One-class Classification to Detect Botnets in IoT devices *

Vitor Hugo Bezerra[1], **Victor G. Turrisi da Costa**[1],
Sylvio Barbon Junior[1], **Rodrigo Sanches Miani**[2], **Bruno Bogaz Zarpelão**[1]

[1]Computer Science Department – State University of Londrina (UEL)
Londrina – PR – Brazil

[2]School of Computer Science (FACOM) – Federal University of Uberlândia
Uberlândia – MG – Brazil

{vitorbezerra, victorturrisi, barbon, brunozarpelao}@uel.br,

miani@ufu.br

Abstract. *With the increasing number of different Internet of Things devices, new threats to network security emerge due to these devices' low security. Botnets are a widespread threat that takes advantage of IoT devices vulnerabilities to compromise multiple devices and perform coordinated attacks. To tackle this, new methods addressing IoT botnets detection are required. In this paper, we propose a host-based detection system based on one-class classifiers. It was used a One-class Support Vector Machine built with features such as CPU and memory usage to detect malicious activities. The predictive performance and resource consumption of the proposed approach was evaluated in a controlled network using three different legitimate settings and seven IoT botnets. The results indicate that the proposed system is efficient in detecting different botnets with low resource consumption.*

1. Introduction

Internet of Things (IoT) devices can be found in our everyday life in many situations, e.g., surveillance cameras, healthcare monitors and traffic monitoring services. The IoT paradigm makes machine-to-machine communication over the Internet more practical, connecting more devices online and allowing them to actively participate in the network [Whitmore et al. 2015]. IoT environments usually consist of many heterogeneous and low-cost devices with little or no security embedded into them, which generate a vast amount of private information, and may create many security problems. This open wound in IoT security is likely to prevail for years to come and must not be ignored given the broad range of applications of the IoT paradigm [Angrishi 2017].

This increase in the number of deployed IoT devices has awakened the attention of malicious users, who target those devices to gather computation power and carry out illegal activities. Those users can, for example, perform Distributed Denial of Service (DDoS) attacks by creating large-scale IoT-based botnets [Angrishi 2017]. Moreover, compromised devices may not demonstrate any apparent symptoms of infection, being able to continue with the execution of their normal activities. Therefore, detecting compromised devices is a challenging subject and requires specialised tools [Kolias et al. 2017].

*The authors would like to thank CAPES for financial support.

One of the many threats IoT devices face is botnets. A botnet is a collection of compromised devices, referred to as bots, controlled by one or more malicious users, which communicate with the bots to perform malicious activities. IoT botnets have already been making a huge impact in our lives, having Mirai attack in 2016[1] as the most remarkable example so far. In this attack, a botnet called Mirai infected surveillance cameras - a type of IoT device - by taking advantage of their default security settings and performed a large-scale DDoS attack against Dyn, a major DNS service provider [Mansfield-Devine 2016].

With those constant threats being developed and the lack of versatile tools to update IoT devices, users can not rely on updates for every security breach found in these devices. This scenario makes an intrusion detection system (IDS) fundamental for these devices [Bertino and Islam 2017]. Although IDS for IoT devices are being developed by the research community [Raza et al. 2013, Amaral et al. 2014], they still present some issues, such as, being developed for specific network protocols in simulated devices (e.g., 6LoWPAN in Contiki operating system), relying on labelled data, exploring only network data and not being validated with newer botnets.

In this paper, we propose a host-based detection system for IoT devices using one-class classifiers. One-class classifiers are a different type of machine learning (ML) technique, which instead of classifying an instance in one of multiple pre-defined patterns, model a single pattern and use it to discern if a new instance belongs to the pattern or not. This approach is useful in detecting anomalies in data or machine faults, for instance. IoT devices have a specialised behaviour, performing simple tasks with a well-defined use of computational resources. We believe that modelling the device resources usage by a one-class classifier can support the detection of a botnet infection as an anomaly in its behaviour. The main advantage of this proposal is to build this model without needing a specialist to label the data or having access to data on compromised behaviour.

The One-class Support Vector Machine (OSVM), which is an SVM adaptation for the one class scenario, was used in the proposed approach. First, the model for legitimate resources consumption is built on a remote server. After that, the built model is deployed in the IoT device and the system starts analysing its resources consumption to detect behaviour deviations. In our tests, the proposed approach presented a great performance in detecting botnets, considering three different device settings and seven botnets, whilst it kept a low resource consumption.

The main contributions of this paper can be summarised as follows:

- We propose a lightweight approach that uses the One-class SVM to detect IoT botnets using data about the device resources;
- The OSVM had high predictive performance using only CPU and memory usage, electric potential difference, number of running tasks and CPU temperature;
- The approach is capable of protecting IoT devices from botnets without interfering with the devices' functionality;
- We evaluate the approach on a real IoT device, using seven different botnets;

This paper is organised as follows: in Section 2, we describe prior work done in IDS for IoT. Section 3 contains background information about traditional machine learning and one-class classification. In Section 4, the description of the proposed approach

[1] http://money.cnn.com/2016/10/22/technology/cyberattack-dyn-ddos/index.html

is presented. Section 5 describes the experimental environment constructed, and, in Section 6, we present the results and discussion of the proposed approach. Lastly, Section 7 provides the conclusion and future work.

