IoT Botnet Detection based on Anomalies of Multiscale Time Series Dynamics

João B. Borges, João P. S. Medeiros, Luiz P. A. Barbosa, Heitor S. Ramos, and Antonio A. F. Loureiro

Abstract—In this work, we propose a solution for detecting botnet attacks on the Internet of Things (IoT) by identifying anomalies in the temporal dynamics of their devices. Given their limited computing capabilities, IoT devices are more vulnerable to attacks than conventional computers. In this scenario, botnets have a high degree of severity since they are used to trigging distributed denial-of-service attacks, which are amplified by a large number of IoT devices. Thus, solutions aiming to identify and mitigate the damage caused by botnets in IoT are urgent and essential. We evaluate the number of packets a device transmits, following a multiscale ordinal patterns transformation, and uses Isolation Forest for anomaly detection. By investigating how devices evolve, we can distinguish between normal and anomalous behaviors. We apply the proposed solution to detect two major botnets for IoT: Mirai and Bashlite. We evaluated our model throughout two experimental setups. The first, using a single model for all devices, reaching 99.5% of accuracy and 99.6% of specificity, and the second, by tuning a model per device, reaching 100% of accuracy. These results show that, with the proper transformation, it is possible to use simple methods for detecting anomalies in IoT devices' behaviors.

Index Terms—Internet of Things, Botnet detection, Anomaly detection, Ordinal patterns, Time series dynamics, Multiscale dynamics

1 Introduction

The growing number of Internet of Things (IoT) devices can create several new services and solutions to ease modern life [1], [2], [3]. Ranging from small sensors to powerful devices, the number of IoT-enabled devices is estimated to reach up to 25 billion in the next years [4]. However, because of their computational restrictions and misconfigurations, IoT devices are easy targets for attacks, making their security an urgent concern [5], [6], [7].

A recurrent attack involving IoT devices in recent years, serving as the basis for other attacks, is their infection by bots. A bot is a malicious software that can be used for remote controlling these devices by an attacker, the botmaster. A network of devices compromised by bots is a botnet [5]. The main security threat is that once a device is compromised, the power of an attacker goes from collecting data of its targets, such as password cracking and keylogging, to large-scale orchestrated attacks, such as spam delivery and distributed denial-of-service (DDoS) attacks [5], [6].

Furthermore, although an attack from a small IoT device might not be harmful, as pointed by Kolias et al. [6], the number of IoT devices compensates for their lack of computational resources. For instance, between March and April 2019, a massive botnet attack that used more than 400,000 IoT devices worldwide was detected. This attack produced more than 292,000 requests per minute, aiming to affect the availability of a remote server[1]. This is an aspect that enhances the impact of DDoS attacks, making them hard to be handled by any powerful server.

Thus, following the threat model proposed by Abbas et al. [8], and considering the severity and destructive potential of DDoS attacks once empowered by botnets of billions of compromised IoT devices [9], there is an urge for solutions to identify and mitigate their impact. However, since DDoS attacks are difficult to defend against, given the volatility and different attack patterns [10], the early detection of such digital threats is a reasonable solution. This is possible by evaluating anomalies in the typical behavior of IoT devices. Thus, given the expected behavior of IoT devices, the detection of any deviation can be signaled as an anomaly [11]. In fact, following the claim of Meidan et al. [12], this approach is well suited for IoT since their devices are dedicated to a specific task, and sudden changes may indicate potential attacks of compromised devices [5].

In this paper, our main objectives are to investigate the following research questions:

1) Do these behavioral changes in devices' operation affect their temporal dynamics?
2) In the affirmative case, can these dynamics be used to detect anomalous behavior, which represents an attack?

The temporal dynamics indicate how a system evolves, being an essential aspect for distinguishing data that changes its behavior as a function of time [13], [14], [15]. We argue that this duality between IoT devices' typical and anomalous behavior is an aspect their temporal dynamics can capture. Thus, our proposed strategy for detecting botnet

- *João B. Borges is with the Department of Computing and Technology, Universidade Federal do Rio Grande do Norte, Caicó, RN, 59300-000, Brazil, and the Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 30123-970, Brazil. E-mail: joao.borges@ufrn.br*
- *João P. S. Medeiros and Luiz P. A. Barbosa are with the Department of Computing and Technology, Universidade Federal do Rio Grande do Norte, Caicó, RN, 59300-000, Brazil. E-mails: joao.paulo.medeiros@ufrn.br, luiz.paulo@ufrn.br*
- *Heitor S. Ramos and Antonio A. F. Loureiro are with the Department of Computer Science, Universidade Federal de Minas Gerais, Belo Horizonte, MG, 30123-970, Brazil. E-mails: ramosh@dcc.ufmg.br, loureiro@dcc.ufmg.br*

Manuscript received May 31, 2021; revised May 31, 2021.

1. Massive Botnet Attack Used More Than 400,000 IoT Devices. Accessed in April 04, 2021. https://www.bankinfosecurity.com/massive-botnet-attack-used-more-than-400000-iot-devices-a-12841.

attacks in IoT consists of evaluating anomalies in the dynamical behavior of devices' network traffic during regular and attack periods, aiming to distinguish them.

For instance, once a device is infected by the Mirai malware [6], one of the most prominent botnets for DDoS attacks, it automatically does some operational steps that change its network traffic. Some of them include scanning the network for new victims, sending reports and receiving commands from a command and control (C&C) server, and send attack traffic to a target server [6], [12]. This change in the network traffic can be interpreted as an anomalous behavior and be used to detect a compromised device [6].

To capture these dynamics, we use the ordinal patterns transformation, a cornerstone contribution from Bandt and Pompe [16]. This transformation consists of constructing a set of symbolic ordinal patterns from time series data to distinguish between different dynamics, such as deterministic, noise, or chaotic behaviors [13], [17], [18], [19].

For this work, we add another intrinsic characteristic of dynamical systems: strong dependence on the scale used for sampling the signals [20]. Thus, we consider evaluating the time series following a multiscale approach, i.e., using different time intervals to construct the ordinal patterns. With this approach, we can understand how different time series evolve as well as their temporal correlation dependence. In fact, the multiscale analysis of time series has been demonstrated to be useful information for the correct characterization of time series dynamics [15], [20].

In summary, our main contributions to this work are:

1) We propose a strategy for detecting botnet attacks in IoT based on anomalies in the dynamics of network traffic captured by the ordinal patterns transformation.
2) We propose using a set of features extracted from the multiscale ordinal patterns transformations for representing the changes in dynamics of IoT devices, which are inputs for the anomaly detection algorithm.
3) We demonstrate that with the proper transformation, which is able to capture distinguishing aspects from the typical phenomena, it is possible to use simple methods for detecting anomalies in the behavior of devices.

With these contributions, we advance the state of the art of detecting IoT botnet attacks in a simpler and more efficient way than other strategies from literature. Following the strategy of Nomm et al. [21], we apply a shallow anomaly detection algorithm. Instead of using deep learning methods, we use the Isolation Forest, a model-free anomaly detection algorithm with linear time complexity and low memory requirements [22], [23], [24]. In fact, from the experiments of Nomm et al. [21], the Isolation Forest was the algorithm with the best results.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the background concepts in which our proposal is based. Section 4 presents our proposed strategy for detecting botnets in IoT. Section 5 shows our experiments and results. Finally, Section 6 presents our conclusions and future directions.

2 Related Work

The network traffic data is an essential asset for monitoring and evaluating devices' behaviors in many domain areas. Strategies range from remote distinguishing operating systems [25] to the malfunction detection of devices by the unexpected behavior of their communications [26], [27]. Thus, information extracted from network traffic is of fundamental importance to the analysis of a device's behavior.

With respect to the security concerns of networked devices, unexpected behavior of a device may indicate it is hacked and performing undesired operations. Thus, an important strategy to discover security issues on devices is detecting anomalies in their network traffic. Chandola et al. [26] presented a survey on anomaly detection techniques and applications. They noted that once a device is compromised by a worm, for example, the anomalous traffic is more frequent than the regular traffic. Likewise, García-Teodoro et al. [11] discussed strategies for detecting abnormalities in network traffic and their important role for intrusion detection systems.

Many solutions were proposed to detect network traffic anomalies. Agarwal and Mittal [28] proposal is based on the entropy of network measures. After extracting the network measures, they compute their normalized entropy which are further classified with Support Vector machines (SVM) as normal or attack traffic. The work of Yu et al. [29] consists of detecting flooding attacks by observing network measures of devices, which are collected by Simple Network Management Protocol (SNMP) requests. After collecting data, such as the number of received, sent, and error packets, they use a C4.5 decision tree to identify the attacks.

For the specific case of detecting botnet attacks, Wang et al. [10] presented a study on the characterization and analysis of the behavior of 50,704 different Internet DDoS attacks observed during seven months. For our concerns, they reveal that understanding DDoS attack patterns is the key to defending against them. One of the most prominent patterns they identified is some periodicity of the inter-attack time interval for the DDoS attacks, which may be an interesting aspect for their detection.

