming connections, checking source ports, and ASN verification. Once these steps are done, the output traffic flows from Argus are sent to the database update phase described in Chapter 3.

4.3 Hornet 15 Dataset

A possible variable that can influence the efficacy of blacklists is location. It is possible that a blacklist that is generated using data collected in one country is not as effective

Table 4.3 Honeypot locations for data validation

Location	Number of Honeypots
Amsterdam	1
Bangalore	1
Frankfurt	1
London	1
New York	1
San Francisco	1
Singapore	1
Toronto	1

in another country because different groups are targeting it. Therefore, to perform an evaluation that is not geographically constrained, we used the Hornet 15 Dataset [28].

This is a dataset of traffic flows collected from honeypots in eight different locations around the world, the exact locations shown in Table 4.3. The data collection method is the same as explained in Section 4.2. The data connections are captured and processed by Argus and aggregated into connection flows. We processed these flows and then sorted them according to the incoming connections, port, and ASN data to remove false positives from the attacks. This dataset is used in Chapter 5 to evaluate if the results of the evaluations explained in that chapter are location biased.

Chapter 5

Experiments and Comparisons

The methodology to compare blacklists was explained in Section 3.6, and it was used to evaluate the efficacy of the three AIP models as well as the four downloaded blacklists over four months.

Using the above methodology of comparing each blacklist to the next 24 hours of malicious network traffic and computing four metrics for comparison, we were able to compute the average performance for each blacklist, which is displayed in Table 5.1. Comparing only the size of the blacklists is not enough to understand the differences, that is why we evaluated all different features.

Experiment of Comparing the Size of Blacklists The first comparison between all the blacklists was regarding their size and can be seen in Figure 5.1. Every day each blacklist was updated according to its rules. In the RF models, the vertical lines are because there has to be a wait of different amounts of time to have enough data to train them. Note that the scale of the size is logarithmic, given the significant differences in size. It seems that the IPsum blacklist has a limit in the amount of IPs and FireHOL, DigitalSide-Threat Intel, and Emerging Threats.

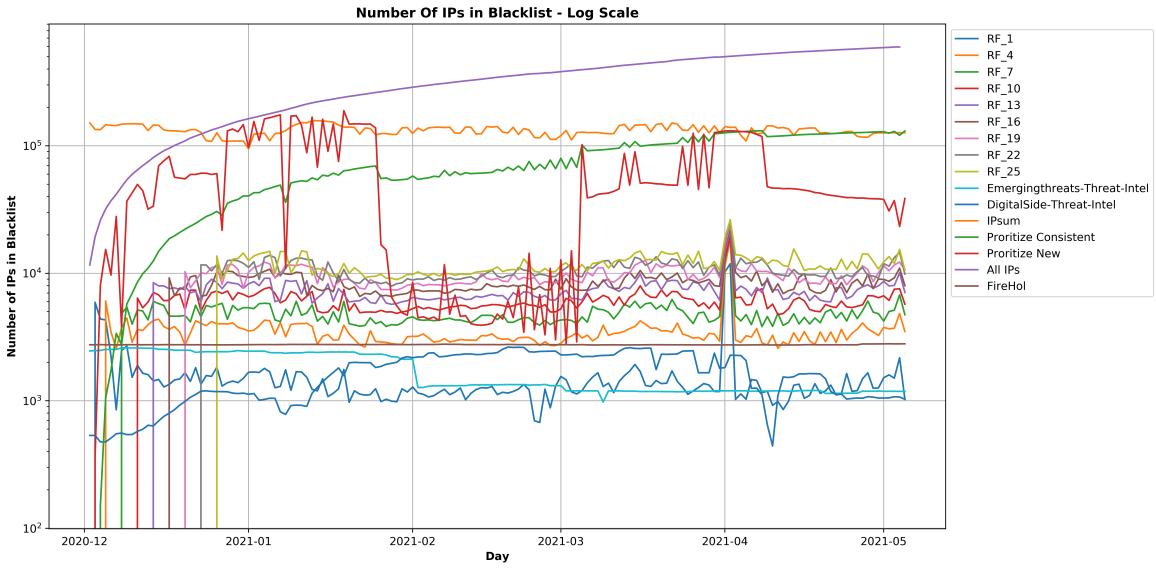

Figure 5.1 Comparison of Blacklist Sizes in Log Scale. RF_X stands for Random Forest model trained up to X days in the past. Y axis is in Logarithmic scale.

Experiment of Comparing the Bytes Blocked The following comparison between all the blacklists was regarding their total bytes blocked and shown in Figure 5.2. The ALL IP blacklist is the best, followed by IPsum, the PC blacklist, and the PN blacklist. The RF blacklist performs much less in absolute terms. However, the FireHOL, DigitalSide-Threat Intel, and Emerging Threats are the lines that are close to zero at the bottom of the graph. At the end of the evaluation, there is a spike in performance in all the blacklists, except FireHOL, DigitalSide-Threat Intel, and Emerging Threats.

