future internet

MDPI

Article

An Automated Behaviour-Based Clustering of IoT Botnets

Tolijan Trajanovski * and Ning Zhang

School of Computer Science, University of Manchester, Kilburn Building, Manchester M13 9PL, UK; ning.zhang-2@manchester.ac.uk

* Correspondence: tolijan.trajanovski@manchester.ac.uk

Abstract: The leaked IoT botnet source-codes have facilitated the proliferation of different IoT botnet variants, some of which are equipped with new capabilities and may be difficult to detect. Despite the availability of solutions for automated analysis of IoT botnet samples, the identification of new variants is still very challenging because the analysis results must be manually interpreted by malware analysts. To overcome this challenge, we propose an approach for automated behaviour-based clustering of IoT botnet samples, aimed to enable automatic identification of IoT botnet variants equipped with new capabilities. In the proposed approach, the behaviour of the IoT botnet samples is captured using a sandbox and represented as behaviour profiles describing the actions performed by the samples. The behaviour profiles are vectorised using TF-IDF and clustered using the DBSCAN algorithm. The proposed approach was evaluated using a collection of samples captured from IoT botnets propagating on the Internet. The evaluation shows that the proposed approach enables accurate automatic identification of IoT botnet variants equipped with new capabilities, which will help security researchers to investigate the new capabilities, and to apply the investigation findings for improving the solutions for detecting and preventing IoT botnet infections.

Keywords: IoT botnets; malware analysis; clustering; behavioural analysis

Citation: Trajanovski, T.; Zhang, N. An Automated Behaviour-Based Clustering of IoT Botnets. *Future Internet* **2022**, *14*, 6. https://doi.org/10.3390/fi14010006

Academic Editors: Christos Tryfonopoulos and Skiadopoulos Spiros

Received: 26 November 2021
Accepted: 20 December 2021
Published: 23 December 2021

1. Introduction

An Internet of Things (IoT) botnet is a network of IoT devices infected by botnet malware. A botnet malware is a self-propagating malware, capable of automatically identifying and infecting vulnerable devices on the Internet. The infected devices, or bots, can be instructed by the botnet owner to execute numerous malicious actions, including Distributed Denial of Service (DDoS) attacks [1], credential theft, fake social media endorsement [2], etc. The widespread deployment of poorly secured IoT devices has resulted in unprecedented IoT botnet propagation rates [3]. The Satory botnet is estimated to have infected 280,000 IoT devices in just 12 h, while the Mirai botnet variants have enslaved between 800,000 and 2,500,000 IoT devices [4]. The large-scale of these botnets can be leveraged for executing high-impact attacks, such as the infamous DDoS attack against the DNS provider Dyn [5], which caused outages of Twitter, Reddit and Netflix.

The leaked source-codes of popular IoT botnets, such as Gafgyt and Mirai [6] have facilitated the proliferation of different IoT botnet variants. The changes applied to the botnet variants may be minor or significant. The minor changes include modification of the botnet configuration settings, such as IP addresses of the command and control (C2) and malware distribution (MD) servers, names of the botnet samples, firewall commands, addition or removal of exploits which are not rare or unknown, etc. As significant changes, we consider the addition of new capabilities for more effective propagation, persistence, detection evasion, and prevention of infection remedy. Such capabilities include infection vectors, as well as techniques for establishing persistence, evading detection, and preventing infection remedy, which have not been previously observed in other botnets. The botnet variants equipped with new capabilities may be more challenging to detect.

To identify IoT botnet variants, it is necessary to analyse the samples captured from IoT botnets propagating on the Internet. However, the large number of IoT botnet samples collected by antivirus vendors makes it impossible for the malware analysts to examine each botnet sample [7]. Although significant efforts have been made for automating the analysis of IoT botnet samples using sandboxes [8–10], the analysis results still need to be interpreted by malware analysts. This challenge can be overcome by grouping together samples that exhibit similar behaviours into clusters. After the samples are clustered, the malware analyst would need to examine only one sample per cluster to identify the distinct capabilities of that cluster.

Over the years, multiple approaches for clustering IoT botnet samples have been proposed. The proposed approaches use different features, such as botnet configuration parameters [1], multidimensional features extracted from strings-based and image-based botnet binary representations [11], Function Call Sequence Graphs (FCSG) [12], strings [13], sizes of compressed botnet binaries [14], Trend Micro Locality Sensitive Hashes (TLSH) [15], and a combination of static and dynamic features [16]. However, these features are extracted or computed using static analysis and disassembly of the botnet samples. As a result, the proposed approaches may not be effective for clustering obfuscated samples. In our recent study [8], we observed that 38% of the samples captured by our honeypots were packed, while 59% had encoded strings. Therefore, we argue that to identify botnet variants equipped with new capabilities, both obfuscated and non-obfuscated samples should be effectively clustered. Another limitation of some of the proposed approaches is that they were evaluated using samples compiled for only one CPU architecture.

Different from the existing approaches, the approach taken in the work reported in this paper uses behaviour profiles generated via a behavioural analysis of the botnet samples. The approach was evaluated using both obfuscated and non-obfuscated IoT botnet samples, compiled for different CPU architectures. The behaviour profiles are text files describing the actions executed by the botnet samples. To identify clusters representing botnet variants, the behavioural profiles are first transformed into feature vectors using the Term Frequency-Inverse Document Frequency (TF-IDF) method [17], and then clustered with the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [18]. The similarity between the TF-IDF vectors is measured using the cosine similarity metric. To the best of our knowledge, and relying on our extensive literature search on published work, no similar proposal has been made until now. In summary, this paper makes the following contributions:

- It presents an automated approach for capturing the behaviour of IoT botnet samples. The behaviour is recorded by tracing the system calls and capturing the network traffic. The captured behaviour is abstracted as a set of actions executed by the samples.
- It describes the challenges that may affect the behavioural analysis of botnet samples and the actions taken to address them.
- It presents a novel approach for automated clustering of IoT botnet samples based on their behaviour profiles.
- It provides the evaluation of the proposed clustering approach using a collection of samples captured from IoT botnets propagating on the Internet. The evaluation shows that the proposed approach has successfully identified IoT botnet variants equipped with new capabilities.

The research method used consists of literature review identifying knowledge gaps and areas of improvement, collection of botnet samples using honeypots and a malware tracking service, execution and behavioural analysis of the collected samples, investigation of factors affecting the capturing of IoT botnet behaviour, investigation of methods for extracting features from text documents, and evaluation of different algorithms for clustering the extracted features.