2. Related Work

Intrusion Detection System (IDS) is a program or mechanism that focuses on the detection of attacks against a system or a network by analysing features in the network or in the host itself [Raza et al. 2013]. These systems can be divided into two major classes: network-based IDS (NIDS), which is dedicated to monitor network traffic to detect attacks against various elements in a network, and host-based IDS (HIDS), which is positioned on an individual host and is dedicated to detect attacks against only that host.

Most of the works on IDS for IoT are based on simulated networks using the protocol 6LoWPAN [Zarpelão et al. 2017]. The two most common operating systems deployed at these 6LoWPAN-based devices are Contiki and TinyOS. Although these works have promising results, exploring host-based, network-based and hybrid IDS, their focus on 6LoWPAN networks may prevent their use in networks based on other IoT protocols. 6LoWPAN is an IETF (Internet Engineering Task Force) standard for low power and lossy networks (LLN). However, other protocols are often used in IoT networks, such as Bluetooth Low Energy, and the conventional TCP/IP stack.

One of the most mature IDS for 6LoWPAN networks is named SVELTE [Raza et al. 2013]. It is composed of several modules to protect devices from different attacks. The first module of the system uses the LLN routing protocol RPL (Routing Protocol for Low power and Lossy Networks) to map the network topology and discover the nearest nodes to IDS agents that are distributed in the network. Then, the distributed IDS agents check the packets of their neighbours. This process aims to protect the devices from insider attacks. Another module is a distributed firewall deployed on the devices, which protects them from an outsider attack, closing ports and blocking addresses. SVELTE was tested in a network of Contiki-based devices.

In [Amaral et al. 2014], an IDS is developed for TinyOS. The proposed IDS has some watchdogs devices to analyse the packets send by their neighbours. This allows the system to detect traffic-based attacks and abnormal behaviour on the network. The IDS also enables the creation of rules and countermeasures that must be executed by the devices upon the detection of an intrusion. A limitation of their approach is that the rules may not have the needed complexity for responding accurately to a network attack, and the communication between the devices and the management module is insecure.

Other works in IDS for IoT focus on conventional TCP/IP stack or use host features, such as syscalls. Most of these works are NIDS intended to be a module in a router or gateway, using whitelists, machine learning methods or pre-defined rules. In [Habibi et al. 2017], it is proposed an IDS dubbed Heimdall. This IDS uses a whitelist to prevent IoT devices from connecting to malicious addresses and avoid communications with botnets C&C or private data leaks. Heimdall is intended to be a module in the router, acting as a gateway for IoT devices, using third-party analysis of malicious addresses from organizations such as VirusTotal, Metadefender, and VirScan, combined with an auditor and DNS validation. They tested this approach with a large number of real devices and the method is effective against the attacks developed by the researchers and has minimal overhead. However, tests with real botnet attacks were not carried out, and the IDS depends on the maintenance of the third-party systems.

Other works such as [Meidan et al. 2018] and [An et al. 2017] aim to build anomaly-based systems to detect IoT botnets. These works present techniques that model the legitimate behaviour of IoT devices. This type of approach can be effective due to the systematic behaviour of IoT devices, which usually perform specialised tasks. Thus, behaviour deviations are detected as malicious, regardless of the botnet.

Meidan et al. [2018] proposed the N-BaIoT, which is a NIDS that uses deep autoencoders to detect botnets in IoT devices. They use the network traffic of IoT devices to build a model of the legitimate behaviour and detect any anomaly. They tested the approach on two botnets, Mirai and BashLite, and without the use of labelled data, they achieved great results in detecting the attacks, with low false positives rate. Despite the great results, the use of deep autoencoders can be computationally costly even to a gateway and demands large amounts of data to train the model. The work proposed by [An et al. 2017] explores the construction of an IDS to protect Linux routers, a very popular target in recent years. They tested three different types of anomaly detection techniques, Principal Component Analysis (PCA), One-class SVM and a naive detector using n-grams, to analyse syscall data from routers. They used two botnets to test their approach, MrBlack and Mirai, in simulated routers, and all tested methods presented good results. However, it can be difficult to find the best features among syscalls during data pre-processing.

Overall, different issues were found in these works such as the lack of testing in real devices or with different botnet samples, dependency on third-party systems, use of computationally costly methods, and need for large amounts of data to train the models. Also, the analysed works did not propose host-based methods and did not explore resource consumption data to detect botnets. Considering these issues, we proposed an approach for botnet detection in IoT devices that does not need malicious labelled data to build a detection model and relies on host-based data such as CPU and memory usage. Additionally, it is a lightweight approach that does not interfere in the device operation and can be used in IoT systems based on traditional TCP/IP stack.

3. One-Class Classification

In traditional classification ML techniques, previously collected data associated to a class label is used to induce a model capable of representing and detecting each class. This requirement for labelled data from each possible class can be problematic in some cases, e.g., when labelling cost is too high or some class occurs much less frequently [Khan and Madden 2009]. When considering botnet detection, labelled data may be collected by a specifically designed testbed or honeypot. This requires manual work to set up vulnerable devices, and, likewise, some kind of access to botnets. Additionally, there is an inherited class unbalance problem when dealing with malware detection, where legitimate data is easily available and in far greater scale.