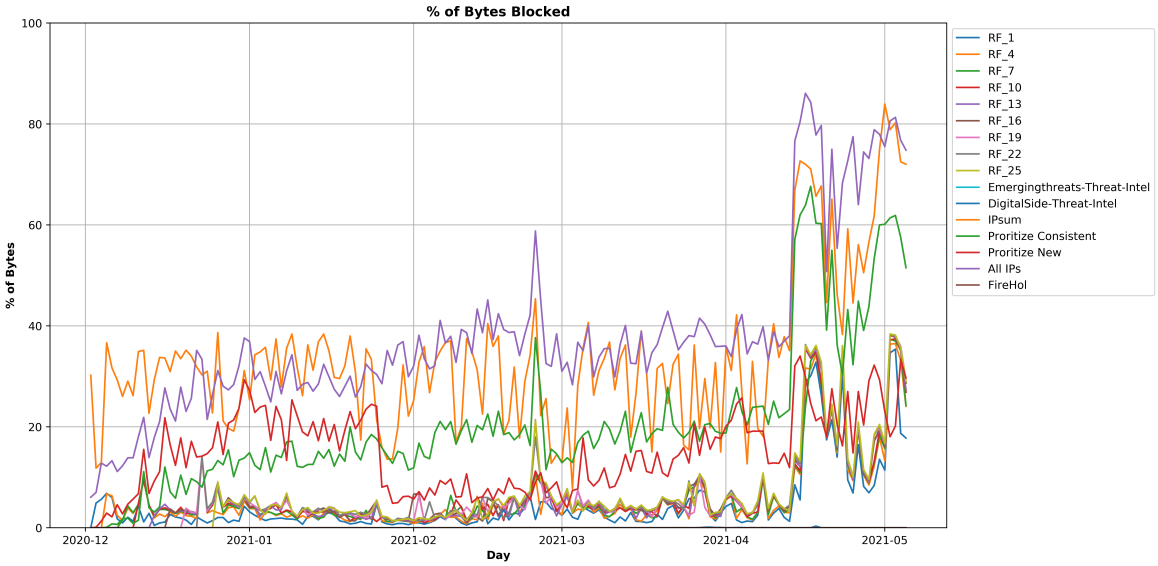

Figure 5.2 Comparison of the Percentage of Malicious Bytes Blocked in attacks. RF_X stands for Random Forest models trained up to X days in the past.

Experiment of Comparing the Duration of Attacks Blocked The following comparison between all the blacklists was regarding their total duration blocked and shown in Figure 5.3. Once again, the ALL IP blacklist is the best, followed by IPsum, the PC blacklists, and the PN blacklists. Also, here the FireHOL, DigitalSide-Threat Intel, and Emerging Threats are the lines that are close to zero at the bottom of the graph. It should be noted that the performances in this graph are much more volatile.

Experiment of Comparison of Amount of Flows Blocked The following comparison between all the blacklists was regarding their total flows blocked and can be seen in Figure 5.4. Once again, the ALL IP blacklist is the best, but the IPsum blacklist is surpassed by the PC and PN blacklists in some cases later in the evaluation. Also, here the FireHOL, DigitalSide-Threat Intel, and Emerging Threats are the lines that are close to zero at the bottom of the graph. The RF blacklists are not as tightly grouped in this case, suggesting that increasing the dataset size affects performance in this case.

Experiment of Comparing the Amount of IP Blocked The last comparison between all the blacklists was regarding their total number of IPs blocked and shown in Figure 5.5. This one is fascinating because even though the All IP blacklist is consistently the best, the IPsum blacklist is overtaken by the PC model and even the PN at specific points. Also, here the FireHOL, DigitalSide-Threat Intel, and Emerging Threats are the

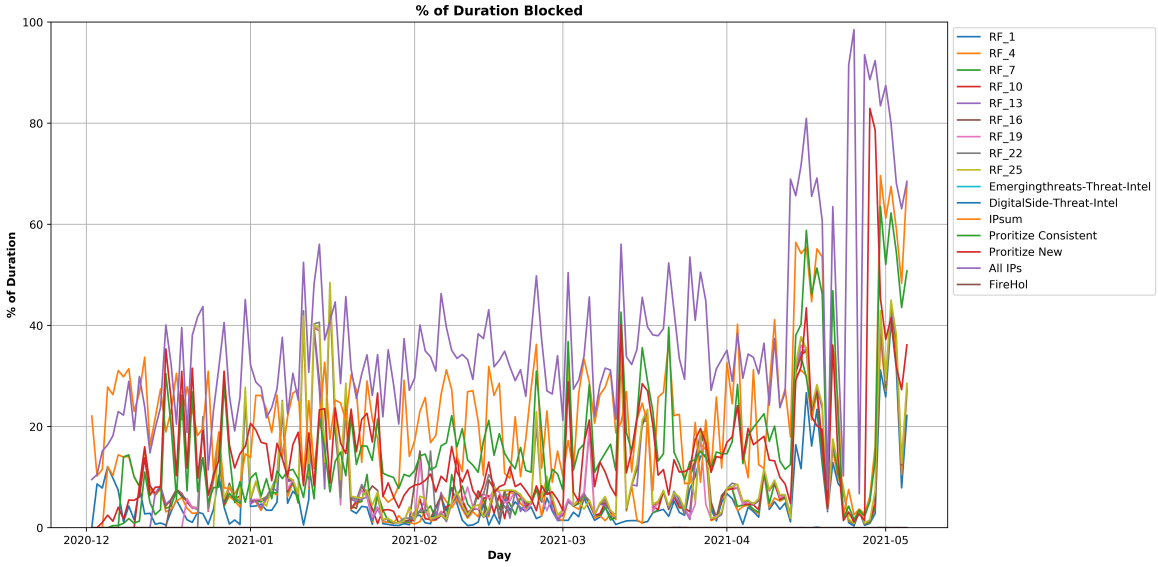

Figure 5.3 Comparison of the Percentage of Total Duration Blocked in attacks. RF_X stands for Random Forest models trained up to X days in the past.

lines that are close to zero at the bottom of the graph. The RF blacklists clearly improve performance as more data is used for training, clearly seen when comparing the RF_1 blacklist to RF_25.

We also calculated the average performance and standard deviation per blacklist, shown in Table 5.1, where the bold rows are the blacklists that performed the best on an absolute scale.

Performance Index After the previous comparisons with the four metrics, it was still unclear how to show the differences between blacklists. Since a larger blacklist blocks more IPs, how to express the unbalance between size and coverage?