The rest of this paper is organised as follows: Section 2 describes the IoT botnet behaviour and how it can be captured using sandboxes. Section 3 provides information on the work related to this study. Section 4 details the proposed clustering approach and its implementation. Section 5 presents the evaluation of the proposed approach using real-world IoT botnet samples, and provides discussion on the key findings. Lastly, Section 6 contains final conclusions.

2. IoT Botnet Behaviour

The IoT botnet behaviour can be described as the set of actions executed by the botnet samples. To better understand these actions, we first look at the IoT botnet operation.

2.1. IoT Botnet Operation

The IoT botnet operation has two main goals, botnet growth and botnet monetization. The botnet growth is achieved by identifying and infecting vulnerable IoT devices. An IoT botnet infection typically involves two steps: (1) exploiting a vulnerability to obtain access to the victim device and (2) downloading and executing bot malware. The bot malware is downloaded from a MD server configured by the botnet herder. When the malware is executed, it connects to a C2 server and starts listening for instructions sent by the botnet herder. After connecting to the C2 server, the bot malware usually starts self-propagating by scanning the Internet for vulnerable IoT devices and infecting the discovered vulnerable devices. Alternatively, it may report the discovered vulnerable devices to a loader server which is used by the botnet herder for executing infection attacks.

In addition, the bot malware can also perform multiple actions to preserve the control over the enslaved devices. Examples of such actions include (a) establishing persistence on the infected device by modifying the initialisation system configuration, (b) detecting and killing competing malware by enumerating the running processes, (c) preventing re-infection by blocking the open port used to exploit the vulnerable service, (d) preventing infection remedy by blocking remote access to the device, and (e) removing and hiding infection evidence to evade detection.

Over the years, the botnet herders have monetized IoT botnets in different ways. The most common IoT botnet monetization is through distributed DDoS attacks offered as a service by the botnet herder. The bot malware typically supports multiple different DDoS attack types [19]. To launch an attack, the DDoS service clients specify the target host, one of the supported DDoS attack types and the attack duration. Other ways of monetizing IoT botnets include crypto-currency mining [20], credential theft [21], social media fraud [2], etc.

2.2. Capturing IoT Botnet Behaviour

The behaviour of the IoT botnet samples can be captured by executing them inside a sandbox. A sandbox is a controlled environment comprised of one or more virtual machines (VMs) equipped with tools for execution tracing and network traffic capturing. The sandbox records the behaviour of a botnet sample during its execution on one of the VMs. However, the botnet behaviour capturing through sandbox execution may be challenged by the following factors:

1. **Anti-sandbox Techniques**. The sandbox execution of botnet samples may be challenged by anti-sandbox techniques. A sample equipped with sandbox detection capability may halt its execution or hide its true behaviour if it detects that it is being run inside a sandbox [22]. A botnet sample may be able to detect and disrupt execution tracing and traffic capturing tools to prevent the capturing of its behaviour [9]. Another anti-sandbox technique that may be used by IoT botnets is delayed execution. For instance, a botnet sample can be configured to stay idle for two minutes after it is executed to avoid discovery of the C2 server. To ensure a successful execution of the botnet samples, the execution duration should not be too short, and the sandbox should detect, avoid, and report the use of anti-sandbox techniques.

2. **IoT Botnets Heterogeneity**. A botnet may infect IoT devices with different CPU architectures or with the same CPU architecture. In our recent study of IoT botnets [8], we identified that approximately one third of the analysed IoT botnets infected only a single CPU architecture. To be able to capture the behaviour of the botnets infecting a single CPU architecture, the sandbox should support the CPU architectures which are the most targeted by IoT botnets. Furthermore, a botnet sample may require software tools and libraries that are expected to be available on the vulnerable devices [10]. If the sandbox lacks some of the required software, the sample may fail to execute or may execute partially. Therefore, to increase the chances for a successful sample execution, the sandbox should provide the software tools and libraries that may be required by IoT botnets.
3. **Time of Sample Execution**. It is important the botnet to be active when the sample is executed to ensure that the sample can exhibit all its functionalities, including the C2 communication. Moreover, some IoT botnet infection attacks may include a stage-two payload. In such case, the MD server hosting the stage-two payload must be available during the sample execution for the infection to complete. If a sample is executed long after it was captured, there is a high probability that the C2 and MD servers have been suspended or their configuration has been changed [23]. Thus, the botnet samples should be executed in the sandbox as soon as they are captured or discovered.

3. Related Work

Over the past years, multiple approaches for clustering IoT botnets and for investigating their variation have been proposed. The proposed approaches and their key characteristics are presented in Table 1.

The authors of [1] present different schemes for classifying and tracking Mirai botnet variants using three artefacts: the botnet configuration including the C2 settings and the encryption key, the supported DDoS attack methods, and the dictionary of usernames and passwords used in brute-force attacks. The artefacts are extracted automatically using static analysis and emulation. This study, however, is only concerned with Mirai IoT botnet variants that perform DDoS attacks.

In [14], the authors propose an automated approach for clustering IoT botnet samples using NCD distance as a similarity measure. To achieve faster clustering, the authors present an algorithm for efficiently constructing a phylogenetic tree by reducing the compression attempts. The proposed clustering approach was evaluated using botnet samples compiled for only one CPU architecture. An approach for clustering IoT botnet samples using TLSH fuzzy hash as similarity measure is proposed in [15]. The authors evaluate the performance of two clustering algorithms, k-medoids and OPTICS, and propose a new clustering algorithm which achieves a performance superior to both k-medoid and OPTICS. Nonetheless, the proposed clustering approach was evaluated using only non-packed and non-encrypted samples, compiled for the ARM CPU architecture. The authors of [13] propose an approach for clustering IoT botnet samples and investigating their underlying correlations using strings-based similarity. The strings are extracted from the botnet binary samples using static analysis. Natural Language Processing (NLP) techniques such as word tokenization are then applied to process the identified strings and extract meaningful words. The meaningful words are compared using a combination of the Jaccard and overlap similarity coefficients. The ClusterONE algorithm is used to investigate correlated malware samples and to identify groups of similar IoT malware implementation. A novel approach for effective IoT botnet classification and family attribution by combining multi-dimensional features extracted from strings-based and image-based botnet binary representations is presented in [11]. The features are extracted automatically using Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) deep learning models. However, the approaches proposed in [11,13–15] may not effectively cluster packed or obfuscated samples since they rely on static analysis for extracting the required features.

Table 1. Related work concerned with the clustering of IoT botnets.