Mirsky et al. [30] proposed the Kitsune, a network intrusion detection system (NIDS), which uses an ensemble of autoencoders to detect attacks on local networks. At the core of Kitsune is a feature extraction framework used for capturing measures from the network traffic, which serves as input for the autoencoders. This framework is used for extracting network measures that will serve as a basis for other related proposals, including this one.

Blaise et al. [7] proposed a collaborative intrusion detection system (CIDS) for detection of anomalies in network traffic by considering the aggregated traffic at target ports. They assume that, once a device is compromised, the sudden rise in traffic towards a port is the first steps of their operation. The authors claim that it is possible to early identify a botnet attack by focusing on port-based detection.

Given the increase of IoT botnets and their harmful impact on current network threats, many studies have investigated this problem focusing on anomaly detection. Meidan et al. [12] proposed detecting IoT botnets using autoencoders. They follow the Kitsune method [30] for extracting network traffic measures, which is the input for their autoencoders. The measures were extracted from real IoT devices during normal (benign) and attack operations, which were infected by the Mirai and Bashlite botnets [5],

[6]. The resulting N-BaIoT dataset, detailed in Section 5, is publicly available and used for other studies.

Nomm et al. [21] proposed a strategy for detecting IoT botnets based on shallow methods, rather than using deep learning models as Meidan et al. [12], which the authors affirm to be more appropriated for IoT devices. The authors propose a feature selection approach to consider only the best network traffic measures for further classification as benign and attack event. Alqahtani et al. [31] also used the N-BaIoT dataset for their experiments to detect IoT botnet attacks. The authors propose a feature selection strategy for simplifying the number of network measures in the original N-BaIoT dataset. With the best features, they use extreme gradient boosting (XGBoost) as a classification method. Unlike the other proposals, which used only the benign traffic data for training their models, in this last one, the authors considered both benign and attack data for the model training step. Popoola et al. [32] proposed a federated deep learning method for zero-day botnet attack detection in IoT edge devices. Their strategy uses a federated averaging strategy to aggregate all local models from edges, producing a global deep neural network model. The authors used a subset of five attacks from the N-BaIoT dataset.

To put our work in perspective, let us compare it with the previous related work. Our focus is to detect anomalies in the N-BaIoT network traffic dataset, collected by Meidan et al. [12]. Thus, we can compare our results with their work and the work of Nomm et al. [21], Alqahtani et al. [31], and Popoola et al. [32] which also use the same dataset. Our first difference from them is in the number of network measures used to detect anomalies. These works use all the 115 measures from the N-BaIoT or select the 3 to 10 best ones. Instead, we only consider one single network traffic measure, the number of packets sent from the device.

3 Background

In this work we propose using time series dynamical behavior to detect anomalies in IoT devices, which may indicate a compromised device performing some step of a botnet attack. To capture such dynamics, our solution uses features extracted from the ordinal patterns transformation. This section introduces these transformations, which implementation codes are available at a public repository[2].

3.1 The ordinal patterns transformation

Bandt and Pompe [16] proposed a methodology for the representation of time series that is a cornerstone contribution to the study of time series dynamics [15], [17], [18], [19], [20], [33], [34]. This transformation consists of creating a set of symbolic patterns based on the ordinal relation between successive data points of the time series. It uses two parameters, the embedding dimension D, which is related to the length of the patterns, and the embedding delay τ, which defines the interval between consecutive data points.

Formally, for a given time series $\mathbf{x} = (x_1, \ldots, x_n)$ of length n, an embedding dimension $D \in \mathbb{N}$, and an embedding delay $\tau \in \mathbb{N}$, the method consists of generating sliding windows $\mathbf{w}_t \subseteq \mathbf{x}$ of length D each, for each time instant t, such that the elements within each sliding window can be separated by intervals of size τ, corresponding to a sample of the time series by regular spaced intervals [20]. Thus, the sliding window for each instant $t = 1, \ldots, n - (D-1)\tau$ is

$$\mathbf{w}_t = (x_t, x_{t+\tau}, \ldots, x_{t+(D-2)\tau}, x_{t+(D-1)\tau}). \quad (1)$$

The ordinal relation [13] for each sliding window is the necessary permutation in the elements of $\mathbf{w}_t$, so they are sorted in ascending order by their values. Thus, for each $\mathbf{w}_t$, at a given instant t, the ordinal pattern consists of the permutation $\pi_t = (r_1, r_2, \ldots, r_D)$ of $(1, 2, \ldots, D)$ such that

$$x_{t+r_1-1} \leq x_{t+r_2-1} \leq \cdots \leq x_{t+r_{D-1}-1} \leq x_{t+r_D-1}. \quad (2)$$

Figure 1 illustrates the process of constructing the ordinal patterns. For the presented time series, with $D = 3$ and $\tau = 1$, the sliding windows of length 3 are sequentially obtained, and the ordinal relation for each of them are computed. The first sliding window obtained at $t = 1$ is given by $\mathbf{w}_1 = (w_{1,1}, w_{1,2}, w_{1,3}) = (-0.37, 1.22, 0.34)$, and the necessary permutation to order it is to switch the second and third elements, keeping the first data point. The new ordered $\mathbf{w}'_t = (w_{1,1}, w_{1,3}, w_{1,2})$ corresponds to the 132 pattern, as presented in the figure.

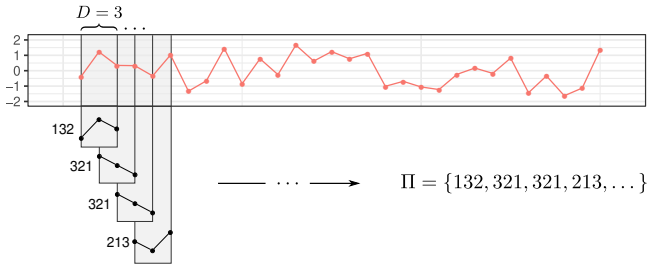

Fig. 1: Illustration of the ordinal patterns transformation process, considering an embedding dimension $D = 3$ and an embedding delay $\tau = 1$.

For the second and third sliding windows we have $\mathbf{w}_2 = (1.22, 0.34, 0.32)$ and $\mathbf{w}_3 = (0.34, 0.32, -0.33)$, respectively. Since both of them are in reverse order, they need the same permutation to be ordered, corresponding to the 321 patterns. It is worth noting that the transformation does not consider variations in the amplitudes of data points. The pattern is computed only with respect to the ordinal relation between them, thus the ordinal name. After the transformation, the time series is converted onto a set of ordinal patterns $\Pi = \{\pi_i : i = 1, \ldots, n - (D-1)\tau\}$ and each π_i represents a permutation from the set of $D!$ possible permutations. The choice of D depends on the length n of the time series and the condition $n \gg D!$ must be satisfied to obtain reliable statistics [13], [20].

This transformation is our strategy's main time-consuming operation, dominating all other computational costs. Its time complexity is $O(nD^2)$, assuming a naive sorting algorithm obtains the permutation by classifying each sliding window. However, since for practical purposes D is in the interval between 3 and 7 [20], and in our experiments, we consider $D \in \{3, 4\}$, the sorting algorithm will take at most 4 elements. The complexity of this strategy

2. Implementations of the Bandt and Pompe's ordinal patterns transformations: https://github.com/labepi/bandt_pompe.

depends mainly on the size n of the time series, which gives practically $O(n)$.

3.2 Derived transformations from ordinal patterns

Once the ordinal patterns are found, the next step is to analyze them to capture the time series dynamics. Two reasonable methods to do this are taking into account the frequency of patterns and the sequence of their occurrences. For the first method, the analysis is performed through the ordinal patterns probability distribution p_π. For the second case, we map the transitions between adjacent patterns to the ordinal patterns transition graph G_π. In this section, we present both transformations, obtained from the set Π.

3.2.1 Ordinal patterns probability distribution (p_π)

Given the set Π of ordinal patterns, the ordinal patterns probability distribution p_π consists of assigning a probability distribution to the permutations identified in the time series. Thus, for each possible permutation $\pi_t \in \Pi$, with $t \in \{1, \ldots, D!\}$, let $|s_{\pi_t}| \in \{0, \ldots, m\}$ be the number of observed patterns of type π_t, then $p_\pi = \{p(\pi_t) : \forall t \in 1, \ldots, D!\}$ is defined as

$$p(\pi_t) = \frac{|s_{\pi_t}|}{n - (D-1)\tau}, \tag{3}$$

satisfying the conditions $p(\pi_t) \geq 0$ and $\sum_{\pi_t} p(\pi_t) = 1$.

3.2.2 Ordinal patterns transition graph (G_π)

Also, from a given set of ordinal patterns Π, the ordinal patterns transition graph G_π represents the relations between consecutive ordinal patterns. It is defined as a directed weighted graph $G_\pi = (V, E)$ with $V = \{\pi_i : i = 1, \ldots, D!\}$, where each vertex π_i corresponds to one of the $D!$ possible permutations from Π for an embedding dimension D, and a set of directed weighted edges $E = \{(\pi_i, \pi_j) : \pi_i, \pi_j \in V\}$.