Therefore, we created a performance index that is designed to show the efficiency of the blacklists in terms of how much *each* IP address contributes to the final blockage.

The performance index is calculated using the Equation 5.1, where d the is number of evaluation days, i is the number of day, $metric$ is the metric being indexed, and $blacklist\_size$ is the size of the blacklist each day. The idea is to compute the ratio of each average performance metric over the average size of that blacklist. This index is crucial for comparing blacklists of different sizes since it shows how well the model that generates the blacklist can choose the most critical IPs. Blacklists with a higher performance index have IPs that block more, fewer IPs that are never used for blocking, and are therefore more efficient. Although they might be less effective on an absolute scale, the performance index is helpful to pick up blacklists that are smaller but better, as is discussed in Chapter 6. The question the index helps to answer is: Which is the smallest blacklist that blocks the more IPs?

A comparison of the performance index for all the blacklists is shown in Table 5.2, where the AIP framework blacklists are in bold, along with the best performing downloaded blacklist, IPsum.

$$index = \frac{\sum_{i=1}^{d}(metric_i)}{\sum_{i=1}^{d}(blacklist\_size_i)} \tag{5.1}$$

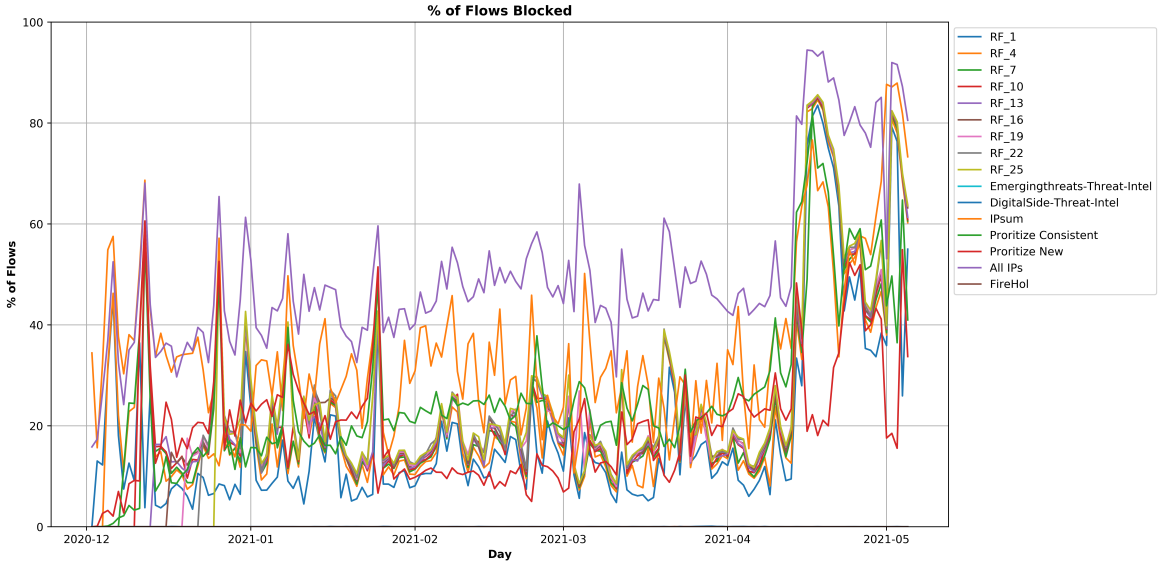

Figure 5.4 Comparison of the Percentage of Malicious Flows Blocked in attacks. RF_X stands for Random Forest models trained up to X days in the past.

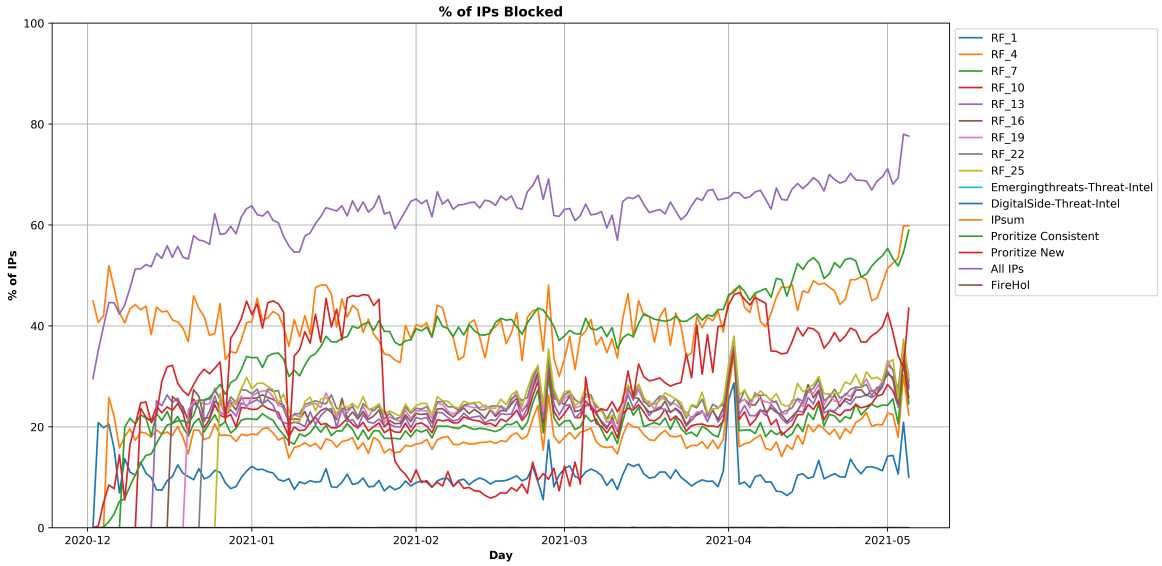

Figure 5.5 Comparison of the Percentage of Malicious IPs Blocked in attacks. RF_X stands for Random Forest models trained up to X days in the past.