Proposed Approach	Automated or Manual	Operations Performed	Data Used	Features
[1]	Automated	Static analysis and emulation	DDoS IoT botnets	Configuration; Supported DDoS attack methods; Brute-force dictionary
[14]	Automated	Compression	IoT botnet samples from same CPU architecture grouped in 3 clusters	NCD as similarity measure between binary samples
[15]	Manual	TLSH fuzzy hash calculation	Non-packed ARM botnet samples	TLSH as similarity measure
[24]	Manual	Dynamic analysis	Various IoT botnet samples	System calls
[12]	Manual	Static analysis	Various IoT botnet samples	FCSG
[13]	Manual	Static analysis	Various IoT botnet samples	Strings-based features (IP addresses and/or embedded commands/payloads)
[25]	Automated	Static analysis	IoT botnet samples for multiple architectures expect Intel and AMD	Program code
[11]	Automated	Static analysis	Various IoT botnet samples	Features extracted from string-based and image-based representations of the executable binaries
[26]	Mixed	Static analysis	Honeypot logs and IoT botnet samples	Honeypot login sessions and executable binaries
[16]	Mixed	Static and dynamic analysis	Various Linux malware samples	Static, dynamic and hybrid features

A method for identifying behavioural differences between IoT botnet samples using FCSG is proposed in [12]. The FCSG are generated from the disassembled code of botnet samples. Therefore, the proposed method requires the botnet samples to be disassembled. In [25], the authors propose a clustering approach based on code-level similarity for tracking the evolution over time of a given IoT botnet family as well as the code reuse and functionalities borrowed among different IoT botnet families. Similar to [12], this approach also requires the botnet samples to be disassembled. However, if a sample is packed, it needs to be properly unpacked before it is disassembled. In addition, the unpacking of samples packed with custom packers can be challenging and time-consuming. Thus, the approaches presented in [12,25] may not be effective for clustering packed botnet samples. The authors of [16] propose a clustering approach using a custom distance function that relies on three types of features: static, dynamic and hybrid. The hybrid features are a combination of the static and dynamic features. However, in this study, the packed samples were discarded from the dataset since the method used for extracting static features from the botnet samples is not effective for packed samples. Botnet families represented with less than ten samples were also removed from the dataset.

In [24], the authors evaluate different algorithms for clustering IoT botnet samples based on their behaviour represented as system calls. The system calls are captured via a sandbox execution of the botnet samples. The features used by the clustering algorithms

are extracted from the system call logs using the 2-g method discussed in the paper and reduced using PCA. The similarity between feature vectors is measured using Euclidean distance. The clustering algorithms evaluated in this study are DBSCAN, Mean-shift, and Hierarchical clustering. The Mean-shift algorithm achieved the best performance.

The authors of [26] propose an approach for measuring the variation among IoT botnets based on infection attacks recorded by honeypots. A list of arguments from the executed commands is created for each attack session in which a botnet binary was created or downloaded. The similarity between the attack sessions is measured as a Levenshtein distance between the argument lists. To better understand the scope of the IoT botnet infection attacks, the authors also investigate the relationship between the IP address of the attacker and the hash of the downloaded file(s) for each attack session.

Many of the related studies propose the use of features extracted via static analysis for clustering IoT botnets, based on the observation that the IoT malware obfuscation is not as common as the Windows malware obfuscation. However, in our recent study [8], concerned with the analysis of IoT botnet samples captured by honeypots, we observed that a significant proportion of the analysed samples were obfuscated. Approximately 38% of the analysed samples were packed, 25% of which were packed with custom packers. In addition, string encoding, a simpler form of obfuscation, was identified in 59% of the samples. The obfuscation can prevent the static analysis methods, such as code disassembly and string extraction, from identifying valuable features. As a result, the obfuscated samples may not be effectively clustered. Another limitation of the clustering using static features is that some of the static features may differ between different CPU architectures [16].

We argue that to facilitate the identification of IoT botnet variants equipped with new capabilities, the clustering must be effective for both obfuscated and non-obfuscated samples. To effectively cluster both obfuscated and non-obfuscated samples, our approach uses features extracted via behavioural analysis of the botnet samples. Similar to [24], we record the botnet samples behaviour by tracing the system calls. However, different from [24], the system calls in our solution are abstracted to actions and objects on which the actions are performed. This is because the traces of system calls can vary significantly, and the need for abstracting the system call traces in such cases was discussed in [27]. Furthermore, the behaviour profiles describing the identified actions are significantly smaller than the system call traces, allowing a more efficient clustering of a large number of samples.

4. The Proposed Approach

This section describes the proposed approach for an automated behaviour-based clustering of IoT botnets. We first provide an overview of the approach and the steps it consists of.

4.1. Overview

The proposed approach consists of four steps, as shown in Figure 1 below:

1. **Obtaining IoT Botnet Samples**. The IoT botnet samples are obtained from honeypots deployed on the Internet and from URLhaus, a service for tracking malware URLs.
2. **Sandbox Execution and Analysis**. The obtained IoT botnet samples are executed inside a sandbox while the botnets are actively propagating on the Internet. Their behaviour is recorded by tracing the system calls, and by capturing the network traffic in a network traffic capture file (pcap). Because the system call traces can vary significantly, the captured behaviour is abstracted at higher level, as a set of actions and objects the actions were performed on. The set of actions and corresponding objects is identified by analysing the recorded system calls and the network traffic capture file, respectively.
3. **Creating Behaviour Profiles**. A behaviour profile containing the actions identified by the sandbox analysis is created for each sample executed in the sandbox. The behaviour profiles are text files describing the actions performed by the sample

and the objects on which the actions were performed. They serve as input to the clustering algorithm.

4. **Clustering**. The clustering of behaviour profiles involves three steps. The behaviour profiles are first transformed into feature vectors using TF-IDF. The pairwise similarities between the behaviour profiles in the dataset are then computed as the cosine similarity between their respective TF-IDF vectors. The computed pairwise similarities are stored in a similarity matrix. Finally, the DBSCAN clustering algorithm is run using the pairwise similarity matrix as input.

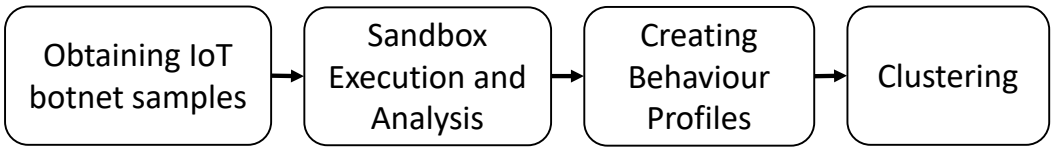

Figure 1. Steps comprising the proposed approach for automated behaviour-based clustering of IoT botnets.