An emerging alternative is to use one-class classification techniques, which do not model all classes. Instead, data from a class of interest is used to create a model capable of yielding if a given new instance of data belongs to this class or not [Khan and Madden 2009]. In the context of botnet detection, after collecting legitimate behaviour data (which is more straightforward to obtain than malicious data), one-class classification algorithms induce a model that discerns if a given new instance is legitimate or not. Here, we considered that when an instance does not belong to the induced model, it is associated with malicious behaviour. In this sense, one-class classifiers can be used

as an alternative to traditional classification algorithms to reduce significantly labelling cost and the need for malicious data in the training phase. Likewise, they are easier to adapt, given that periodical model updates using new legitimate data demand less effort than updates with labelled legitimate and malicious data.

Multiple one-class classification techniques have been developed along the years, with the One-class Support Vector Machine (OSVM) [Sokolova and Lapalme 2009] being successfully used in multiple domains [Khan and Madden 2009]. Support Vector Machine (SVM) is a traditional ML algorithm used for classification problems. It works by creating a hyperplane that better separates two different classes. On the other hand, the OSVM uses hyperplanes to create boundaries around a region that better contains all training data. By doing so, the OSVM is capable of identifying if an instance is inside the area or not. To create these hyperplanes, the OSVM optimises the following objective function:

$$\min_{w,\xi,\rho} \frac{1}{2} w^T w + \frac{1}{\nu N} \sum_{i=1}^{N} \xi i - \rho \tag{1}$$

subjected to $w^T \phi(x_i) \geq \rho - \xi_i$ and $\xi_i \geq 0$, where $x_i \in \mathbb{R}^p$ is a sample of the training set, w is the weight vector of the hyperplane in the inner product space, $\phi(.)$ is a mapping from the original p-dimensional feature space to a inner product space, ξ_i's are penalty terms for error, ρ is a bias term, and $\nu \in (0, 1]$ is a parameter that poses an upper bound on the fraction of outliers in the training set.

The decision hyperplane can be represented by $g(x) \equiv w^T \phi(x) - \rho = 0$. The solution of the convex optimisation problem of the objective function finds the decision hyperplane, and the corresponding decision function $f(x)$:

$$f(x) \begin{cases} w^T \phi(x) - \rho \geq 0 & \text{if } x \text{ belongs to the set} \\ w^T \phi(x) - \rho < 0 & \text{if } x \text{ is an outlier} \end{cases} \tag{2}$$

Here, we used the radial basis function (RBF) kernel to map the input space to a higher dimensional space. The RBF kernel is described by the following equation:

$$\phi(x) = \exp(-\gamma ||x - x'||^2), \text{where } \gamma > 0. \tag{3}$$

Nonetheless, other kernel functions could be used, such as a polynomial or exponential function.

4. Proposed Approach

In this section, we present our proposed approach to detect botnets in IoT devices. Most of these devices are very specialised, running simple, repetitive and well-defined tasks, these devices behaviour should be the same as long as they do not become compromised. Any malicious software that compromises any IoT device should alter its behaviour significantly. In this sense, by using the OSVM to induce a model for the legitimate behaviour data, we can get rid of the effort demanded by the collection and labelling of malicious data. An overview of the approach is presented in Figure 1. The approach is divided into two phases: Model Induction and Continuous Analysis.

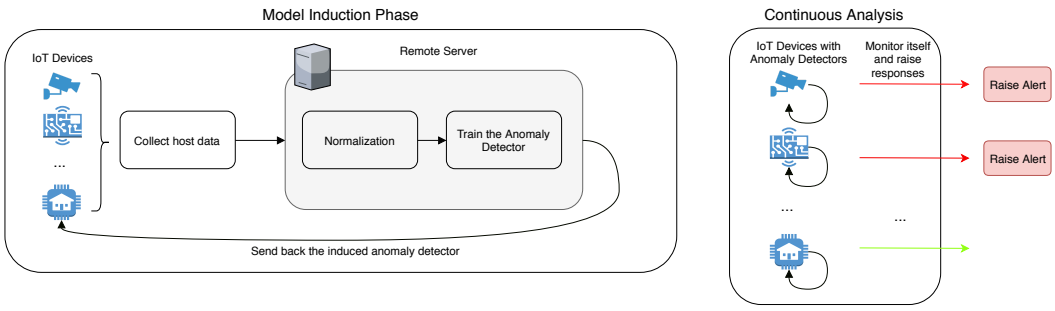

Figure 1. Description of the proposed approach processes.

The first step consists of the process of gathering host data. At this step, a network administrator feeds a remote server with samples of legitimate data collected from IoT devices of interest. This data comprehend to the CPU and memory usage of the device, the electric potential difference, the number of concurrently running tasks in the system and the CPU temperature. Then, in the remote server, host-data is grouped in instances according to a j seconds time window and undergo equation for a normalisation step. All features for each instance are scaled according to their minimum and maximum values between a range from 0 to 1, defined by the following equation:

$$x_i[j] = \frac{x_i[j] - min(x[j])}{max(x[j]) - min(x[j])}, \forall i \in I, \forall j \in J \tag{4}$$

where i corresponds to a given instance, I to all instances collected during that period, j to a feature in the J feature space, and $x_i[j]$ to the value present in feature j for the instance i.