Table 5.1 Average Percent Blocked per Blacklist per Metric

Blacklist	Bytes		Duration		Flows		IPs	
	AVG	STD	AVG	STD	AVG	STD	AVG	STD
RF_1	4.61%	6.89%	6.27%	8.31%	18.38%	17.87%	10.38%	3.19%
RF_4	5.87%	7.64%	8.8%	9.74%	22.56%	18.57%	17.85%	3.71%
RF_7	6.19%	7.63%	9.19%	9.93%	23.05%	18.17%	19.88%	4.8%
RF_10	6.39%	7.95%	9.2%	10.13%	23.17%	18.89%	20.91%	5.83%
RF_13	6.41%	8.26%	9.11%	10.45%	23.19%	18.9%	21.57%	6.83%
RF_16	6.65%	8.26%	9.54%	10.59%	23.29%	19.16%	21.82%	7.63%
RF_19	6.56%	8.36%	9.15%	10.84%	23.28%	19.37%	22.06%	8.45%
RF_22	6.65%	8.37%	9.47%	11.08%	23.09%	19.72%	22.02%	9.11%
RF_25	6.58%	8.48%	9.22%	11.08%	22.93%	20.01%	21.98%	9.83%
PC Model	**20.62%**	**14.71%**	**16.4%**	**13.07%**	**25.8%**	**16.02%**	**37.26%**	**12.14%**
PN Model	14.86%	7.8%	15.05%	12.09%	19.93%	11.53%	27.69%	13.71%
IPsum	**33.79%**	**15.03%**	**23.08%**	**13.33%**	**35.22%**	**15.95%**	**41.75%**	**5.19%**
All IP	**37.98%**	**17.11%**	**37.4%**	**17.99%**	**50.15%**	**16.24%**	**62.12%**	**6.83%**
DigitalSide	0.01%	0.03%	0%	0.01%	0.01%	0.02%	0.03%	0.02%
Emerging	0%	0%	0%	0%	0.001%	0.003%	0.002%	0.005%
FireHOL	0%	0.001%	0.001%	0.006%	0%	0.003%	0%	0.002%

Table 5.2 Performance Index Per Blacklist (Higher is Better)

Blacklist	Index Value (Higher is Better)			
	Bytes	Duration	Flows	IPs
RF_1	**0.003**	**0.0041**	**0.012**	**0.0068**
RF_4	0.0017	0.0025	0.0065	0.0051
RF_7	0.0013	0.002	0.0049	0.0042
RF_10	0.0011	0.0016	0.004	0.0036
RF_13	0.0009	0.0013	0.0034	0.0032
RF_16	0.0009	0.0012	0.003	0.0029
RF_19	0.0008	0.0011	0.0027	0.0026
RF_22	0.0007	0.001	0.0025	0.0024
RF_25	0.0007	0.0009	0.0023	0.0022
PC Model	**0.0003**	**0.0002**	**0.0003**	**0.0005**
PN Model	0.0003	0.0003	0.0003	0.0005
IPsum	**0.0003**	**0.0002**	**0.0003**	**0.0003**
All IP	0.0001	0.0001	0.0002	0.0002
DigitalSide	5e-06	2e-06	8e-06	1.9e-05
Emerging_threats	4.95e-08	2.4e-08	4.387e-07	1.268e-06
FireHOL	4.73e-08	2.492e-07	1.357e-07	8.95e-08

Chapter 6

Results and Analysis

This chapter discusses and analyses the results of the experiments presented in Chapter 5. We discuss how each of the AIP models performs against the third-party open-source threat intelligence feeds, and the ALL-IP blacklist discussed.

6.1 Results of Blacklists Size Comparison

The first comparison between all the blacklists was regarding their size and can be seen in Figure 5.1. The All-IP blacklist (light violet) grows in size at a linear rate, which is expected. After five months, the All-IP blacklist has almost 1,000,000 IPs and keeps growing. The second largest blacklist is IPsum (orange); however, this list has a constant size, which shows that their creators must have a mechanism to forget IP addresses, but we do not know which is it. The PN blacklist (red) has a significant variance which suggests that the rate of appearance of new attackers varies greatly. The PC blacklist (dark green) slowly grows from zero to the size of IPsum, which suggests that we might want to limit its size or change the score threshold in a future version. The RF lists grow in an exponentially decreasing order, which means that adding one more day to its training does not add as many IPs as the first days that were added. All third-party blacklists seem to have a limit in the total amount of IP used. This is in clear contrast with our method, which does not limit the amount of IPs directly, but limits the score, which is a measure of importance.

6.2 Results of the Metrics Comparison

6.2.1 Results of Blacklists Bytes Comparison

The following comparison between all the blacklists was regarding their bytes blocked and shown in Figure 5.2. Note that for this section, performance is discussed on an absolute scale. The IPsum blacklist(orange) seems to have much variance in performance for this metric, which is interesting compared to the PC blacklist (green), which slowly climbs to about the same performance but with much less variability. When comparing these performances to their blacklist sizes in Figure 5.1, we can see that the PC model catches up to the IPsum blacklist in performance around the same time it grows to the same size as it. This suggests that the IPsum and PC blacklists perform about the same in the bytes metric.

The PN blacklist (red) also varies less than IPsum but has large, infrequent jumps in performance. When comparing these jumps to the size graph in Figure 5.1, these drops and gains in the performance line up with massive drops and gains in blacklist size. This suggests that the PN model aging needs to be adjusted in future versions of the framework.