4.2. Obtaining IoT Botnet Samples

The IoT botnet samples are obtained from two sources, honeypots we deployed on the Internet and URLhaus [28], a malware tracking service. The honeypots automatically submit the botnet samples for sandbox analysis as soon as they are captured. This ensures that the botnet is active when the captured samples are executed in the sandbox. The URLhaus service tracks URLs of malware samples reported by malware researchers. The URLs can be queried using specific filters such as file type. URLhaus also periodically checks if the MD servers hosting the reported malware samples are online, as can be seen in Figure 2. To obtain the samples from URLhaus, we implemented a program in Python which periodically queries the service for URLs of Linux malware samples. To do so, the program uses the file type filter to search for Linux executable files in the ELF format, and the URL status filter to make sure the queried URLs are reachable. The program downloads the botnet samples from the queried URLs and checks if the CPU architecture the samples are compiled for is supported by the sandbox. Finally, it submits the supported botnet samples for sandbox analysis.

Malware URL	Status
http://49.70.81.141:60277/Mozi.m	Online
http://46.237.1.17:48710/Mozi.a	Online
http://222.137.196.61:48252/Mozi.m	Online

Figure 2. URLhaus malware tracking service.

4.3. Sandbox Execution and Analysis

The IoT botnet samples are executed using the ELF DIGEST sandbox, presented in our previous work [8]. The ELF DIGEST sandbox supports ARMv5, ARMv7, MIPS, x86 and x64 CPU architectures, as well as older (v.3) and more recent (v.4+) Linux kernel versions. The sandbox can detect the use of anti-sandbox techniques. To increase the chances for a successful botnet sample execution, the VMs comprising the sandbox include software tools and libraries that may be used by IoT botnets [8]. The botnet samples are executed for 310 s to accommodate for a potential use of delayed execution.

The on-host behaviour of the samples is recorded using the program execution tracing tools systemtap [29] and strace [30], while the network traffic is captured using tcpdump [31]. The tracing tools record the system calls made by the botnet sample. A system call is a request made by a program to the Linux kernel for a specific service. There are

various system calls with different purposes [9], such as executing a program, allocating memory, writing to disk or network socket, killing processes, etc. However, a behaviour clustering using system call traces as input may be ineffective because the system call traces can vary significantly, as discussed in [27]. For instance, a program may invoke the 'read' system call to read 256 bytes from a file at once, or may invoke the 'read' system call 256 times, reading one byte per system call. Thus, the same behaviour may result in different system call traces. We argue that the behaviour of the botnet samples can be described more effectively at a higher abstraction level, as a set of actions and objects the actions were performed on.

To identify the actions performed by a botnet sample, the sandbox performs behavioural and network analyses on the recorded system calls and the captured network traffic, respectively. The behavioural and network analyses are performed automatically, after the sample execution is finished. The behavioural analysis can identify actions such as reading, writing, and removing files, executing programs and shell commands, loading of kernel modules, use of persistence techniques, changes made to the firewall configuration, etc. The network analysis can identify DNS queries, HTTP requests, port scanning and C2 communication.

4.4. Creating Behaviour Profiles

After the two analyses performed by the sandbox are finished, a behaviour profile containing the identified actions is created for each botnet sample. The behaviour profile is a text file describing the actions executed by the botnet sample. An action is described as a pair of action and the object on which the action was performed, as can be seen in Table 2.

Table 2. Actions comprising the behaviour profiles.

Action	Object
read/write	file
execute	program/shell command + argument(s)
HTTP request	URI
DNS query	domain
connects to C2 server	domain:port/IP:port + protocol
scans port	port number + protocol
persistence	persistence technique
firewall change	firewall configuration
loaded	kernel module

The files and programs are described using their absolute paths. The recorded executions of programs and shell commands also include the execution arguments. The URIs of the HTTP requests made by a botnet sample are reported in the behaviour profile. The URI capturing can help identify: (1) vulnerability exploitation over HTTP; (2) C2 communication over HTTP; and (3) stage-two payloads, as shown in Listing 1.

Listing 1. URIs of captured HTTP requests.

```
%\begin{lstlisting}[language=html]
%\caption{URIs of captured HTTP requests.\label{listing1}}
1) /login.cgi?cli=aa%20aa%27;wget%20http://176.123.4.234/Dlinkrep.sh%20-O
    %20-%3E%20/tmp/kh;Dlinkrep.sh%20/tmp/kh%27$
2) /YouWorkForYama/BBC.php?key=key1&option=port
3) /wrgjwrgjwrg246356356356/n7
```

The C2 servers are described as a triplet of IP address, port and protocol or domain, port and protocol. A persistence technique is presented as the change made to the system to ensure the bot malware will continue running after a reboot or a power-cycle. The persistence techniques typically involve scheduling the botnet malware execution using cron or modifying the initialisation system configuration to execute the malware at system start-up. The changes to firewall configuration vary from disabling the firewall completely

to altering the firewall rules to allow or block connections on specific port. The loaded kernel modules are described using absolute paths. An excerpt from a behaviour profile is shown in Listing 2. The behaviour profiles serve as input to the clustering algorithm.

Listing 2. Behaviour profile excerpt.

```
scans port 23 tcp
scans port 80 tcp
C2 tcp 185.132.53.104:1024
wrote to /dev/misc/watchdog
executed /tmp/UnHAnaAW.arm
firewall /bin/sh, [sh, -c, iptables -I OUTPUT -p tcp --source-port 22 -j DROP
    ]
```

4.5. Clustering

The behaviour-based clustering of IoT botnet samples implies identifying groups of botnet samples with similar behaviour profiles. Thus, to cluster the samples it is necessary to measure the similarity between the behaviour profiles. Since the behaviour profiles are text documents, their similarity can be measured by transforming them into feature vectors and measuring the similarity between their feature vector representations.

We considered two techniques for vectorising text documents, namely, TF-IDF and Doc2Vec. The TF-IDF is a statistic which reflects how important a word is to a document in a corpus of documents. It is computed by multiplying the TF and IDF metrics. The TF indicates how often a word appears in a document, while IDF reflects how common or rare a word is across the corpus. Doc2Vec, also known as paragraph2vec, computes a feature vector for every document in a corpus, irrespective to its length. It expands the Word2Vec technique [32], used for vectorising words, with another vector referred to as paragraph ID [33]. To select the more feasible of the two techniques for vectorising the behaviour profiles, we investigated the characteristics of the behaviour profiles and identified that they are relatively short, with an average length of 221 characters. A study investigating the effectiveness of text similarity measuring techniques on short-length text documents indicates that Doc2Vec may have poor performance for short-length documents [34]. Thus, we used TF-IDF to vectorise the behaviour profiles.