A directed edge connects two ordinal patterns in the graph if they appear sequentially in the time series, representing a transition between them. There is an edge $(\pi_i, \pi_j) \in E$, between the vertices π_i and π_j, if there is a sequence of ordinal patterns $\Pi_t = \pi_i$ and $\Pi_{t+1} = \pi_j$, $1 \leq t \leq n - (D-1)\tau - 1$. The weight function $w : E \to \mathbb{R}$ of an edge represents the probability of a specific transition in Π. Thus, the weight of a given edge (π_i, π_j) is given by

$$w(\pi_i, \pi_j) = \frac{|\Pi_{\pi_i, \pi_j}|}{m - 1}, \tag{4}$$

where $|\Pi_{\pi_i, \pi_j}| \in \{0, \ldots, n - (D-1)\tau - 1\}$ is the number of transitions between π_i and π_j, with $\sum_{\pi_i, \pi_j \in V} w(\pi_i, \pi_j) = 1$.

Once the graph is constructed, some authors employ unweighted edges [35], [36] representing only the existence of transitions. However, in this work, we consider the relative frequency of transitions as edges' weights [15], [37], [38].

3.3 Numerical analysis of the transformations

A reasonable strategy to capture the characteristics from distinct time series dynamics consists of extracting relevant features from the ordinal patterns transformations. In this work, we use metrics extracted from both p_π and G_π.

3.3.1 Features from p_π

From the ordinal patterns probability distribution p_π, we compute the following features:

1) normalized permutation entropy ($H_S[p_\pi]$),
2) statistical complexity ($C_{JS}[p_\pi]$), and
3) Fisher information measure ($F[p_\pi]$).

These features are information theory quantifiers that have already been used as significant measures for the proper characterization of the underlying time series dynamical behavior [13], [17], [18], [19], [39], [40], [41], [42].

3.3.1.1 Permutation entropy: Since p_π is defined on the permutations of neighboring values, a proposed variation from the classical entropy of Shannon, called the *permutation entropy*, is defined as

$$H[p_\pi] = -\sum_{t=1}^{D!} p(\pi_t) \log p(\pi_t), \tag{5}$$

where $0 \leq H[p_\pi] \leq \log D!$. The permutation entropy is equivalent to the Shannon entropy and is a measure of uncertainty of the process described by p_π [18]. Lower values of $H[p_\pi]$ indicate a more deterministic time series and higher values of $H[p_\pi]$ a completely random system [16].

3.3.1.2 Normalized permutation entropy: The maximum value for $H[p_\pi]$ occurs when all $D!$ possible permutations have the same probability, which is the uniform distribution p_u. Thus, Rosso et al. [13] defined the normalized Shannon entropy from the permutation entropy as

$$H_S[p_\pi] = \frac{H[p_\pi]}{H_{\max}}, \tag{6}$$

where $0 \leq H_S[p_\pi] \leq 1$ and $H_{\max} = H[p_u] = \log D!$.

3.3.1.3 Statistical complexity: Defined by Lamberti *et al.* [43], this measure presents a different perspective regarding the knowledge of some underlying process. It captures the relationship between the dynamical components of a system, such as determinism and randomness. Based on the Jensen-Shannon divergence JS between the associated probability distribution p_π and the uniform distribution p_u, i.e., the trivial case for the minimum knowledge from the process, the statistical complexity is given by

$$C_{JS}[p_\pi] = Q_{JS}[p_\pi, p_u] H_S[p_\pi], \tag{7}$$

where $p_\pi = \{p(\pi)\}$ is the ordinal patterns probability distribution, p_u is the uniform distribution, and $H_S[p_\pi]$ is the normalized Shannon entropy, as previously described. The disequilibrium $Q_{JS}[p_\pi, p_u]$ is given by

$$Q_{JS}[p_\pi, p_u] = Q_0 JS[p_\pi, p_u] \tag{8}$$

$$= Q_0 \left\{ S\left[\frac{p_\pi + p_u}{2}\right] - \frac{S[p_\pi] + S[p_u]}{2} \right\}, \tag{9}$$

where S is the Shannon entropy measure. Q_0 is a normalization constant, given by

$$Q_0 = -2 \left\{ \left(\frac{D! + 1}{D!} \right) \ln(D! + 1) - 2\ln(2D!) + \ln(D!) \right\}^{-1}, \tag{10}$$

which is equal to the inverse of the maximum value of $JS[p_\pi, p_u]$, so $0 \leq Q_{JS} \leq 1$ [18], [44].

3.3.1.4 Fisher information: The Fisher information is a measure able to capture the concentration of a distribution. Contrary to Shannon entropy, which provides a notion of the uncertainty of a system by measuring the distribution's global spreading, the Fisher information is considered to have a locality property because it reflects the differences among consecutive probabilities of a distribution [14], [45]. The Fisher information for the discrete case is given by

$$F[p_\pi] = F_0 \sum_{t=1}^{D!-1} \left(\sqrt{p_{t+1}} - \sqrt{p_t}\right)^2, \tag{11}$$

where F_0 is a normalization constant defined as

$$F_0 = \begin{cases} 1 & \text{if } p_{i^*} = 1 \text{ for } i^* = 1 \text{ and } p_i = 0, \forall i \neq i^*, \\ 1 & \text{if } p_{i^*} = 1 \text{ for } i^* = N \text{ and } p_i = 0, \forall i \neq i^*, \\ 1/2 & \text{otherwise.} \end{cases} \tag{12}$$

By measuring these local changes, the Fisher information can provide a quantifier of the gradient of the distribution, which increases as the density is more concentrated [45].

3.3.2 Features from G_π

From the ordinal patterns transition graph G_π, we extracted the following set of features:

1) number of edges (N_E),
2) norm. Shannon's entropy of edges weights ($H_S[E_w]$),
3) statistical complexity of edges weights ($C_{JS}[E_w]$),
4) Fisher information of edges weights ($F[E_w]$), and
5) probability of self transitions (p_{st}).

Once we have the graph of the transitions between patterns, we can compute several features. For instance, Ravetti *et al.* [46] compute their features from the nodes degree distribution, but this requires a large number of vertices to be relevant, which is not our intention, since this needs a large D. On the other hand, following the studies of Zhang *et al.* [38], Borges *et al.* [15], and Olivares *et al.* [47], we focus on the features that represent the transitions themselves, which are related to the edges of the graph.

3.3.2.1 Number of edges: Since the edges of G_π represent the occurrence of transitions between consecutive patterns, their number is an indicator of time series dynamics. For instance, as the randomness of a process increases, it also increases the chances for all possible transitions in the graph to occur. The number of edges is computed by

$$N_E = |E[G_\pi]|. \tag{13}$$

3.3.2.2 Information theory quantifiers from the weight distribution: Similarly to the features computed from p_π, we calculate the same information theory quantifiers from the edge weights of G_π, as previously described.

3.3.2.3 Probability of self-transitions: The probability of self-transitions is defined by Borges et al. [15] as the probability of occurring a sequence of identical patterns within the set of ordinal patterns, and is expressed as

$$p_{st} = p(\pi_i, \pi_i) = \sum_{i \in \{1,\ldots,D!\}} w(v_{\pi_i}, v_{\pi_i}). \tag{14}$$

Since the self-transitions, which are loops in the graph, are directly related to the temporal correlation of time series, it is a valuable indication of their underlying dynamics. In fact, Borges *et al.* [15] show that the p_st,when evaluated as a function of the embedding delay τ, is a valuable indicator of the main characteristics of the time series, even for small values of D, and independently on a single value of τ.

4 Detection of Botnet Attacks in IoT

This section presents our strategy of using the ordinal patterns transformations for detecting anomalies in IoT devices. Our proposed strategy consists of three main parts:

1) Construct time series from the number of packets sent by a suspicious device directly from its network traffic.
2) Transform time series, via the multiscale ordinal patterns transformation, onto a set of features that can capture the dynamical behavior of the devices' operation.
3) Classify the transformed time series to distinguish an anomalous from a regular devices' behavior.

In the following, we detail this proposed strategy.

4.1 The network traffic capture

Figure 2 illustrates the network model used to capture the traffic of packets from a given suspicious IoT device, following the Kitsune framework [30]. Let us suppose a scenario where all the traffic from a given device flows through a local router. Thus, an agent within this router will passively capture and parse the received packets to compute network traffic measures.

Our strategy only needs data from a single network traffic measure, the number of packets from a device, aggregated within a time window of 100 ms. However, since we evaluate the evolution of this measure over time, we need to collect consecutive samples of it to create a time series, which will be further used to detect the anomalies.

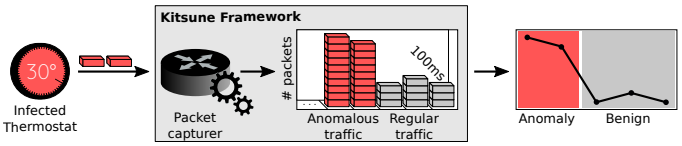

Fig. 2: Network model to capture the IoT devices' traffic.

The next step consists of transforming the constructed time series by the multiscale ordinal patterns transformation, described in Section 4.2. After this transformation, we will have a set of relevant features to represent the characteristics from the distinct dynamics of the devices' operations. Then, these features are the input for the Isolation Forest anomaly detection algorithm [22], described in Section 4.3.