The eight RF blacklists do not change much in performance between the different training data sizes, suggesting that the Random Forest Classifier did not find the bytes metric to be essential to determining the label of a given prediction IP. These blacklists spike in performance near the end of the evaluation like the others, suggesting much overlap between the Random Forest classifications and the other blacklist choices.

6.2.2 Results of Blacklists Duration Comparison

The next comparison between all the blacklists was regarding their duration blocked and shown in Figure 5.3. All of the blacklists seem to have a lot more variance than in the bytes comparison. Note that for this section, performance is discussed on an absolute scale.

Note that in terms of absolute performance, even the baseline blacklist, the All IP blacklist (light violet), seldom rises above 40% blocking. This suggests that the IPs that are entirely new and cannot be predicted by any of the blacklists are responsible for most of the connection duration. This is supported by the fact that the rest of the blacklists seldom rose above 20% blocking.

The IPsum blacklist (orange) seems to have much variance in performance for this metric, which is interesting compared to the PC blacklist (green), which slowly climbs to about the same performance but with less variability. The PN blacklist acts similarly to its performance in the bytes comparison, only this time with much more variance. The RF blacklists all perform very similarly to each other, suggesting that the Random Forest Classifier did not find the duration metric crucial to determining a given prediction IP label.

6.2.3 Results of Blacklists Flows Comparison

The next comparison between all the blacklists was regarding their flows blocked and can be seen in Figure 5.4. Note that the vertical lines in the graphs of the RF blacklists occur because these blacklists need to wait until they have enough prediction data. Also, note that for this section, performance is discussed on an absolute scale.

The All IP blacklist performs much better than any other blacklists in this metric by a much larger scale than in the previous two metrics. As was explained in Section 3.5, the main difference between the All IP blacklist and the other blacklists is that the other blacklists are forgetting specific IPs as time goes on, while the All IP is not. The fact that the All IP blacklist performs so much better than the other blacklists in this metric suggests that the IPs that are being forgotten by the other blacklists are responsible for a substantial number of connections. This might need to be addressed in a future version of the AIP framework.

The IPsum and PC models performed more closely in this metric than in the other two metrics discussed, but with the PC being more consistent as before. Whereas in the other metrics discussed, it took the PC blacklist at least four months to catch up with the IPsum blacklist in performance, this metric caught up in only two months, and they

follow each other closely for the rest of the evaluation period. When comparing this to the size graph in Figure 5.1, these two start performing similarly while the PC blacklist is still substantially smaller than IPsum.

The PN blacklist is has variability in this metric, and unlike the other metrics, it does not seem to correlate with its size. Sometimes it performs better than the PC model, and at others, it performed worse than the Random Forest 1 Day model, which is surprising considering that it was almost an order of magnitude larger. Also, it massively loses performance near the end while all the other blacklists are gaining.

The Random Forest Blacklists seem to follow a similar trajectory as the PN model, with minor variance. This is an excellent sign considering that the RF blacklists are, in fact, the smallest ones, and despite that, can block a large amount of traffic.

6.2.4 Results of Blacklists IPs Comparison

The last comparison between all the blacklists was regarding the number of IPs blocked and can be seen in Figure 5.5. Note that the vertical lines in the graphs of the RF blacklists occur because these blacklists need to wait until they have enough prediction data. Also, note that for this section, performance is discussed on an absolute scale. All blacklists seem to perform at a much more steady rate than in any other metric.

The All-IP blacklist consistently has a 10% performance uplift over the PC and IPsum blacklists. In this case, however, the PC blacklist even more clearly catches up with the IPsum blacklist within the first two months of the evaluation and performs better by the end.

The PN blacklist acts similarly as it did in the bytes metric, having large, infrequent jumps in performance. When comparing the jumps to the size graph in Figure 5.1, these drops and gains in the performance line up with massive drops and gains in blacklist size. This is very interesting compared to the PC and IPsum blacklists, especially since it performed better than either of them for a time before dropping below the performance of the RF_1 blacklist. This suggests that an alteration might need to be made in the aging method in the PN model.

The RF blacklists seem to gain performance at a logarithmic rate in proportion to the amount of data they predicted on. The RF_1 blacklist was the lowest and the RF_25 the highest, but the most significant difference in performance was between RF_1 and RF_4. Note that all of the RF blacklists had a high performance in this metric at the very beginning of the evaluations, and then it dropped down and settled at about 50% its previous value after a few days. The difference here is in the amount of data the RF classifier is training on. This amount of data, as explained in Section 3.4, starts with just the most recent data on day 1 of training and then grows up to 30 days of data in the main database. This suggests that a future RF model should be tested that only trains on the most recent data to see if this high performance continues over time.

6.3 Results of Blacklists Performance Index Comparison

In order to compare the blacklists in a standardized framework, we computed a table of averages and standard deviation, shown in Table 5.1, and then using this to compute a table of performance indexes shown in Table 5.2. As was explained in Chapter 5, the

performance index is calculated using the Equation 5.1. The idea is to compute the ratio of each average performance metric over the total size of the blacklist. This index is crucial for comparing blacklists of different sizes since it shows how well the model that generates the blacklist can choose the most important IPs. Therefore, for a blacklist with a large Performance Index, each IP in the blacklist blocked *more* and consequently there are less IPs that are not used. Thus, models that have higher performance indexes are more efficient when choosing what IPs to block. This can be very useful when trying to find small blacklists that have the most impact, our goal for IoT devices.

As can be seen in Table 5.2, the RF blacklists perform the best in this regard. Even the worst RF blacklist, the RF_25, is still more than twice as good as the PC, PN, IPsum blacklists for the Bytes and Duration metrics, and an order of magnitude better in the case of the Flows and IPs metric. The best performing RF blacklist is the RF_1 blacklist, a whole order of magnitude better than the PC, PN, and IPsum blacklists in bytes and duration, and two orders of magnitude better in the flows and IPs metrics. This shows that the RF models are an excellent resource for producing small and practical blacklists for IoT.