With instances normalised and separated in time windows, the server trains the OSVM using the legitimate data. This is all done in the server since the OSVM optimisation process is too computationally demanding for an IoT device. Nonetheless, after trained, the resulting function $f(x)$, described in Equation (2), uses very few resources from the IoT device to define whether a new instance belongs to the legitimate class or not. Finally, the induced model is deployed to the IoT devices.

Lastly, the Continuous Analysis phase consists of using the induced model to monitor the device. Each device captures its data in intervals of x seconds, generating a new instance. It is worth noticing that x should not be too small, so that this monitoring may impact the device performance, nor should it be too large, that botnet detection is delayed. This new instance undergoes the same normalisation step described in Equation (4), except that this time, the $max(x[j])$ and $min(x[j])$ are the same as the ones used when the OSVM was induced. After that, by applying the function in Equation (2), the approach decides if this new instance is a legitimate behaviour or an anomaly. If it is the latter, the device raises an alert.

5. Evaluation

In this section, we describe the experimental environment to test our proposed IDS and the metrics to evaluate the efficiency of the approach.

5.1. Experimental environment

To test the proposed approach we build an experimental environment as presented in Figure 2. This environment allows to use different botnets not used in previous works and capture host-based data. The network components consist of a switch with Ethernet and Wi-Fi interfaces, and four computers (*Machine 1*, *Machine 2*, *Machine 3*, and a *Gateway*) connected to the switch through Ethernet. There is also a Raspberry Pi model 3B connected to the switch through the Wi-Fi interface. *Machine 2* provided two virtual machines, with the first one hosting a Web Server and the second being a client, referred to as *Device Admin*.

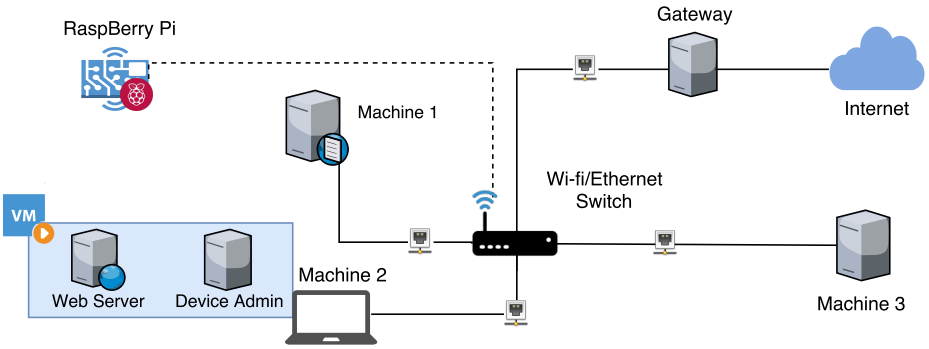

Figure 2. Experimental environment network topology.

The Raspberry Pi emulates three different behaviour profiles: two focused on security cameras and one focus on a multimedia device.

- **Multimedia Centre (MC)**: found in today's living rooms, a multimedia centre is a device that consumes streams of video such as movies or TV shows, to transform regular TVs in smart TVs. In addition to the use of video content, it also accesses an application store for updates and installation of other programs. The task of rendering video combined with the heavy traffic generated by video streams make this device to use a significant portion of its resources during its operation;
- **Surveillance Camera with Additional Traffic (ST)**: used both on the inside and outside parts of houses and companies, the operation of surveillance cameras is mostly characterised by the transmission of a video stream to a computer or station. However, this kind of device can also present traffic from other protocols. Telnet or SSH connections can be established to check if the device is working, its temperature, and the running processes, for example. These cameras may also include configuration web pages, which users access from their browsers to set up the parameters of cameras operation. As the device do not have to render video before transmitting, it consumes fewer resources than the multimedia centre;
- **Surveillance Camera (SC)**: this profile is similar to the ST profile, but it does not include interactions with users through Telnet, SSH, and configuration web pages. Having the video transmission as their single task, devices of this profile consume fewer resources than those of ST profile.

The *Gateway* was used to provide Internet access and DHCP functionality to the network. *Machine 1* hosted a DNS server that was used by the Mirai botnet. The same machine also hosted a VLC client that consumed the video stream generated by SC/ST profiles and a VLC server that generated a video stream for the MC profile. *Machine*

2 hosted the *Web Server* and the *Device Admin*. The *Web Server* was responsible for providing a static Web page, which was accessed by the *Raspberry Pi* in all the profiles to emulate the consumption of web services. The *Device Admin* emulated the transactions of an administration software used to control and set up the emulated camera in the ST profile. To do so, it interacted with the *Raspberry Pi* through Telnet, SSH, and the Web configuration page. Lastly, *Machine 3* was responsible for emulating an attacker on the network, which infected IoT devices with botnet samples. Particularly for Mirai, this machine also hosted a C&C server, which could be used to make the *Raspberry Pi* launch DoS attacks against selected targets.