4.2 The multiscale ordinal patterns transformation

Figure 3 depicts the detailed process to perform the transformations necessary on the time series. For a given time series $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,m})$, of length m, the ordinal patterns transformation is applied as described in Section 3.1. The parameters are D and $\tau_i \in \mathcal{T}$, with $i = \{1, \ldots, t\}$, corresponding to a given embedding dimension and an embedding delay (time interval) from the list, respectively.

For each pair (D, τ_i), the result of this step is a set of ordinal patterns Π_{τ_i} representing the transformation of the

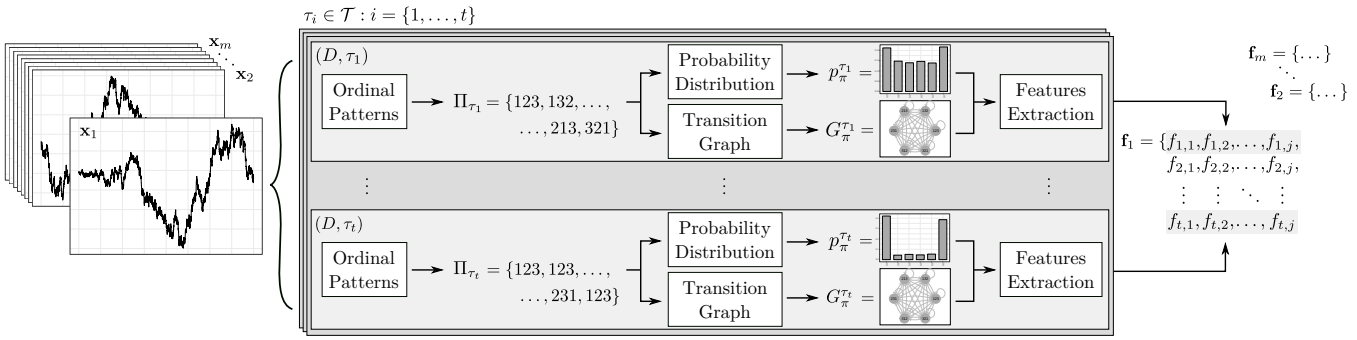

Fig. 3: Process for the transformation of a time series $\mathbf{x}_i$ onto a new vector of features $\mathbf{f}_i$, based on the multiscale ordinal patterns transformations, for a given embedding dimension D and a list of embedding delays $\mathcal{T}$ of length t.

time series with D and $\tau_i \in \mathcal{T}$ as parameters. Thus, given the list of considered embedding delays, each time series is transformed into t different sets of ordinal patterns.

Once a set of ordinal patterns Π_{τ_i} is obtained, it is transformed onto both an ordinal patterns probability distribution ($p_\pi^{\tau_i}$) and an ordinal patterns transition graph ($G_\pi^{\tau_i}$), Sections 3.2.1 and 3.2.2, respectively. From both $p_\pi^{\tau_i}$ and $G_\pi^{\tau_i}$, a set of features is extracted, consisting of the new representation for the time series $\mathbf{x}_i$, which will be used to create the new features vector. Therefore, for each time series $\mathbf{x}_i$ and for each pair (D, τ_i), we have the set of features

$$\mathbf{f}_{\tau_i} = \{f_{\tau_i,1}, f_{\tau_i,2}, \ldots, f_{\tau_i,j}\}, \tag{15}$$

where j is the number of features extracted.

In this work, from a given time series $\mathbf{x}_i$, there is a total of $j = 8$ features extracted for each pair (D, τ_i), presented in Section 3.3. Thus, the vector presented in Equation 15 can be instantiated to the following features,

$$\mathbf{f}_{\tau_i} = \{N_E^{\tau_i}, H_S[E_w]^{\tau_i}, C_{JS}[E_w]^{\tau_i}, F[E_w]^{\tau_i}, p_{st}^{\tau_i}, H_S[p_\pi]^{\tau_i}, C_{JS}[p_\pi]^{\tau_i}, F[p_\pi]^{\tau_i}\}. \tag{16}$$

The next step for transforming the time series corresponds to the composition of a vector $\mathbf{f}$ of all extracted features for each $\tau_i \in \mathcal{T}$, representing the final transformation. To illustrate this step, Equation 17 shows the set of features $\mathbf{f}$ constructed from the time series $\mathbf{x}$.

$$\mathbf{f} = \bigcup_{i=1}^{t} \mathbf{f}_{\tau_i} = \bigcup_{i=1}^{t} \{f_{\tau_i,1}, f_{\tau_i,2}, \ldots, f_{\tau_i,j}\} \tag{17}$$

After these steps, the new ordinal patterns domain for the raw time series corresponds to a new features vector of length tj. Algorithm 1 describes the transformation process of a given time series onto this novel features representation, following the multiscale ordinal patterns transformations.

ALGORITHM 1: MULTISCALETRANSFORMATION

Input: A time series $\mathbf{x} = \{x_1, \ldots, x_m\}$ of length m, an embedding dimension D and a list of embedding delays $\mathcal{T}$ of length t.
Output: The resulting $\mathbf{f} = \{f_1, \ldots, f_{tj}\}$ features after the multiscale transformations process.

```
// Transforming the time series for each τ
foreach τ_i ∈ T do
    // Main transformation
    Π_τi ← ordinalPatterns(x, D, τ_i);
    // Probability Distribution
    p_π^τi ← ordinalPatternsPD(Π_τi, D);
    // Transition Graph
    G_π^τi ← ordinalPatternsTG(Π_τi, D);
    // Extracting the features
    f_τi ← extractFeatures(p_π^τi, G_π^τi);
// Joining features from all τ's as a single features vector
f = {f_τ1, ..., f_τt};
return f;
```

4.3 The anomaly detection strategy

For detecting anomalies in IoT devices, our method requires time series samples to learn its behavior. These time series are constructed by consecutive observations of a single network traffic measure at distinct moments. Thus, for a given network measure k, our proposal consists of training a model with the time series observations of k during the benign operation of the devices and, then, performing the anomaly detection for unseen observations.

4.3.1 The training phase

For training the model, let us assume the dataset of time series from a given network measure k, collected during its benign operation, is available. Let the matrix $\mathbf{X}_{n\times m}^k = [\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_n^k]^\top$ be this dataset consisting of n distinct time series for a specific network measure k. For the sake of clarity, let us assume each time series $\mathbf{x}_i^k = x_{i,1}^k, \ldots, x_{i,m}^k$ consists of consecutive samples and has the same length m. However, it is important to highlight that the method does not require time series of equal lengths.

After performing the multiscale transformation for each time series, described in Section 4.2, the novel matrix of features vectors $\mathbf{F}_{n\times tj}^k \leftarrow [\mathbf{f}_1^k, \mathbf{f}_2^k, \ldots, \mathbf{f}_n^k]^\top$ is obtained. This matrix of features will be used for training a model with the Isolation Forest anomaly detection algorithm. The processes for training the model c^k, specific for the network traffic measure k, is presented in Algorithm 2.

4.3.2 The anomaly detection phase

For the anomaly detection phase, a given time series of unseen observations is passed through the isolation forest model c^k, which results in an anomaly score s. According to the value of s, this time series will be concluded as a regular time series, i.e., similar to the time series presented to the model during its training or an anomalous time series.

Thus, let $\mathbf{y}^k = (y_1^k, \ldots, y_m^k)$ be a time series of m suspicious observations from the network traffic measure

ALGORITHM 2: MODELTRAINING

Input: The time series dataset $\mathbf{X}^k_{n\times m}$ for a specific network traffic measure, an embedding dimension D, and a list of embedding delays $\mathcal{T}$ of length t.
Output: The resulting classification model c^k for the network traffic measure k.

```
// Transforming each time series from the dataset
   onto a set of features
foreach x^k ∈ X^k do
    // Main transformation
    f^k ← MultiscaleTransformation(x^k, D, T);
// The matrix of transformed features
F^k_{n×tj} ← [f^k_1, f^k_2, ..., f^k_n]^⊤
// Fitting a model
c^k ← IsolationForest(F^k, n_trees);
return c^k;
```

k. As in the case when training the model, the observations are consecutive samples collected from the same network measure k. After computing the features vector $\mathbf{f}^k$, with the multiscale ordinal patterns transformation, its anomaly score s is computed with the isolation forest model c^k.

The behavior of the given time series is predicted by simply evaluating the value of s according to a defined anomaly threshold e. As originally defined by Liu et al. [22], significant anomaly scores potentially indicate an anomaly. Section 5.1 gives more details about the value for this threshold as a limit for our detection strategy. Algorithm 3 illustrates these steps for the anomaly detection phase.