6.4 About the Emerging Threats, DigitalSide and FireHOL blacklists

According to our experiments and as shown in Figures 5.2, 5.4, 5.5 and 5.3, the Emerging Threats, DigitalSide and FireHOL blacklists seem to have a poor performance in new unseen attacks. The percentage of traffic being blocked was either zero or very close to zero in all four of the metrics.

This lack of performance could be due to the blacklists not including the correct IPs, or it could be that the blacklists were generated for other geographical locations. It is suspected that IPs may attack differently in different parts of the world. In order to evaluate which is the valid reason for their poor performance, we reevaluated all the blacklists in another dataset called Hornet 15 [28]. Hornet 15 is a dataset of honeypots attacks captured for 15 days in eight different locations. The idea was to evaluate how these same blacklists (including our AIP blacklists trained in data from Prague) performed on datasets from eight different locations in the world.

6.4.1 Location Bias Testing using the Hornet 15 Dataset

Hornet 15 is composed of two weeks of data, captured from 8 honeypots placed in eight different locations across the world, seen in Table 4.3. This dataset was processed, sorted, and evaluated with the same methodology presented in previous Section 4.2. We performed the same evaluation methodology explained in Chapter 3, the only difference being that the Hornet 15 dataset was used for the evaluation data.

Since our blacklists are trained in data from honeypots in Prague, and Emerging Threats, DigitalSide, IPsum, and FireHOL were trained in other locations, we wanted to perform a non-location biased evaluation. The models of the AIP framework, including the random forest, were trained in Prague and tested in the data from Hornet 15. This guaranteed a fair comparison with the other blacklists.

The results of these evaluations are shown in Figures 6.1, 6.4, 6.2 and 6.3. These graphs show the average performance of each of the blacklists per day from each of the

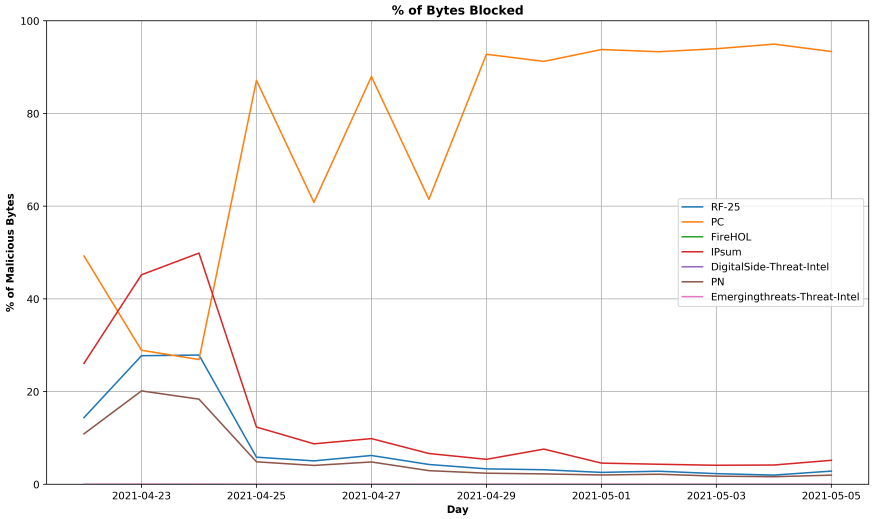

Figure 6.1 Average Bytes blocked per Blacklist, Hornet 15 Evaluation

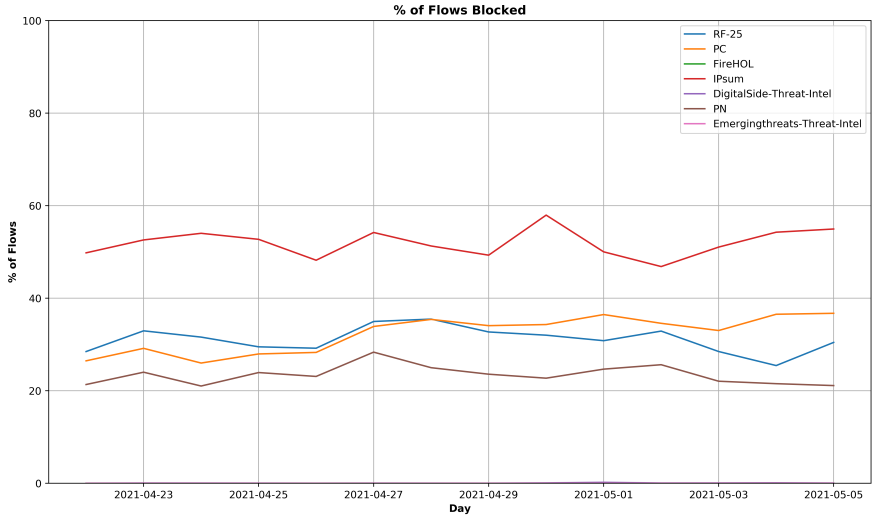

Figure 6.2 Average Flows blocked per Blacklist, Hornet 15 Evaluation

8 locations. As we can see, the results are very similar to the original evaluations using the dataset from Prague, and the performance of the Emerging Threats, DigitalSide, and FireHOL blacklists are the same.

This is a very interesting finding, especially if we compare the results to the Random Forest model that was trained on a single day of data, seen in Figures 5.2, 5.4, 5.5 and 5.3. This blacklist is about the same size as the three third-party blacklists, as can be seen in Figure 5.1, but it performed much better. This shows the importance of a standard of evaluation for blacklists. This second evaluation in eight different geographic locations confirms that the third-party blacklists Emerging Threats, DigitalSide, and FireHOL did not have poor performance due to a geographical location but because of how they were built.