We executed the following botnets in our experimental network: Mirai and Bashlite, the most famous IoT botnets that uses Telnet [Angrishi 2017]; Hajime, a botnet of unknown purpose that patches vulnerabilities in IoT devices [Stavrou et al. 2017]; Aidra, Tsunami and Dofloo, very famous botnets that moved from personal computers domain to IoT devices using ARM processors [Abdul Kadir et al. 2015].

With those botnets, we have three types of infection, as presented in Table 1. The legitimate data was captured following the description of each behaviour. The MC profile received video streams from YouTube, Twitch and HD video. The SC device streamed video to another computer. The ST device streamed video, was accessed via SSH and Telnet and performed updates in programs. We captured one hour of legitimate data from each device, and captured one hour of each infection on each device, erasing the data in the device after each capture to prevent contamination from the previous botnet. We used the *top* program to gather memory usage data. To capture the temperature of CPU and the electric potential difference, we used the *vcgencmd* program present in the Raspberry Pi.

Table 1. List of infections made in the Raspberry Pi profiles.

Infection Type	Profile	Botnet(s)	Method of Infection	Number of Samples
1	MC	Hajime	SSH	4
	SC	Aidra		11
	ST	BashLite		
2	MC	Mirai	SSH	1
	SC			
	ST			
3	MC	Mirai, Doflo, Tsunami,Wroba		28
	SC			
	ST			

5.2. Evaluation Metrics

The metrics used to evaluate the proposed IDS are [Sokolova and Lapalme 2009]:

- **Accuracy**: $\frac{TP+TN}{TP+FN+FP+TN}$ the overall effectiveness of the approach;
- **AUC**: $\frac{1}{2}(\frac{TP}{TP+FN} + \frac{TN}{TN+FP})$ ability of the approach to avoid false classification;
- **F1-score**: $\frac{2}{\frac{1}{recall}+\frac{1}{precision}}$ relation between recall and precision;
- **Precision**: $\frac{TP}{TP+FP}$ the percentage of classified botnets instances that are truly botnets;
- **Recall**: $\frac{TP}{TP+FN}$ effectiveness of the approach in identifying botnets;
- **Specificity**: $\frac{TN}{FP+TN}$ how effective is the approach in identifying instances that are legitimate.

TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives, respectively.

5.3. Experiment setup

To evaluate our proposed approach, we used nine datasets, which are composed of one hour of legitimate traffic and one hour of malicious traffic each one. Each dataset comprehends to a different profile and infection type combination. We organised the evaluation in three phases: finding an optimal time window size, considering that smaller windows will result in faster identification of botnets; discovering the least number of instances needed for training; and optimising the OSVM's hyperparameters. All tests were performed using the holdout with 50 repetitions to minimise variance in the results. Lastly, we used the OSVM implementation from the scikit-learning library in Python [2].

6. Results and discussion

In this section, we discuss the experimental results of our proposed approach. First, we used the default hyperparameters for the OSVM ($\nu = 0.1$ and $\gamma = 0.1$) to analyse the impact on the predictive performance of changing the window size. In Table 2 we present the accuracy, precision, recall, F1, specificity, and AUC. It is possible to see that by reducing the window size, predictive performance increased. Although precision is around 70%, the other metrics present very satisfactory results, considering that no botnet data was used during training. Additionally, by choosing the 1 second time window, botnets would take less time to be detected. Therefore, considering the time window with the best performance and a bias towards smaller windows, we chose the 1 second time window.

Table 2. Evaluation of time window size in all datasets.

Time window size (s)	Accuracy	Precision	Recall	F1-score	Specificity	AUC
1	0.9112	0.7054	0.9553	0.8111	0.9000	0.9277
5	0.9116	0.7037	0.9675	0.8143	0.8976	0.9325
10	0.8988	0.6883	0.8968	0.7769	0.8993	0.8981
30	0.8989	0.6822	0.8974	0.7712	0.8993	0.8983

After deciding the best time window, we evaluated the amount of legitimate data needed to yield a good predictive performance. First, we fixed the time window to 1 second and $\nu = 0.1$ and $\gamma = 0.1$. Then, sample percentage was varied from 0.1 to 0.9 with steps of 0.1. At each iteration, for each profile-infection combination, we sampled a percentage of all legitimate data and induced the OSVM on it. After that, this OSVM was tested using the remaining legitimate data and all botnet data. All performance metrics presented very similar results and, in this sense, the OSVM can use a small fraction of legitimate data and still reach high predictive performance. This indicates that all legitimate data are very similar. Thus, the sample size of 0.1 was chosen.

After selecting the best time window and sample size, an optimisation of the OSVM hyperparameters was performed. We used a random search to find the best combination for ν and γ. To do so, twenty values between 0 and 1 (excluding both values) were randomly selected to each metric. Then, by performing a product combination of the two groups of values, 400 ν-γ pairs were created. Recall that ν poses as the upper bound on the fraction of outliers in the training set, whereas γ is used in the RBF kernel to, intuitively, describe the influence of a single instance when creating the hyperplanes. For each combination and each profile-infection pair, we performed the same procedure of inducing an OSVM on 0.1 percent of sampled legitimate data and testing this OSVM

[2] http://scikit-learn.org/stable/

Table 3. Evaluation of Sample Size in all datasets.