ALGORITHM 3: ANOMALYDETECTION

Input: A suspicious time series $\mathbf{y}^k$ for the network traffic measure k, the isolation forest model c^k, an anomaly score threshold e, an embedding dimension D, and a list of embedding delays $\mathcal{T}$ of length t.
Output: The detection result of the suspicious time series as *benign* or *attack* traffic.

```
// Applying the multiscale transformation
f^k ← MultiscaleTransformation(y_k, D, T);
// Computing the anomaly score
s ← predictInstance(c^k, f^k);
// Evaluate the time series behavior
if s < e then
    return benign;
else
    return attack;
```

5 RESULTS AND DISCUSSION

This section presents our results for the IoT botnet detection by evaluating anomalies in the temporal dynamics in their network traffic. To evaluate our approach, we compare our results with related work in the literature by performing experiments on the real-world N-BaIoT dataset [12]. However, we first give more details on the chosen methods and decisions, the data preprocessing, and software versions.

5.1 Materials and methods

The experiments in this work used the N-BaIoT public dataset[3]. It was created by Meidan et al. [12] and consists of network traffic collected from nine IoT devices during both regular operation (benign) and under botnet attacks (anomaly). These devices are one thermostat, one baby monitor, one webcam, two doorbells, and four security cameras.

The anomalous network traffic was generated by infecting each device with the Mirai and Bashlite botnet families [5], [6]. These are two of the most prominent botnets that are still a risk for the current IoT. For instance, although the Mirai source code was released in late 2016 [6], in 2021, we still find numerous Mirai-based botnets infecting and exploring vulnerabilities of IoT devices[4]. Furthermore, the operation of a device once infected by these botnets is similar to the original botnets they are derived from, such as contacting a C&C server, flooding the network, and scanning for other vulnerable devices. Thus, this makes the N-BaIoT dataset still relevant for evaluating anomalous behavior.

For each botnet, the authors captured the network traffic data under five different attacks. For the Mirai botnet, the attacks were: 1) ACK flooding; 2) SCANning the network for new victims; 3) SYN flooding; 4) UDP flooding; and 5) UDPPLAIN, which is another UDP flooding but with fewer options. For the Bashlite botnet, the attacks were: 1) SCANning the network for new vulnerable devices; 2) TCP flooding; 3) UDP flooding; 4) JUNK, which consists of sending spam data; and 5) COMBO of sending spam data and connecting to a specific IP address and port. Thus, there are 11 distinct classes, ten classes of attacks, and one class of benign data, for each of the nine IoT devices. Table 1 presents the dataset properties, illustrating the devices used for the experiments and the numbers of samples for each of the 11 classes. It can be observed that there is no traffic data from the Mirai botnet for two devices, only for Bashlite.

To collect the network traffic measures for each of these samples, the authors followed the method proposed by Mirsky et al. [30], the Kitsune network intrusion detection system. Thus, for each one of these 11 classes, for each IoT device, after capturing the raw traffic data from those devices, the authors aggregated the samples within five different time windows: 1 min, 10 s, 1.5 s, 500 ms, and 100 ms; which are coded as their decay factor λ as L0.01, L0.1, L1, L3, and L5, respectively. Furthermore, they computed 23 distinct network traffic measures for each of these time windows, consisting of statistics from the packet's sender and the traffic between the packet's sender and receiver [30]. Consequently, each sample of Table 1 consists of a snapshot with 115 distinct network traffics measures.

To handle the samples from the N-BaIoT dataset, the other related work, presented in Section 2, uses as input for their anomaly detection methods the samples snapshot directly, whether it be the total 115 measures or a subset of them. Instead, our method needs to construct a time series from consecutive samples of a single network measure to learn their temporal dynamics. Thus, for our experiments, we chose the `MI_dir_L5_weight` network traffic measure, as the dataset notation, to be our k measure, as described in Section 4.3. This measure consists of the number of packets (weight) originated from a single device, represented by a

3. N-BaIoT dataset is available to download at the UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/datasets/detection_of_IoT_botnet_attacks_N_BaIoT.

4. Dark Mirai botnet targeting RCE on popular TP-Link router. Accessed in Dec. 14, 2021. https://www.bleepingcomputer.com/news/security/dark-mirai-botnet-targeting-rce-on-popular-tp-link-router/.

TABLE 1: Description of the N-BaIoT dataset, illustrating the IoT devices used in the experiment and the number of samples for each benign and attack classes.

#	Device	Benign	Mirai		Bashlite	
1	Danmini Doorbell	49,548	ACK	102,195	COMBO	59,718
			SCAN	107,685	JUNK	29,068
			SYN	122,573	SCAN	29,849
			UDP	237,665	TCP	92,141
			UDPPLAIN	81,982	UDP	105,874
			TOTAL:	652,100	**TOTAL**:	316,650
2	Ecobee Thermostat	13,113	ACK	113,285	COMBO	53,012
			SCAN	43,192	JUNK	30,312
			SYN	116,807	SCAN	27,494
			UDP	151,481	TCP	95,021
			UDPPLAIN	87,368	UDP	104,791
			TOTAL:	512,133	**TOTAL**:	310,630
3	Ennio Doorbell	39,100	–	–	COMBO	53,014
			–	–	JUNK	29,797
			–	–	SCAN	28,120
			–	–	TCP	101,536
			–	–	UDP	103,933
			TOTAL:	0	**TOTAL**:	316,400
4	Philips B120N10 Baby Monitor	175,240	ACK	91,123	COMBO	58,152
			SCAN	103,621	JUNK	28,349
			SYN	118,128	SCAN	27,859
			UDP	217,034	TCP	92,581
			UDPPLAIN	80,808	UDP	105,782
			TOTAL:	610,714	**TOTAL**:	312,723
5	Provision PT 737E Security Camera	62,154	ACK	60,554	COMBO	61,380
			SCAN	96,781	JUNK	30,898
			SYN	65,746	SCAN	29,297
			UDP	156,248	TCP	104,510
			UDPPLAIN	56,681	UDP	104,011
			TOTAL:	436,010	**TOTAL**:	330,096
6	Provision PT 838 Security Camera	98,514	ACK	57,997	COMBO	57,530
			SCAN	97,096	JUNK	29,068
			SYN	61,851	SCAN	28,397
			UDP	158,608	TCP	89,387
			UDPPLAIN	53,785	UDP	104,658
			TOTAL:	429,337	**TOTAL**:	309,040
7	Samsung SNH 1011 N Webcam	52,150	–	–	COMBO	58,669
			–	–	JUNK	28,305
			–	–	SCAN	27,698
			–	–	TCP	97,783
			–	–	UDP	110,617
			TOTAL:	0	**TOTAL**:	323,072
8	SimpleHome XCS7 1002 WHT Security Camera	46,585	ACK	111,480	COMBO	54,283
			SCAN	45,930	JUNK	28,579
			SYN	125,715	SCAN	27,825
			UDP	151,879	TCP	88,816
			UDPPLAIN	78,244	UDP	103,720
			TOTAL:	513,248	**TOTAL**:	303,223
9	SimpleHome XCS7 1003 WHT Security Camera	19,528	ACK	107,187	COMBO	59,398
			SCAN	43,674	JUNK	27,413
			SYN	122,479	SCAN	28,572
			UDP	157,084	TCP	98,075
			UDPPLAIN	84,436	UDP	102,980
			TOTAL:	514,860	**TOTAL**:	316,438
	TOTAL	555,932	3,668,402		2,838,272	

pair of source MAC and IP addresses (MI), aggregated by time windows of 100 ms (L5), constructed in an incremental approach [30]. The reason for choosing this measure follows the results of Alqahtani et al. [31], which analyzed the measures' importances by their Fisher scores. However, differently from those authors, we chose the measure with the smallest time window, while they chose the one with the highest time window (L0.01).

Concerning the parameters chosen for the algorithms and experiments, we varied the length m of the time series constructed from the selected measure in our experiments to check its impact on the detection results. We also evaluated the values of the embedding dimension D, and the length t of the embedding delays list $\mathcal{T}$ by varying their values. The number of trees used in our experiments was $n_{\text{trees}} = 300$, which provided good results with reasonable execution time. The remaining Isolation Forest parameters was kept as default. For the anomaly score threshold e, which is used to conclude a benign or attack behavior, as described in Section 4.3, we chose the fixed value of $e = 0.59$. The original Isolation Forest work [22] partially supports the reason for this choice since it shows that potential anomalies can be identified when the anomaly score is $s \geq 0.6$. However, by empirical observations, we obtained the value of $e = 0.59$ was more appropriate, giving better detection results.

Our algorithms and experiments were implemented in the R statistical software suite [48], version 4.0.3, with some code excerpts in C++. For the Isolation Forest anomaly detection, we used the R package `isotree`[5] version 0.2.7. All codes used in our experiments are available at a public repository[6]. All experiments were performed on a computer with Intel Core i9-9900X CPU at 3.50GHz x 20, 128 GB RAM, and running a Linux Ubuntu 18.04.4 LTS 64-bit.

5.2 Temporal dynamics of a Botnet attack

The success of the whole anomaly detection process depends on the right choice of the features and their ability to represent the different behaviors from the time series. When dealing with the ordinal patterns transformations, this involves the proper definition of the parameters (D, τ).

The embedding dimension D depends on the length n of the time series, since it will define the alphabet of size $D!$ to which the possible patterns will be mapped. As previously discussed in Section 3.1, the main recommendation is that the condition $n \gg D!$ must be satisfied to obtain reliable statistics [13], [20]. Given the rapid increase of the factorial function, for practical purposes, Bandt and Pompe [16] recommended $D \in \{3, \ldots, 7\}$.