To cluster the TF-IDF vectors, it is necessary to select a metric for measuring the similarity between a pair of TF-IDF vectors. The botnet samples may execute different number of actions, resulting in behaviour profiles with different lengths. The length of the behaviour profiles in the corpus varies in the range from 221 characters to 21,418 characters. Thus, the similarity between behaviour profiles should be computed irrespective of their length. Considering this requirement, we selected the cosine similarity metric for measuring the similarity between the TF-IDF fectors, since it measures the cosine of the angle between the vectors, and not their magnitude.

Because there is no prior knowledge about the botnet samples comprising the dataset, the number and the shape of the clusters cannot be identified beforehand. Therefore, we clustered the TF-IDF vectors using the DBSCAN algorithm which does not require the number of clusters to be specified beforehand. The DBSCAN algorithm performs well with arbitrary shaped clusters and is robust to outliers. The vectorisation of the behaviour profiles, the computation of the cosine similarity matrix, and the DBSCAN clustering were performed using the Python implementation of TF-IDF, cosine similarity, and DBSCAN, provided by the scikit-learn machine learning framework [35].

5. Findings and Discussion

This section describes the evaluation of the proposed approach and presents the clustering results along with the key findings.

5.1. Dataset

The dataset consists of 1311 samples captured from active IoT botnets on the Internet. As shown in Table 3, the samples are compiled for the x86, MIPS and ARM CPU architectures. Approximately 60% of the obtained samples are packed, 9.5% of which are packed with a custom packer, as can be seen in Table 4. The samples were successfully executed and analysed in the sandbox, and a behaviour profile was created for each sample. The effectiveness of the sandbox execution and analysis was validated by investigating the created behaviour profiles. The number of recorded actions per behaviour profile vary from 10 to 504, indicating that the behavioural analysis of the samples was successful.

Table 3. CPU architecture per number of IoT botnet samples.

CPU Architecture	Number of Samples
X86	302
MIPS	510
ARM	499

Table 4. Packer used per number of IoT botnet samples.

Packer Used	Number of Samples
none	536
default UPX packer	700
custom packer	75

5.2. DBSCAN Parameter Tuning

The DBSCAN clustering algorithm takes two parameters, the maximum distance between two samples for them to be considered neighbours, referred to as epsilon (eps), and the minimum number of samples required to define a cluster. The Python implementation of DBSCAN, provided by the scikit-learn framework, assigns the samples cluster labels in the range from 0 to $N - 1$, where N is the number of identified clusters. The outliers are labelled as -1.

We evaluated the effectiveness of the DBSCAN clustering for identifying botnet variants equipped with new capabilities using a fixed value for the minimum number of samples per cluster, and different values for the eps parameter. We chose two as the minimum number of samples for defining a cluster because it may be possible that only two, or even only one sample of an IoT botnet variant has been captured by the honeypots or the security researchers. By selecting two as the minimum number of samples per cluster, the DBSCAN algorithm will be able to identify IoT botnet variants represented by only two samples in our dataset. The variants represented by only one sample would be labelled as outliers. The eps parameter was varied in the range between 0.2 and 0.9 at 0.1 intervals. The number of clusters and outliers for each eps value used in the evaluation is shown in Table 5. As can be seen from Figure 3, varying the eps parameter had significant effect on the number of clusters and outliers identified by DBSCAN. With the smallest eps value in the range, 0.2, DBSCAN identified 186 clusters and 226 outliers, while with the greatest eps value in the range, 0.9, the algorithm identified only two clusters and four outliers. As part of the evaluation, we investigated the effect the different eps values had on the degree of difference between the behaviour profiles of samples belonging to different clusters and within the same cluster. In doing so, we manually read the behaviour profiles and noted the observed differences.

The clustering with eps values smaller than 0.6 resulted in larger number of clusters and very fine-grained differences among the behaviour profiles assigned to different clusters. The fine-grained differences include different domain names, different C2 IP addresses and/or ports, different names of the executed files, different IP addresses of MD servers, different commands for disabling the firewall, etc. We also observed different

combinations of ports scanned or blocked/opened in the firewall, from the set of ports associated with widely exploited IoT vulnerabilities. These minor differences may be due to different botnet configuration parameters or slight changes in the botnet source-code. Because of this, the clustering based on such differences may not be beneficial for identifying botnet variants equipped with new capabilities. A significant difference that can be an indicator of a new capability could be, for instance, a unique set of scanned ports, including ports which are not associated with known infection vectors. On the other hand, the clustering with eps values greater than 0.6 resulted in significantly different behaviour profiles being assigned to the same cluster. In other words, samples presumably belonging to different IoT botnet variants were assigned to the same cluster.

Therefore, we identified the clustering with eps = 0.6 as the most effective for identifying botnet variants equipped with new capabilities. The behaviour profiles within each cluster described the same behavioural patterns and capabilities, with minor differences which do not indicate that the cluster comprises more than one botnet variant. The distinct behavioural characteristics of the clusters are discussed in the following subsection.

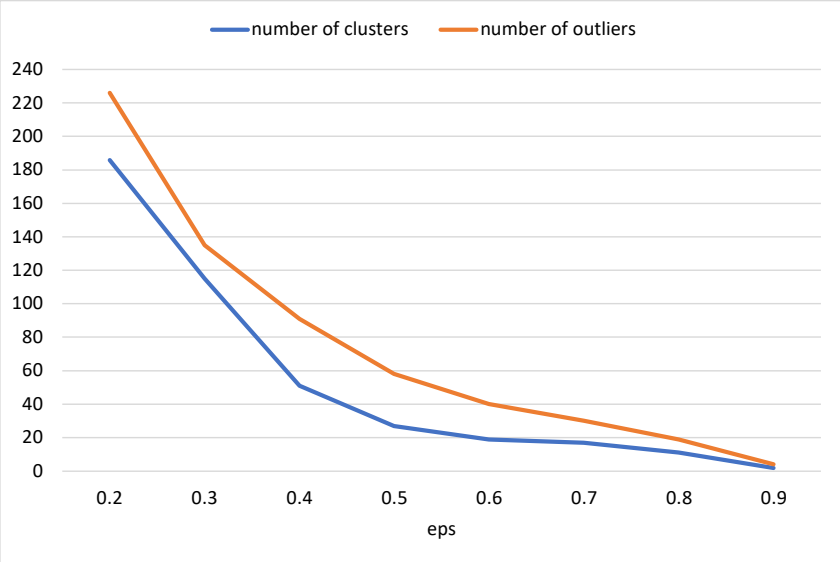

Figure 3. Effect of varying the eps parameter on the number of clusters and outliers.