Sample Size	Accuracy	Precision	Recall	F1-score	Specificity	AUC
0.1	0.9104	0.7034	0.9563	0.8100	0.8988	0.9276
0.2	0.9106	0.7041	0.9555	0.8102	0.8993	0.9274
0.3	0.9111	0.7052	0.9556	0.8110	0.8999	0.9277
0.4	0.9111	0.7050	0.9556	0.8109	0.8998	0.9277
0.5	0.9112	0.7055	0.9554	0.8111	0.9001	0.9277
0.6	0.9111	0.7051	0.9554	0.8109	0.8999	0.9277
0.7	0.9110	0.7050	0.9555	0.8109	0.8998	0.9277
0.8	0.9111	0.7052	0.9554	0.8110	0.9000	0.9277
0.9	0.9110	0.7050	0.9551	0.8108	0.8999	0.9275

using the remaining legitimate data as well as all botnet data. By doing so, the best values were $\nu = 0.006$ and $\gamma = 0.6$.

In Table 4, we present the mean predictive performance metrics for each profile and infection type when using a time window of 1 second, 0.1 percent of sampled legitimate data, $\nu = 0.006$ and $\gamma = 0.6$. Since each dataset are composed by 3600 legitimate and 3600 botnet instances, only 360 legitimate instances were used for training. By tuning the hyperparameters of the OSVM, we significantly increased the precision, from around 0.70 to 0.95, while maintaining a high recall rate. Accuracy, F1, specificity, and AUC also increased, due to the same reason. Except for the dataset SC_I3, metrics presented values higher than 0.90.

Table 4. Best results for the proposed approach.

Dataset	Accuracy	Precision	Recall	F1-score	Specificity	AUC
MC_I1	0.9847	0.9597	0.9648	0.9622	0.9898	0.9773
MC_I2	0.9744	0.9627	0.9080	0.9345	0.9911	0.9495
MC_I3	0.9909	0.9659	0.9900	0.9778	0.9911	0.9906
SC_I1	0.9928	0.9654	1	0.9823	0.9909	0.9954
SC_I2	0.9926	0.9646	1	0.9819	0.9908	0.9954
SC_I3	0.9449	0.9596	0.7571	0.8463	0.9919	0.8745
ST_I1	0.9665	0.9459	0.8843	0.9140	0.9872	0.9358
ST_I2	0.9931	0.9671	1	0.9832	0.9913	0.9956
ST_I3	0.9933	0.9678	1	0.9836	0.9916	0.9958

To summarise the results in Table 4, a boxplot of the results is presented in Figure 3. It is possible to see that the median values for all metrics were higher than 0.95. Some outliers are present, with the recall metric presenting the highest discrepancy. The median value for this metric was close to 1, but an outlier presented a value of 0.75 (scenario SC_I3). This may be an indication that, for this specific scenario, the hyperparameters selected were not the best, since they were selected considering all scenarios.

The proposed approach relies on the detection of unusual changes in resources consumption that botnet malware causes. The MC profile is the one that has the highest resource consumption among the three observed profiles. Therefore, before the tests, it was thought that detection in the MC profile could be more challenging, because malware behaviour would cause less apparent changes in device resource consumption than in other profiles. The results for the MC profile showed that the proposed approach was successful even in this kind of situation.

Lastly, the results for the ST profile, which has legitimate tasks that could be mistaken as botnet activities, are observed. The basic behaviour of this device is similar

to the SC profile, but it also performs updates in programs and resources checking, does CPU and memory tests, and has periodic legitimate network communication through SSH and Telnet protocols. In the case of Type 3 infection, it has the best results for all datasets, with an accuracy of 99%. These results reinforce that the proposed detection model is capable of discerning legitimate from malicious actions, which are represented, in this case, by botnet actions.

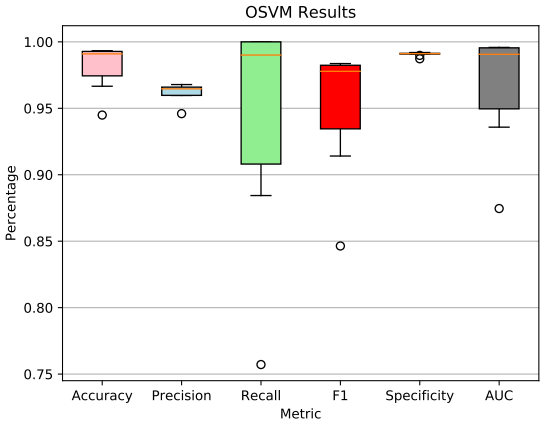

Figure 3. OSVM results with optimised hyperparameters for all datasets.