For the embedding delay τ, there is no strict recommendation, but it is already known that τ is related to some intrinsic time-space characteristics of the series, such as periodicity and time scales [20]. While a common choice of several studies is $\tau = 1$, we show that different values of τ enable the features extracted from both p_π and G_π to result in a more separable space between the different classes.

Figure 4 illustrates the first 400 samples of the chosen `MI_dir_L5_weight` network measure for different operations of the Danmini Doorbell device from the N-BaIoT

5. isotree: Isolation-Based Outlier Detection: https://cran.r-project.org/web/packages/isotree/index.html.

6. The repository with the code for our IoT Botnet Detection is available at https://github.com/labepi/anomaly_iot. This code requires the Bandt-Pompe ordinal patterns transformations, available at https://github.com/labepi/bandt_pompe.

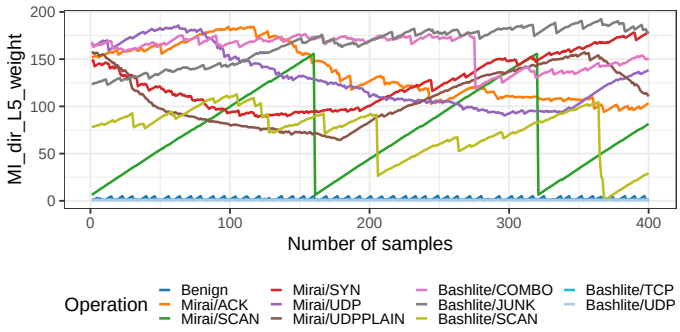

Fig. 4: Illustration of the first 400 samples of the chosen MI_dir_L5_weight network measure for different operations of the Danmini Doorbell device of the N-BaIoT dataset.

dataset. It can be noticed that different moments of the operation result in very distinct behaviors for this measure. For instance, the samples collected during the benign operation have a pattern similar to the additive increase, multiplicative decrease (AIMD) operation of TCP. Instead, other botnet operations, such as the ACK or SYN operations from Mirai, are more random, with no apparent pattern.

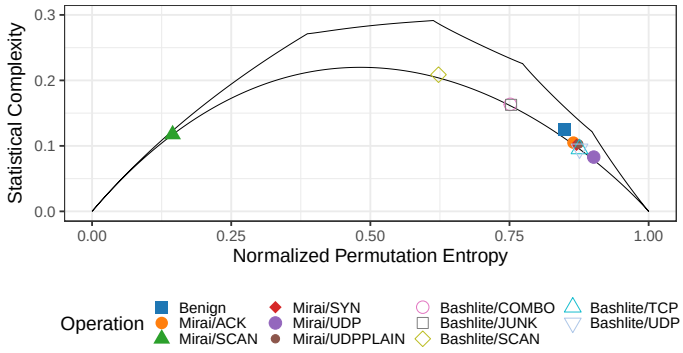

Fig. 5: Illustration of the CCEP for the different operations of the MI_dir_L5_weight feature, presented in Figure 4, computed with $D = 3$ and $\tau = 1$.

By considering the temporal dynamics of these behaviors, we can observe some patterns that can be used for distinguishing them. Figure 5 presents the causality complexity-entropy plane (CCEP), proposed by Rosso et al. [13], computed from the time series of Figure 4, with $D = 3$ and $\tau = 1$. The CCEP is a plane formed by the $H_S[p_\pi]$ and $C_{JS}[p_\pi]$ features, computed from the ordinal patterns probability distribution p_π, as the x-axis and the y-axis of the plane, respectively.

According to the $(H_S[p_\pi], C_{JS}[p_\pi])$ pair, different time series dynamics are expected to be placed at specific regions of the plane, bounded by minimum and maximum limits of the statistical complexity. In general, random dynamics are placed to the right of the plane, while the more deterministic behaviors are placed to the left [13], [15], [46]. For instance, the SCAN operation of Mirai botnet presents the highest deterministic dynamics, followed by the SCAN of the Bashlite botnet. While the COMBO and JUNK operations of Bashlite have a slight determinism, the other operations present a more random behavior.

This clear distinction of SCAN is an essential asset for botnet detection since it is one of their most common operations leading to applying the method for other botnets [5], [6]. However, the distinction between regular and other attack operations is not clear in the CCEP. To address this, Figure 6 presents the feasibility of our multiscale approach for a better distinction of those different operations. It illustrates the behavior of the p_{st} feature, described in Section 3.3.2.3, evaluated as a function of the embedding delay $\tau \in \{1, \ldots, 10\}$, with fixed $D = 3$. Other features are not presented to save space. It can be observed that as τ increases, the separation between the device's regular and other attack operations is easier to distinguish.

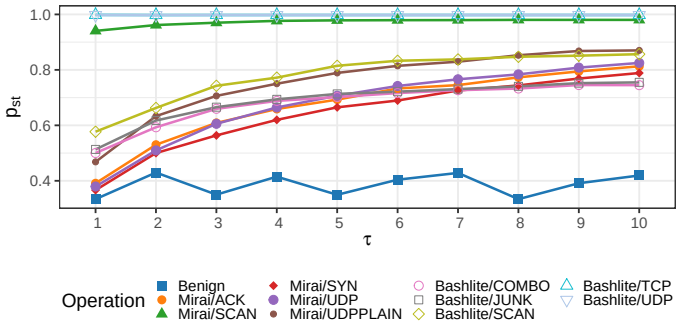

Fig. 6: Probability of self transition (p_{st}) from measure MI_dir_L5_weight, for distinct operations of a device, presented in Figure 4, evaluated for $D = 3$ and $\tau \in \{1, \ldots, 10\}$.

5.3 IoT botnet detection results

This section presents our results for the detection of botnets in IoT, by applying this multiscale approach to the samples from the N-BaIoT dataset.

5.3.1 One model for all devices

This first experiment consists of creating a single model for all dataset devices, which are used for detecting anomalies for each of them. It is based on a similar experiment made by Nomm et al. [21], used to evaluate the performance of one model created regardless of the device.

To create this model, we first have to construct time series from consecutive samples of the original dataset, as described in Section 5.1. Once the time series for both benign and attack data are constructed considering all devices, we used two-thirds of the benign time series for training phase. The remaining one-third of the benign time series plus all the attack time series are used for testing phase. This split is similar to the experiments in Meidan et al. [12], except that, since we do not have an optimization parameter phase, all the two-thirds are used for training.

Since the time series length is an important aspect to consider by the transformations, we consider evaluating the accuracy of our model for different time series lengths. Figure 7 presents the achieved accuracies for our method, combining different parameters of $D = \{3, 4\}$ and $t = \{5, 10\}$. This last parameter t is used for the maximum value of τ in the multiscale approach, i.e., $\tau_i \in \mathcal{T}$, with $i = \{1, \ldots, t\}$.

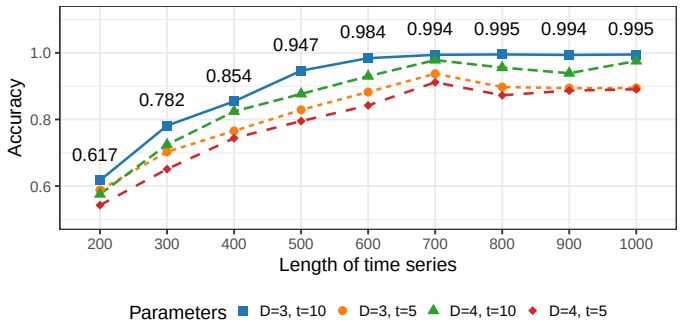

Fig. 7: Accuracy results for the one model for all devices experiment, evaluated as a function of the time series length, combinations of D and t, and maximum τ.

It can be noticed that the accuracies increase with the length m of time series presented to the method, with significant results occurring for $m \geq 500$. Also, the best results are achieved when $D = 3$ and $t = 10$, reaching the maximum value of 99.5% for larger values of m. This result beats Nomm et al. [21], which achieved a maximum of 95.6% accuracy in their similar experiment, and it is also better than Popoola et al. [32], which achieved an average accuracy of 99% with their federated learning strategy. It is also important to emphasize that reaching the best values with $D = 3$ is a fundamental aspect of the method, since larger values of D require more processing time.

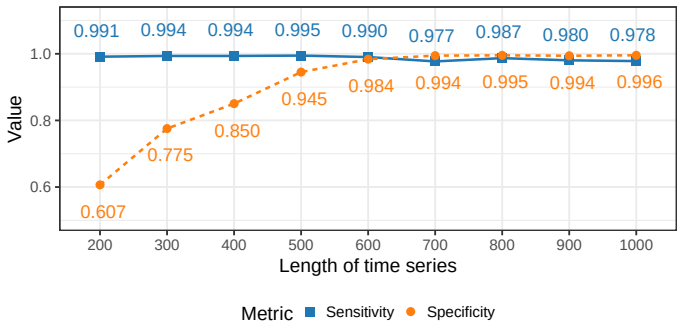

Fig. 8: Sensitivity and specificity results for the one model for all devices experiment, considering the configuration with the best overall accuracy from Figure 7 ($D = 3$ and $t = 10$), evaluated as a function of the time series length.