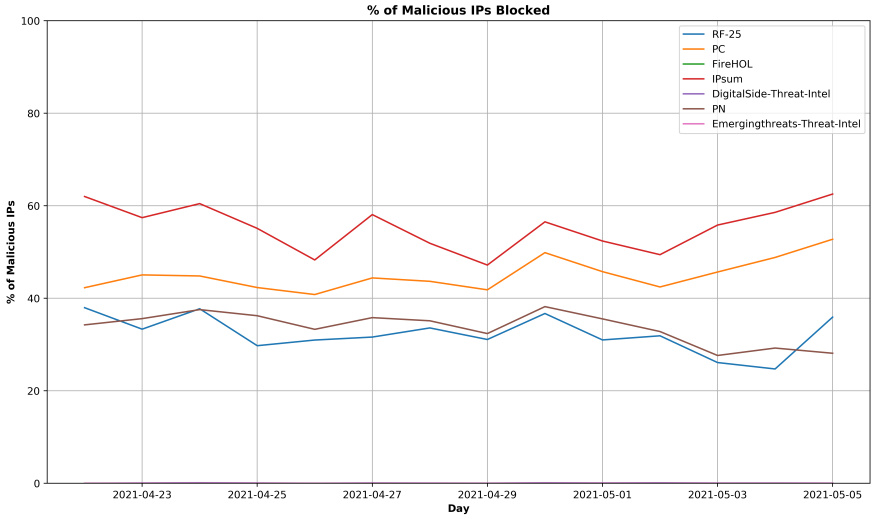

Figure 6.3 Average IPs blocked per Blacklist, Hornet 15 Evaluation

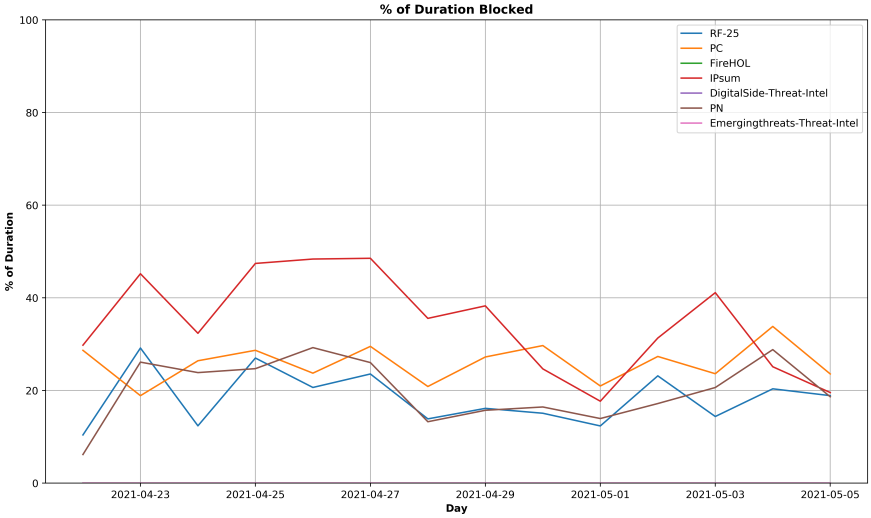

Figure 6.4 Average Duration blocked per Blacklist, Hornet 15 Evaluation

Chapter 7

Conclusion

Blacklists are the core defense method in our current Threat Intelligence feeds and IDS systems. However, there was no known evaluation and comparison of blacklists. The lack of an evaluation may imply that blacklists may not work as well as we believe [2]. In particular for IoT, where blacklists may play a crucial role, a better blacklists may improve their protection while consuming low resources.

This thesis presented an alternative framework for generating and evaluating blacklists, called Attacker IP Prioritizer (AIP). AIP consists of three algorithms that take network data and output blacklists. The first algorithm prioritizes IPs that attack more consistently, the second algorithm prioritizes IP that are newer. Both algorithms use a score per IP, that is later aged in order to forget unnecessary IP addresses. These algorithms aim at maximizing the amount of malicious traffic blocked while minimizing the size of the blacklist. In this way, the blacklists are easily deployable in devices with small memory, such as smart-doorbells [1].

The third algorithm implemented in the AIP framework is a Random Forest. This algorithm uses a concatenated database of labeled past data to train the model and then predicts which IPs seen in the past shall be blocked in the next day.

In order to create our blacklists and evaluate our methods, we created a specific dataset of real IoT attacks. This dataset was created by capturing traffic in the Aposemat IoT laboratory using real IoT devices for five months. These devices were exclusively used as honeypots, and the traffic was filtered such that every connection to them can be considered an attack. Another separate dataset, the Hornet 15 dataset by Veronica Valeros, was used to test any location bias in the evaluations.

The evaluation of our models was done by comparing them to our three blacklists, and a group of state-of-the-art free threat intelligence feeds found online. A particular blacklist was used as a baseline, a blacklist that remembers all attackers until now. The evaluation using the Hornet 15 dataset found that there was no geographically location bias in our results.

We found that the AIP models performed better than the downloaded open-source threat intelligence feeds according to the evaluation metrics. The first two models performed as well as other blacklists comparable in size, being able to block on average 37% of incoming malicious IP addresses. The Random forest models performed much better, blocking on average up to 22% of the incoming malicious IP addresses but being order of magnitudes smaller.

We calculated a performance index for each blacklist to compare blacklists of different size. We found that the Random Forest model has an Performance Index of of 0.0068.

This made it 13x better than the other AIP models and 22x times better than the IPsum blacklist, which was the best performing downloaded blacklist. The Random Forest model proved to be the best model for generating blacklists for IoT devices.