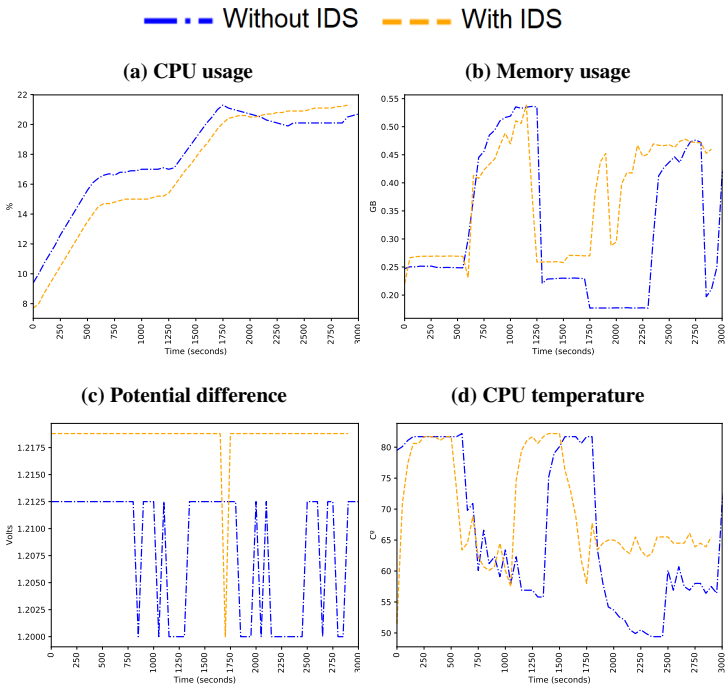

Figure 4. Behaviour of the MC profile regarding host data captured with and without the IDS.

After evaluating the predictive performance of our approach, we also need to assess its impact when running in the device. Since we built a host-based approach and IoT devices have limited resources, it is essential that the IDS uses the resources from the

device in a way to not harm its performance. To address this, we computed the CPU and memory usage, the electric potential difference of the device and the CPU temperature with and without the IDS running. Each feature has, respectively, the following minimum and maximum values: 0% to 100%, 0 GB to 1 GB, 1.2 V to 1.23 V and 0°C to 100°C. In Figures 4, 5 and 6, these values, for each IoT device, are presented. The blue line represents values computed when the IDS was not running, whereas orange lines comprehend to values collected with the IDS running (using the best hyperparameters found).

In Figure 4, the MC profile is considered. The IDS seems not to have affected the device since the behaviours in the two scenarios are very similar. In this sense, even when running the OSVM to classify new instances, the IDS did not compromise the device's performance. The CPU and memory usage, as well as CPU temperature, were similar. Although the electric potential difference in the IDS scenario was higher for the majority of the evaluation period, the difference between the two scenarios were only 0.005 volts.

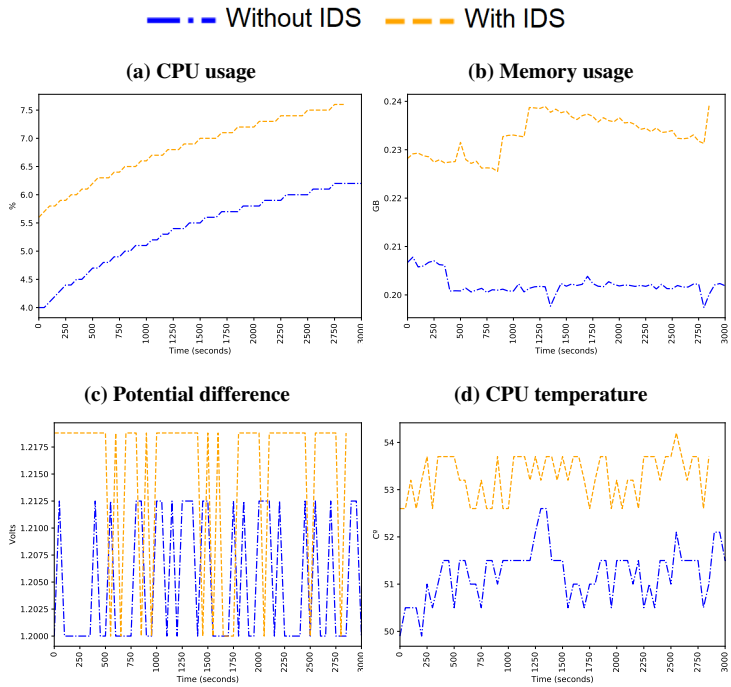

Figure 5. Behaviour of the SC profile regarding host information captured with and without the IDS.

Figure 5 presents the results regarding the SC profile. Since this profile has the lowest use of resources among the three profiles, it is easier to see the impact of the IDS on the device. CPU usage increased by 2 percentage points also followed by the CPU temperature. Memory usage increased by 40 MB but remained constant throughout the time. Likewise, this profile also presented the same increase in the electric potential difference as seen in MC profile. Despite this, resources were far from being exhausted, showing that the IDS did not compromise the functionality of the device.

Lastly, Figure 6 shows the results for the ST profile. Its behaviour was similar to the SC profile. When this profile was running the IDS, CPU usage had a peak of 30%, then it quickly decreased, stabilising around 20%. This peaks could have been caused by the initialisation of the IDS. The electric potential difference also increased by 0.02 V. In

the same manner, as in other profiles, the increase in resource consumption caused by the IDS did not seem to impair the device.