For the detection performance, the method's sensitivity and specificity results for the configuration with the best accuracy ($D = 3$ and $t = 10$) are presented in Figure 8. The sensitivity, which measures the proportion of benign (positives) correctly identified, is practically constant for all lengths, with a slight decay for larger values. However, the essential aspect to assess in our case is the achieved specificity, which accounts the proportion of attacks (negatives) correctly identified since it is most important for us to not miss a genuine attack. In this situation, the specificity increases with the length of time series, with significant results occurring for $m \geq 700$.

5.3.2 *One model per device*

This section follows another common experiment setup from the literature, which consists of fitting a model for each device. This analysis assesses the ability of the method to individually detect the botnet attacks considering only the data that each device produced.

Figure 9 presents the accuracy results for the one model per device experiment. The devices are identified by the same numbers as presented in Table 1. Each bar represents the average accuracy of all ten botnet attacks considered in the N-BaIoT dataset, previously described in Section 5.1, with a confidence interval at 95% of confidence level.

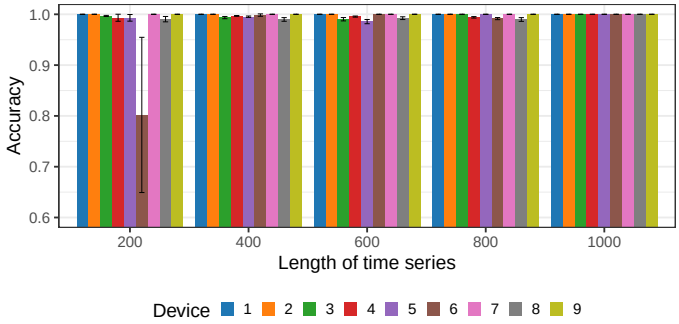

Fig. 9: Accuracy results for the one model per device experiment, evaluated as a function of the time series length, considering $D = 3$ and $t = 10$. The identification of devices is the numbers presented in Table 1.

It can be observed that, in general, the models for specific devices present significant accuracy results. However, the length of the time series is also an important aspect of this experiment. When the length is $m \geq 400$, all devices reach high accuracies, ranging from a minimum of 98.5% when $m = 600$ for the device (5) Provision PT 737E Security Camera, to a maximum of 100% for all devices when $m = 1000$.

The worst cases occur when $m = 200$, where few data hinder the models' ability to capture the temporal dynamics of attacks correctly. For instance, for device (6) Provision PT 838 Security Camera, when time series length is $m = 200$, the detection accuracy is 80.19%, with confidence interval of ± 0.21. To assure this is far beyond other results, the second-worst detection accuracy for $m = 200$ occurs for device (8) SimpleHome XCS7 1002 WHT Security Camera, with an accuracy of 99.05%, with confidence interval of ± 0.007.

However, when we take a closer look at this specific worst case, Figure 10 shows that the method can reach significant accuracies for some of the botnet attacks. This is the case for the ACK, SCAN, and SYN attacks of Mirai, and the TCP and UDP attacks of Bashlite, with more than 90% of accuracy for all of them. To infer what is happening here, we should have access to more information from the dataset collection process from Meidan et al. [12], but only the processed data is available. Given its discrepancy with other results, and since we are dealing with real-world data, it is reasonable to suggest that some spurious data was added during the data collection of this specific device.

This is valid to consider once, as the number of samples increases, the method can correctly distinguish between attack and benign data for all devices. In fact, for the devices

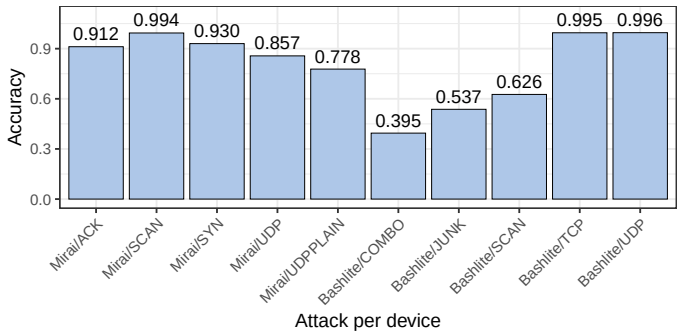

Fig. 10: Accuracy results per single attack, for the one model per device experiment, considering the worst case of Figure 9, when device is the (6) Provision PT 838 Security Camera and time series length $m = 200$.

(1) Danmini Doorbell, (2) Ecobee Thermostat, (7) Samsung SNH 1011 N 52,150 Webcam, and (9) SimpleHome XCS7 1003 WHT Security Camera, all detection accuracies are 100%, independently of the time series length.

With respect to the sensitivity and specificity of the results, considering the case with time series length $m = 1000$, since all devices correctly detected all of their attacks, both sensitivity and specificity are 100%. These results surpass the related work for similar experiments with a model per device. For instance, Meidan et al. [12] have a false positive rate of 0.007 ($1-$specificity), and Nomm et al. [21] have an average accuracy of 96.64% for all devices. Alqahtani et al. [31] also achieve 100% accuracy when performing an experiment with a model per device; however, the authors used both benign and attack data during the training phase, differently from us. While this improves their method's ability to differentiate benign and attack events correctly, it may impact their ability to detect novel attacks not seen during the training phase.

5.4 Detection time analysis

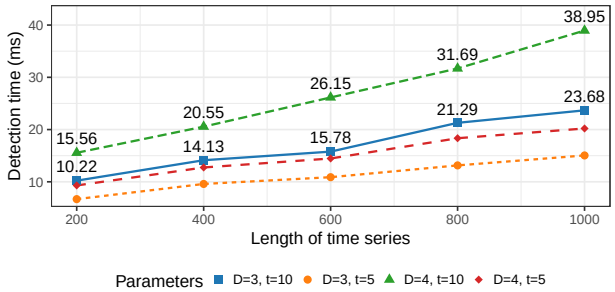

Fig. 11: Average detection time for the one model for all devices experiment, evaluated as a function of the time series length, combinations of D and maximum τ.

To evaluate the time spent by our method to detect an attack, we analyzed the time spent for the anomaly detection phase by the one model for all devices experiment as a function of the time series length. This is the most time-consuming experiment as it handles more data. Figure 11 shows the average detection time over 10 realizations, for different combinations of D and maximum τ. The linear increase on m confirms the theoretical analysis of the computational cost in Section 3.1. In the worst case, our method needs under 40 ms to detect an anomaly. However, considering the case which resulted in better accuracies, when $D = 3$, the maximum $\tau = 10$, and the time series length is $m = 1000$, the detection time is under 24 ms.

Another aspect to consider in this analysis is the time spent extracting the measure and constructing the time series. For instance, for $m = 1000$ and for the chosen measure, which needs 100 ms windows to compute the number of packets from a given device, we need another 100 seconds to construct the time series before the transformation step. In fact, the related work does not consider the time spent to extract measures when evaluating their detection time. Thus, since all other methods require network measures with larger time windows, they all need at least 1 minute to compute their measures and then perform their detection.

6 Conclusion

In this work, we presented a solution for the detection of botnet attacks in IoT by identifying anomalies in the temporal dynamics of their devices. Their dynamics were obtained by features extracted after the multiscale ordinal patterns transformation, from time series of the number of packets transmitted by devices. Using features computed during the devices' regular operation, we trained an Isolation Forest model to detect anomalies when presented to features computed from the devices are under attack.

To evaluate our proposal, two main experiments were conducted using the N-BaIoT dataset: the first consists of creating a single model for all dataset devices, and the second consists of fitting a specific model per device. While the first experiment is used to evaluate the detection of anomalies regardless of the device, the second analyzes the method's ability to individually detect the botnet attacks considering only the data each device produced.

Our results show that the proposed method can reach significant detection accuracies for both experiments, overcoming other strategies from literature. For the first experiment, our best results achieved a maximum 99.5% accuracy, beating the related work, which achieved a maximum 95.6% accuracy in their similar experiment. For the second experiment, we reached the maximum accuracy of 100%. Other related work also achieved this accuracy, but they used both regular and attack data for training the model, which may reduce their ability to detect novel attacks.

We obtained these results when the time series length was $m = 1000$, indicating that its value is fundamental to the transformation process, i.e., this is an important aspect to consider in our method. This is not a drawback since both best results are also obtained when $D = 3$, which requires less processing time. As future work, we plan to investigate alternatives to reduce this dependence on large time series. Furthermore, future solutions should consider transfer learning and online strategies to follow the dynamical evolution of the botnets better, which our proposal can facilitate being another advantage of our method. Finally, another future work includes the application of the proposed method to other datasets, considering other botnets, to assure its viability to different scenarios.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Computer Networks*, vol. 54, pp. 2787–2805, 2010.

[2] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Fut. Gen. Comp. Systems*, vol. 29, pp. 1645–1660, 2013.

[3] J. B. Borges, H. S. Ramos, R. A. F. Mini, A. C. Viana, and A. A. F. Loureiro, "The Quest for Sense: Physical phenomena Classification in the Internet of things," in *15th Int. Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE, 2019, pp. 701–708.