Table 5. Number of clusters and outliers for different eps values.

Min. Number of Samples	eps	Number of Clusters	Number of Outliers
2	0.2	186	226
2	0.3	115	135
2	0.4	51	91
2	0.5	27	58
2	0.6	19	40
2	0.7	17	30
2	0.8	11	19
2	0.9	2	4

5.3. Results

The identified clusters, their unique behavioural characteristics, and the number of samples per cluster are shown in Table 6. Most of the samples, or 90%, were assigned to cluster 0. These samples exhibited behaviour very similar to the documented analyses of the leaked botnet and exploit source-codes [6]. We refer to such behaviour as generic IoT botnet behaviour. The generic IoT botnet behaviour typically includes the following actions: scanning for open ports associated with vulnerabilities commonly exploited by IoT botnets; exploiting the discovered vulnerable services using public exploits included in the

leaked IoT botnet source-codes; querying domain(s); connecting to C2 server(s); removing bash history and logs; disabling the firewall or blocking specific port(s); preventing watchdog from rebooting the device; etc. The dominant number of samples with generic IoT botnet behaviour supports the common belief that most of the IoT botnets are operated by cybercriminals with basic technical skills, who typically apply minor changes to the leaked IoT botnet source-codes [36].

We also inspected the behaviour profiles identified as outliers and observed that the actions described by each outlier behaviour profile differed significantly from those of the clusters and the other outliers. We assume that the behaviour profiles identified as outliers represent botnet variants, equipped with new capabilities, which are represented by a single sample in our dataset. An excerpt from an outlier behaviour profile is shown in Listing 3. The distinct behavioural characteristics of this outlier are a persistence technique involving a unique shell-script code which has not been observed in other variants, and the use of multiple configuration files downloaded as a rar archive during the initial infection stage.

Listing 3. Excerpt from an outlier behaviour profile.

```
cnc tcp 23.253.46.64:80
wrote to /etc/init.d/rorqefvqgc
wrote to /etc/cron.hourly/gcc.sh
wrote to /lib/libudev.so
dns question ww.search2c.com
http request /config.rar
dns question aa.hostasa.org
wrote to /usr/bin/rorqefvqgc
executed /usr/bin/rorqefvqgc
```

As can be seen from Table 6, the identified clusters exhibited new infection capabilities, and used unique techniques for establishing persistence, evading detection, and preventing infection remedy. The cluster 1 samples scanned a unique set of ports which have not been seen in other botnets, indicating a potential exploitation of unknown or less known vulnerabilities. The samples comprising cluster 9 exploited multiple vulnerable web services which are not typically exploited by IoT botnets, indicating that the variant may be infecting both IoT devices and Linux servers. The clusters 14, 16, and 18 used unique techniques for establishing persistence on an infected device. The cluster 2 samples sent a unique HTTP request, presumably to authenticate with an MD server. The authentication with an MD server may be a new capability aimed to prevent the discovery of the MD server and the botnet samples. The clusters 4,5, and 8 used unique HTTP requests to download stage-two payloads, presumably to avoid detection. The samples comprising clusters 12 and 18 used techniques for evading detection and preventing infection remedy which have not been observed in other variants. The cluster 12 samples executed a command which removed the traces of the stage-two payload execution, replaced the ls tool used for listing directory contents on Linux, and modified the user accounts to prevent remote access to the device. The cluster 18 samples replaced the ps, lsof, and netstat tools, used for getting information about running processes and open connections, as well as the sshd program which enables remote access to the device via SSH. The unique behavioural characteristics of the clusters can also indicate some of their monetization capabilities, such in the case of the clusters 14 and 17 which downloaded Python libraries used in cryptocurrency mining.

5.4. Comparison with Mean-Shift Clustering

We also evaluated the effectiveness of the Mean-shift clustering algorithm for identifying botnet variants with new capabilities from our sample collection. Mean-shift is a centroid-based algorithm which does not require prior knowledge of the number of clusters. In our evaluation, we used the Python implementation of the algorithm provided by the scikit-learn framework [37]. We configured the algorithm to automatically estimate the bandwidth used in the RBF kernel using the 'estimate_bandwidth' function. Because the Mean-shift algorithm requires the input data to be dense, we converted the sparse matrix

produced by the TfidfVectorizer to a dense one. The Mean-shift algorithm identified 66 clusters, 48 of which contain only one sample.

Table 6. Unique behavioural characteristics and number of samples per cluster.

Cluster	Unique Behavioural Characteristics	No. of Samples
0	generic IoT botnet behaviour	1180
1	scanned a unique combination of ports not seen in other botnets	4
2	unique HTTP request not associated with exploits; presumably a technique for authenticating with an MD server	2
3	created a unique set of directories and copied the botnet binary to the directories	2
4	downloaded and executed a stage-two payload using a unique HTTP request	3
5	downloaded and executed a stage-two payload using a unique HTTP request	2
6	downloaded multiple botnet samples compiled for different CPU architectures; loaded a kernel module	7
7	replaced the rm,tftp, and kill Linux programs	2
8	downloaded and executed a stage-two payload using a unique HTTP request; replaced /dev/tty	37
9	exploited multiple vulnerable web services which are not typically exploited by IoT botnets	3
10	created and wrote to files using encoded filenames, presumably to hinder analysis	2
11	obtained a list of malicious URLs, presumably used for botnet propagation	4
12	modified and added user accounts; downloaded and executed a stage-two payload using a unique command aimed to remove infection traces; replaced the ls tool	2
13	performed multiple DNS queries for google domains	3
14	downloaded multiple files; created a service for crypto-mining purposes and used several unique persistence techniques	2
15	installed UPnP plug and play service; modified /etc/hosts	2
16	used unique persistence techniques in which the botnet sample is run with arguments	2
17	downloaded a unique set of multiple different Python libraries, typically used for crypto-mining, and stored them in a uniquely named directory	3
18	used unique persistence techniques with multiple runlevels; loaded kernel modules; replaced the ps, lsof, sshd and netstat programs	9
outliers	each outlier exhibited unique behavioural characteristics	40