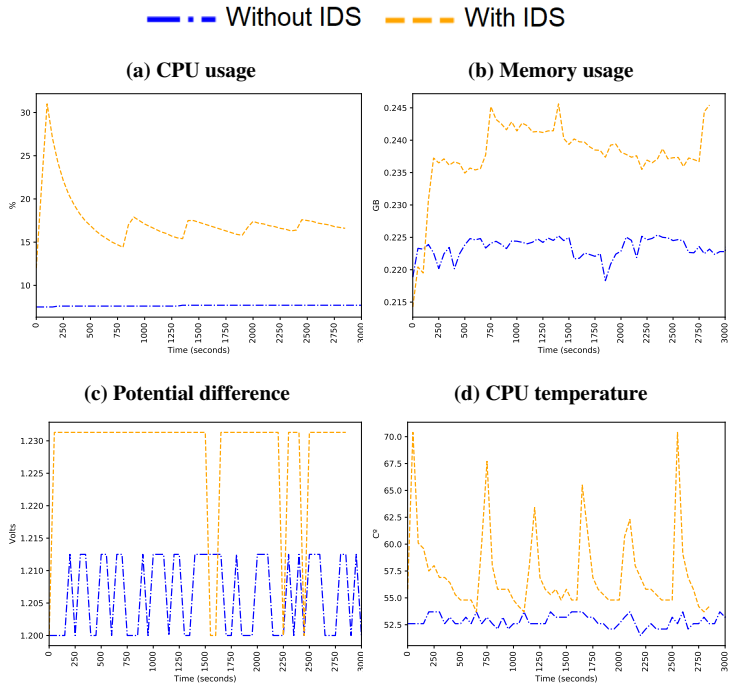

Figure 6. Behaviour of the ST profile regarding host information captured with and without the IDS.

Overall, the proposed approach presented high predictive performance, detecting the seven botnets in all profiles. The different legitimate actions the device performed in the proposed profiles seemed to be distinguishable by the OSVM from most botnet actions. By evaluating multiple time windows, sample sizes, and OSVM hyperparameters, we found out the optimal configuration of 10% of legitimate data used for training, a time window of 1 second and $\nu = 0.006$ and $\gamma = 0.6$, which yielded median values for precision and recall above 95%. Lastly, the impact of the IDS in the performance of the device was also considered. Our tests showed that although there was some increase in resource consumption when the IDS was running, this was far from hindering the capabilities of the device.

7. Conclusion

The increasing number of IoT devices brings new threats to network security since these devices are usually more insecure than usual desktop computers. One way to tackle this issue is to develop IDS to detect infected devices and take the necessary security measures.

In this paper, we developed an approach to detect botnets in IoT devices using the OSVM trained with host-based data, such as the use of CPU and memory, electric potential difference, number of running tasks and temperature of CPU. The approach was tested in a real device using three different profile settings and seven IoT botnets. In our tests, the approach, using only six minutes of legitimate data to induce a model, was capable of detecting all botnets in the different settings using a time window of 1 second. Lastly, we evaluated the impact of running the IDS in a real IoT device. Our tests showed that the approach did not significantly consume the resources from the device.

As future work, we intend to evaluate the proposed approach in different devices and IoT botnets, including other host-based features (e.g., syscalls) and test other one-class classifiers.

References

Abdul Kadir, A. F., Stakhanova, N., and Ghorbani, A. A. (2015). Android botnets: What urls are telling us. In Qiu, M., Xu, S., Yung, M., and Zhang, H., editors, *Network and System Security*, pages 78–91, Cham. Springer International Publishing.

Amaral, J. P., Oliveira, L. M., Rodrigues, J. J., Han, G., and Shu, L. (2014). Policy and network-based intrusion detection system for IPv6-enabled wireless sensor networks. In *Communications (ICC), 2014 IEEE International Conference on*, pages 1796–1801. IEEE.

An, N., Duff, A., Naik, G., Faloutsos, M., Weber, S., and Mancoridis, S. (2017). Behavioral anomaly detection of malware on home routers. In *Malicious and Unwanted Software (MALWARE), 2017 12th International Conference on*, pages 47–54. IEEE.

Angrishi, K. (2017). Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets. *arXiv preprint arXiv:1702.03681*, pages 1–17.

Bertino, E. and Islam, N. (2017). Botnets and Internet of Things Security. *Computer*, 50(2):76–79.

Habibi, J., Midi, D., Mudgerikar, A., and Bertino, E. (2017). Heimdall: Mitigating the internet of insecure things. *IEEE Internet of Things Journal*, 4(4):968–978.

Khan, S. S. and Madden, M. G. (2009). A survey of recent trends in one class classification. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 188–197. Springer.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). DDoS in the IoT: Mirai and Other Botnets. *Computer*, 50(7):80–84.

Mansfield-Devine, S. (2016). DDoS goes mainstream: how headline-grabbing attacks could make this threat an organisation's biggest nightmare. *Network Security*, 2016(11):7 – 13.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., and Elovici, Y. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. *arXiv preprint arXiv:1805.03409*.

Raza, S., Wallgren, L., and Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. *Ad hoc networks*, 11(8):2661–2674.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

Stavrou, A., Voas, J., and Fellow, I. (2017). DDoS in the IoT. *Computer*, 50:80–84.

Whitmore, A., Agarwal, A., and Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. *Information Systems Frontiers*, 17(2):261–274.

Zarpelão, B. B., Miani, R. S., Kawakani, C. T., and de Alvarenga, S. C. (2017). A survey of intrusion detection in Internet of Things. *Journal of Network and Computer Applications*, 84(September 2016):25–37.