We conclude that better blacklists are possible and that blacklist evaluations need to be performed to establish better methodologies for blacklisting.

References

[1] David Beren. *The 8 Best Smart Doorbell Cameras of 2020*. Oct. 2019. URL: https://www.lifewire.com/best-smart-doorbell-cameras-4145064.

[2] Sûnnet Beskerming. *Time to blacklist blacklists*. July 2007. URL: http://www.beskerming.com/commentary/2007/07/01/196/Time%5C_to%5C_Blacklist%5C_Blacklists.

[3] Lars Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, pp. 108–122.

[4] Deepak Choudhary. "Security Challenges and Countermeasures for the Heterogeneity of IoT Applications". In: *Journal of Autonomous Intelligence* 1.2 (2019), pp. 16–22.

[5] Louis Columbus. *IoT Market Predicted To Double By 2021, Reaching $520B*. https://www.forbes.com/sites/louiscolumbus/2018/08/16/iot-market-predicted-to-double-by-2021-reaching-520b/. Forbes. 2018.

[6] Baris Coskun. "(Un) wisdom of Crowds: Accurately Spotting Malicious IP Clusters Using Not-So-Accurate IP Blacklists". In: *IEEE Transactions on Information Forensics and Security* 12.6 (2017). https://ieeexplore.ieee.org/abstract/document/7839928/, pp. 1406–1417.

[7] M. Cotton. "Procedures for the Management of the Service Name and Transport Protocol Port Number Registry". In: (2021). https://datatracker.ietf.org/doc/html/rfc6335.

[8] Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*. 1. Cambridge university press, 1997.

[9] "DigitalSide IPs". In: (2021). https://osint.digitalside.it/.

[10] Sebastian Garcia. "Aposemat Project - MALWARE ON IOT". In: (2021). https://www.stratosphereips.org/aposemat/.

[11] John A. Hawkinson and Tony J. Bates. *Guidelines for creation, selection, and registration of an Autonomous System (AS)*. RFC 1930. Mar. 1996. DOI: 10.17487/RFC1930. URL: https://rfc-editor.org/rfc/rfc1930.txt.

[12] Proofpoint Inc. "Proofpoint Emerging Threats Rules". In: (2021). https://rules.emergingthreats.net/fwrules/.

[13] V Jacobson, C Leres, and S McCanne. "libpcap, Lawrence Berkeley Laboratory, Berkeley, CA". In: *Initial public release June* (1994).

[14] Marc Kührer, Christian Rossow, and Thorsten Holz. "Paint it Black: Evaluating the Effectiveness of Malware Blacklists". In: (2014). https://christian-rossow.de/publications/blacklists-raid2014.pdf.

[15] Marc Kührer, Christian Rossow, and Thorsten Holz. "Paint it Black: Evaluating the Effectiveness of Malware Blacklists". In: (2014). https://christian-rossow.de/publications/blacklists-raid2014.pdf.

[16] Inc. MaxMind. "GeoIP Databases & Services: Industry Leading IP Intelligence". In: (2021). https://www.maxmind.com/en/geoip2-services-and-databases.

[17] "Open Argus". In: (2021). https://openargus.org/.

[18] The FireHOL Project. "FireHOL's IPsets". In: (2021). https://github.com/firehol/blocklist-ipsets.

[19] "Python". In: (2021). https://www.python.org/.

[20] Sivaramakrishnan Ramanthan, Jelena Mirkovic, and Minlan Yu. "Blacklists Assemble: Aggregating Blacklists for Accuracy". In: (2018). https://steel.isi.edu/members/sivaram/papers/blag_technical_report.pdf.

[21] "Logistic Regression". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I Webb. Boston, MA: Springer US, 2010, p. 631. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_493. URL: https://doi.org/10.1007/978-0-387-30164-8_493.

[22] Ankit Kumar Singh. "TCP flags". In: (2019). https://www.geeksforgeeks.org/tcp-flags/.

[23] Sushant Sinha, Michael Bailey, and Farnam Jahanian. *Improving Spam Blacklisting Through Dynamic Thresholding and Speculative Aggregation*. Proceedings of the Network and Distributed System Security Symposium, https://vhosts.eecs.umich.edu/fjgroup//pubs/ndss10_final.pdf. University of Michigan, Ann Arbor, MI. 2010.

[24] Miroslav Stampar. "IPsum Threat Intelligence Feed". In: (2021). https://github.com/stamparm/ipsum.

[25] Michael Bailey Sushant Sinha and Farnam Jahanian. *Shades of grey: On the effectiveness of reputation-based "blacklists"*. IEEE, https://ieeexplore.ieee.org/abstract/document/4690858. 2008 3rd International Conference on Malicious and Unwanted Software (MALWARE). 2008.

[26] Gilbert Tanner. "Random Forest". In: (2021). https://ml-explained.com/blog/random-forest-explained.

[27] Joe Touch et al. "Service Name and Transport Protocol Port Number Registry". In: (2021). https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml.

[28] Veronica Valeros. "Hornet 15: Network Dataset of Geographically Placed Honeypots". In: (2021). https://data.mendeley.com/datasets/rry7bhc2f2/2.

[29] Steve Weisman. "What is a distributed denial of service attack (DDoS) and what can you do about them?" In: (2020). https://us.norton.com/internetsecurity-emerging-threats-what-is-a-ddos-attack-30sectech-by-norton.html.

[30] Christian Wressnegger and Rouven Scholz. *Security Analysis of Devolo HomePlug Devices*. https://dl.acm.org/doi/10.1145/3301417.3312499. EuroSec '19: Proceedings of the 12th European Workshop on Systems Security. 2019.