[4] S. Schmeißer and G. Schiele, "coSense: The Collaborative Sensing Middleware for the Internet-of-Things," *ACM/IMS Transactions on Data Science*, vol. 1, pp. 1–21, 2020.

[5] E. Bertino and N. Islam, "Botnets and Internet of Things Security," *Computer*, vol. 50, pp. 76–79, 2017.

[6] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and Other Botnets," *Computer*, vol. 50, pp. 80–84, 2017.

[7] A. Blaise, M. Bouet, V. Conan, and S. Secci, "Detection of zero-day attacks: An unsupervised port-based approach," *Computer Networks*, vol. 180, 2020.

[8] S. G. Abbas, S. Zahid, F. Hussain, G. A. Shah, and M. Husnain, "A Threat Modelling Approach to Analyze and Mitigate Botnet Attacks in Smart Home Use Case," in *14th Intl. Conf. on Big Data Science and Engineering (BigDataSE)*. IEEE, 2020, pp. 122–129.

[9] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security," *IEEE Communications Surveys & Tutorials*, vol. 22, pp. 1646–1685, 2020.

[10] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving Into Internet DDoS Attacks by Botnets: Characterization and Analysis," *IEEE/ACM Transactions on Networking*, vol. 26, pp. 2843–2855, 2018.

[11] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers and Security*, vol. 28, pp. 18–28, 2009.

[12] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 17, pp. 12–22, 2018.

[13] O. A. Rosso, H. A. Larrondo, M. T. Martin, A. Plastino, and M. A. Fuentes, "Distinguishing Noise from Chaos," *Physical Review Letters*, vol. 99, p. 154102, 2007.

[14] O. A. Rosso, F. Olivares, and A. Plastino, "Noise versus chaos in a causal Fisher-Shannon plane," *Papers in Physics*, p. 070006, 2015.

[15] J. B. Borges, H. S. Ramos, R. A. Mini, O. A. Rosso, A. C. Frery, and A. A. Loureiro, "Learning and distinguishing time series dynamics via ordinal patterns transition graphs," *Applied Mathematics and Computation*, vol. 362, p. 124554, 2019.

[16] C. Bandt and B. Pompe, "Permutation Entropy: A Natural Complexity Measure for Time Series," *Physical Review Letters*, vol. 88, p. 174102, 2002.

[17] A. L. L. Aquino, T. S. G. Cavalcante, E. S. Almeida, A. C. Frery, and O. A. Rosso, "Characterization of vehicle behavior with information theory," *Euro. Physical Journal B*, vol. 88, p. 257, 2015.

[18] A. L. L. Aquino, H. S. Ramos, A. C. Frery, L. P. Viana, T. S. G. Cavalcante, and O. A. Rosso, "Characterization of electric load with Information Theory quantifiers," *Physica A: Statistical Mechanics and its Applications*, vol. 465, pp. 277–284, 2017.

[19] H. V. Ribeiro, M. Jauregui, L. Zunino, and E. K. Lenzi, "Characterizing time series via complexity-entropy curves," *Physical Review E*, vol. 95, p. 062106, 2017.

[20] L. Zunino, M. C. Soriano, and O. A. Rosso, "Distinguishing chaotic and stochastic dynamics from time series by using a multiscale symbolic approach," *Physical Review E*, vol. 86, p. 46210, 2012.

[21] S. Nomm and H. Bahsi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," in *17th IEEE Intl. Conf. on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 1048–1053.

[22] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Eighth IEEE International Conf. on Data Mining*. IEEE, 2008, pp. 413–422.

[23] ——, "Isolation-Based Anomaly Detection," *ACM Transactions on Knowledge Discovery from Data*, vol. 6, pp. 1–39, 2012.

[24] S. Hariri, M. Carrasco Kind, and R. J. Brunner, "Extended Isolation Forest," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, pp. 1–1, 2019.

[25] J. P. S. Medeiros, A. M. Brito, and P. S. Motta Pires, "An Effective TCP/IP Fingerprinting Technique Based on Strange Attractors Classification," in *LNCS*, 2010, vol. 5939 LNCS, pp. 208–221.

[26] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A Survey," *ACM Computing Surveys*, vol. 41, pp. 1–58, 2009.

[27] A. Boukerche, L. Zheng, and O. Alfandi, "Outlier Detection: Methods, Models, and Classification," *ACM Computing Surveys*, vol. 53, pp. 1–37, 2020.

[28] B. Agarwal and N. Mittal, "Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques," *Procedia Technology*, vol. 6, pp. 996–1003, 2012.

[29] J. Yu, H. Kang, D. Park, H.-C. Bang, and D. W. Kang, "An in-depth analysis on traffic flooding attacks detection and system using data mining techniques," *Journal of Systems Architecture*, vol. 59, pp. 1005–1012, 2013.

[30] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *Proceedings 2018 Network and Distributed System Security Symposium*. Reston, VA: Internet Society, 2018.

[31] M. Alqahtani, H. Mathkour, and M. M. Ben Ismail, "IoT Botnet Attack Detection Based on Optimized Extreme Gradient Boosting and Feature Selection," *Sensors*, vol. 20, p. 6336, 2020.

[32] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated Deep Learning for Zero-Day Botnet Attack Detection in IoT Edge Devices," *IEEE Internet of Things Journal*, vol. XX, pp. 1–1, 2021.

[33] O. A. Rosso, F. Olivares, L. Zunino, L. De Micco, A. L. L. Aquino, A. Plastino, and H. A. Larrondo, "Characterization of chaotic maps using the permutation Bandt-Pompe probability distribution," *The European Physical Journal B*, vol. 86, p. 116, 2013.

[34] O. A. Rosso, R. Ospina, and A. C. Frery, "Classification and Verification of Handwritten Signatures with Time Causal Information Theory Quantifiers," *PLOS ONE*, vol. 11, p. e0166868, 2016.

[35] M. McCullough, M. Small, T. Stemler, and H. H.-C. Iu, "Time lagged ordinal partition networks for capturing dynamics of continuous dynamical systems," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, p. 53101, 2015.

[36] C. W. Kulp, J. M. Chobot, H. R. Freitas, and G. D. Sprechini, "Using ordinal partition transition networks to analyze ECG data," *Chaos*, vol. 26, p. 73114, 2016.

[37] T. Sorrentino, C. Quintero-Quiroz, A. Aragoneses, M. C. Torrent, and C. Masoller, "Effects of periodic forcing on the temporally correlated spikes of a semiconductor laser with feedback," *Optics Express*, vol. 23, p. 5571, 2015.

[38] J. Zhang, J. Zhou, M. Tang, H. Guo, M. Small, and Y. Zou, "Constructing ordinal partition transition networks from multivariate time series," *Scientific Reports*, vol. 7, p. 7795, 2017.

[39] J. Zhang, J. Sun, X. Luo, K. Zhang, T. Nakamura, and M. Small, "Characterizing pseudoperiodic time series through the complex network approach," *Physica D: Nonlinear Phenomena*, vol. 237, pp. 2856–2865, 2008.

[40] C. W. Kulp and S. Smith, "Characterization of noisy symbolic time series," *Physical Review E*, vol. 83, p. 26201, 2011.

[41] J. Tang, Y. Wang, and F. Liu, "Characterizing traffic time series based on complex network theory," *Physica A: Statistical Mechanics and its Applications*, vol. 392, pp. 4192–4201, 2013.

[42] B. A. Gonçalves, L. Carpi, O. A. Rosso, and M. G. Ravetti, "Time series characterization via horizontal visibility graph and Information Theory," *Physica A: Statistical Mechanics and its Applications*, vol. 464, pp. 93–102, 2016.

[43] P. W. Lamberti, M. T. Martin, A. Plastino, and O. A. Rosso, "Intensive entropic non-triviality measure," *Physica A: Statistical Mechanics and its Applications*, vol. 334, pp. 119–131, 2004.

[44] O. A. Rosso, L. Zunino, D. G. Pérez, A. Figliola, H. A. Larrondo, M. Garavaglia, M. T. Martín, and A. Plastino, "Extracting features of Gaussian self-similar stochastic processes via the Bandt-Pompe approach," *Physical Review E*, vol. 76, p. 061114, 2007.

[45] P. Sanchez-Moreno, R. J. Yanez, and J. Dehesa, "Discrete Densities and Fisher Information," in *Diff. Eqs. and Appls.*, 2009, pp. 291–298.

[46] M. G. Ravetti, L. C. Carpi, B. A. Gonçalves, A. C. Frery, and O. A. Rosso, "Distinguishing Noise from Chaos: Objective versus Subjective Criteria Using Horizontal Visibility Graph," *PLoS ONE*, vol. 9, p. e108004, 2014.

[47] F. Olivares, M. Zanin, L. Zunino, and D. G. Pérez, "Contrasting chaotic with stochastic dynamics via ordinal transition networks," *Chaos*, vol. 30, p. 063101, 2020.

[48] R. C. Team, "A Language and Environment for Statistical Computing," Vienna, Austria, pp. https://www.R–project.org, 2018.