To evaluate the effectiveness of Mean-shift, we compared the clusters it identified with the clusters found by DBSCAN. We refer to the clusters identified by Mean-shift as Mean-shift clusters and to the clusters identified by DBSCAN as DBSCAN clusters. To ease the comparison, we consider the clusters identified by Mean-shift which contain only one sample as outliers. The number and size of the clusters identified by Mean-shift are similar to those of the clusters identified by DBSCAN, as can be seen in Tables 6 and 7, respectively. Like DBSCAN, the Mean-shift algorithm also identified one large cluster, i.e.,

cluster 0, and several small clusters. Furthermore, Mean-shift identified 48 outliers, labelled as cluster −1 in Table 7, while DBSCAN identified 40 outliers. The clusters identified as cluster 0 by both algorithms are comprised of samples which exhibited generic IoT botnet behaviour. Fifteen of the 17 small Mean-shift clusters labelled as clusters 1–17 in Table 7, were also identified by DBSCAN. These 15 small clusters represent IoT botnet variants equipped with new capabilities. The samples comprising them exhibited unique behavioural characteristics per cluster. The Mean-shift clusters 3 and 12 were not identified by DBSCAN. Instead, the samples comprising these clusters were assigned to the DBSCAN cluster 0. We examined the behaviour profiles of these samples, and found that they describe generic IoT botnet actions, such as establishing persistence using cron, removing bash history and disabling firewall. Therefore, we believe that these samples should have been assigned to the Mean-shift cluster 0, since they exhibited generic IoT botnet behaviour.

On the other hand, the DBSCAN clusters 8, 9 and 11, shown in Table 6, were not identified by Mean-shift. Despite sharing the same behavioural patterns and capabilities, each of the three samples comprising the DBSCAN cluster 9 was classified as an outlier by Mean-shift. We consider this classification inaccurate because an outlier is expected to represent an IoT botnet variant with distinct behavioural characteristics. The samples comprising the DBSCAN clusters 8 and 11 were assigned to the Mean-shift cluster 0, although they exhibited unique behavioural characteristics per cluster, as shown in Table 6. In addition, Mean-shift classified as outliers six samples that exhibited generic IoT botnet behaviour and should have been assigned to the Mean-shift cluster 0. These observations show that despite identifying many of the IoT botnet variants equipped with new capabilities, the Mean-shift algorithm is less effective compared to the DBSCAN algorithm.

Table 7. Clusters identified by the Mean-shift algorithm.

Cluster	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	−1
No. of Samples	1193	9	7	11	3	3	4	3	2	2	2	2	12	2	2	2	2	2	48

5.5. Comparison with Hierarchical Clustering

In addition to the Mean-shift clustering, we also evaluated the effectiveness of the Agglomerative Hierarchical clustering for identifying IoT botnet variants with new capabilities from our dataset. We decided to evaluate the Agglomerative Hierarchical clustering since it is typically better at identifying small clusters than the Divisive Hierarchical clustering. We used the Python implementation of the Agglomerative clustering algorithm provided by scikit-learn which requires the linkage criterion and the number of clusters to be specified [38]. Using the cosine similarity matrix as input, we investigated the clusters identified with each of the following linkage criterions: 'single', 'average', and 'complete'. The number of clusters was set to 59, which is the number of clusters and outliers identified by DBSCAN, and is close to the number of clusters and outliers identified by Mean-shift, 65.

Using the 'complete' and 'average' linkages, the Agglomerative algorithm successfully recognised most of the IoT botnet variants equipped with new capabilities. However, it also identified multiple small clusters comprised of samples which exhibited generic IoT botnet behaviour. The differences between the behaviours of these small clusters are minor, and do not indicate that the clusters represent IoT botnet variants equipped with new capabilities. This limitation makes the Agglomerative clustering using the 'complete' and 'average' linkages less effective compared to the DBSCAN clustering.

Using the 'single' linkage, the Agglomerative algorithm identified the same clusters as the DBSCAN algorithm. The identified clusters, shown in Table 6, include one large cluster comprised of samples with generic IoT botnet behaviour, along with multiple small clusters and outliers representing IoT botnet variants equipped with new capabilities. Therefore, using the 'single' linkage, the Agglomerative clustering is equally effective for identifying IoT botnet variants equipped with new capabilities as the DBSCAN clustering.

5.6. Comparison with AVClass Classification

Some of the related studies have evaluated the clustering effectiveness using the labels assigned to malware samples by different antivirus engines. A clustering is considered as effective if all samples within a single cluster are assigned the same label. An antivirus label typically indicates a malware class the sample belongs to. However, a malware sample may not be detected by all antivirus engines. Furthermore, a malware sample can be assigned different labels by different antivirus engines. In the case of the latter, it is necessary to identify the most likely malware class the sample belongs to. This can be achieved using AVClass.

AVClass [39] is a tool which uses the labels assigned from different antivirus engines to identify the most likely malware class the sample belongs to. It takes as an input a JSON report containing the results from the scans performed by the online antivirus scanner VirusTotal. VirusTotal [40] is an online service for scanning files using more than 60 different antiviruses simultaneously. The VirusTotal report contains the labels assigned to the malware sample by the different antivirus vendors. One limitation of AVClass is that it may unable to identify the most likely malware class if there is a lack of decisive antivirus labels.

We used the API provided by VirusTotal to scan each sample from our dataset and to obtain the scan results. After scanning the samples, we used AVClass to identify the most likely malware class of each sample. We then investigated the malware classes of the samples per cluster and identified two cases: (a) when all samples within the cluster were assigned the same class, we refer to such clusters as pure and (b) when the samples within a cluster were assigned different classes, we refer to such clusters as mixed.

Sixteen of the nineteen clusters identified by DBSCAN, shown in Table 6, are pure, seven of which were assigned the class 'linux', six the class 'mirai', and one cluster was assigned the class 'python'. However, the classes assigned by AVClass may not be helpful for identifying botnet variants. For instance, the distinction between 'linux' and 'mirai' classes is unclear since Mirai is a type of Linux malware. We investigated the behaviour profiles of the samples comprising one of the 'mixed' clusters and observed samples with identical behaviour profiles which have been assigned different malware classes. This may be a result of malware classification performed by some antivirus vendors using only static features. Therefore, we believe that the clustering evaluation based on labels assigned by antivirus vendors may not be effective in the case of IoT malware.

6. Conclusions

In this paper, we proposed a novel approach for automated behaviour-based clustering of IoT botnets. The proposed approach enables automatic identification of IoT botnet variants equipped with new capabilities, and thus overcomes the need to manually investigate the IoT botnet samples for new botnet variants to be identified. The paper also presents an approach for capturing the behaviour of the botnet samples, the challenges that may affect it, and the actions applied to overcome the challenges. The captured behaviours of the samples are first profiled as sets of actions executed by the samples, then vectorised using TF-IDF, and finally clustered with the DBSCAN algorithm.

The effectiveness of the proposed clustering approach was validated using a collection of botnet samples captured from IoT botnets propagating on the Internet. The DBSCAN algorithm successfully identified multiple IoT botnet variants which exhibited new infection capabilities, and used unique techniques for establishing persistence, evading detection, and preventing infection remedy. We also evaluated the effectiveness of the Mean-shift and the Agglomerative Hierarchical clustering algorithms, and found that Mean-shift is less effective than DBSCAN, while the Agglomerative clustering using 'single' linkage is equally effective as DBSCAN. Furthermore, we investigated the malware classification of the collected samples performed by different antivirus vendors, and found that it may not help the identification of IoT botnet variants.

The benefits of the automatic identification of IoT botnet variants equipped with new capabilities are twofold. First, it enables malware analysts to keep apace with the evolution of IoT botnets. Second, it allows security researchers to investigate the new capabilities, and to apply the investigation findings for improving the solutions for detecting and preventing IoT botnet infections. In addition, the clustering results show that approximately 90% of the samples collected in this study exhibited generic IoT botnet behaviour associated with the leaked IoT botnet source-codes, which may be an indicator of the effect the leaking of botnet source-codes has on the reported increase in the number of IoT botnet samples collected daily by antivirus vendors. In the future, we plan to expand the proposed approach by adding support for Android malware.

Author Contributions: Conceptualization, N.Z.; data curation, T.T.; formal analysis, T.T.; investigation, T.T. and N.Z.; software, T.T.; methodology, T.T. and N.Z.; supervision, N.Z.; validation, T.T.; visualization, T.T. and N.Z.; writing—original draft, T.T.; writing—review & editing, N.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: We have made the collection of behaviour profiles of the IoT botnet samples used in this study available at https://drive.google.com/file/d/1VgYmrnLX0VSBoErUJk7txJuR3MfeRBdV/view?usp=sharing (accessed on 18 December 2021).

Acknowledgments: We gratefully acknowledge the financial support by the University of Manchester in this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, Y.; Wang, H. Tracking Mirai variants. *Virus Bull.* **2018**, *10*, 1–18.
2. Paquet-Clouston, M.; Bilodeau, O.; GoSecure Inc. Attacking Linux/Moose 2.0 Unraveled an Ego Market. Available online: https://www.botconf.eu/wp-content/uploads/2016/11/PR08-MOOSE-BILODEAU-PAQUET-CLOUSTON.pdf (accessed on 19 December 2021).
3. Elovici, Y.; Shabtai, A.; Breitenbacher, D.; Bohadana, M.; Mathov, Y.; Meidan, Y.; Mirsky, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
4. Secplicity. IoT Botnets Are Evolving—How Big Can They Get? Available online: https://www.secplicity.org/2018/02/20/iot-botnets-evolving-big-can-get/ (accessed on 19 December 2021).
5. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and Other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
6. Antonakakis, M.; April, T.; Bailey, M.; Bernhard, M.; Arbor, A.; Bursztein, E.; Cochran, J.; Durumeric, Z.; Halderman, J.A.; Arbor, A.; et, al Understanding the Mirai Botnet. *Usenix Secur.* **2017**, *317*, 54–61. [CrossRef]
7. McAfee. McAffee Labs Threat Report 06.21. Available online: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-threats-jun-2021.pdf (accessed on 19 December 2021).
8. Trajanovski, T.; Zhang, N. An Automated and Comprehensive Framework for IoT Botnet Detection and Analysis (IoT-BDA). *IEEE Access* **2021**, *9*, 124360–124383. [CrossRef]
9. Cozzi, E.; Graziano, M.; Fratantonio, Y.; Balzarotti, D. Understanding Linux Malware. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), Francisco, CA, USA, 20–24 May 2018; pp. 161–175. [CrossRef]
10. Le, H.V.; Ngo, Q.D. V-Sandbox for Dynamic Analysis IoT Botnet. *IEEE Access* **2020**, *8*, 145768–145786. [CrossRef]
11. Dib, M.; Torabi, S.; Bou-Harb, E.; Assi, C. A Multi-Dimensional Deep Learning Framework for IoT Malware Classification and Family Attribution. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 1165–1177. [CrossRef]
12. Kawasoe, R.; Han, C.; Isawa, R.; Takahashi, T.; Takeuchi, J. Investigating behavioral differences between IoT malware via function call sequence graphs. In Proceedings of the ACM Symposium on Applied Computing, Virtual Event, 22–26 March 2021; pp. 1674–1682. [CrossRef]
13. Torabi, S.; Dib, M.; Bou-Harb, E.; Assi, C.; Debbabi, M. A Strings-Based Similarity Analysis Approach for Characterizing IoT Malware and Inferring Their Underlying Relationships. *IEEE Netw. Lett.* **2021**, *3*, 161–165. [CrossRef]
14. He, T.; Han, C.; Isawa, R.; Takahashi, T.; Kijima, S.; Takeuchi, J.; Nakao, K. A Fast Algorithm for Constructing Phylogenetic Trees with Application to IoT Malware Clustering. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer International Publishing: Cham, Switzerland, 2019; Volume 11953, pp. 766–778. [CrossRef]
15. Bak, M.; Papp, D.; Tamas, C.; Buttyan, L. Clustering IoT Malware based on Binary Similarity. In Proceedings of the IEEE/IFIP Network Operations and Management Symposium 2020: Management in the Age of Softwarization and Artificial Intelligence, NOMS 2020, Budapest, Hungary, 20–24 April 2020. [CrossRef]

16. Carrillo-Mondéjar, J.; Martínez, J.; Suarez-Tangil, G. Characterizing Linux-based malware: Findings and recent trends. *Future Gener. Comput. Syst.* **2020**, *110*, 267–281. [CrossRef]
17. Qaiser, S.; Ali, R. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *Int. J. Comput. Appl.* **2018**, *181*, 25–29. [CrossRef]
18. Schubert, E.; Sander, J.; Ester, M.; Kriegel, H.P.; Xu, X. DBSCAN Revisited. *ACM Trans. Database Syst.* **2017**, *42*, 1–21. [CrossRef]
19. Margolis, J.; Oh, T.T.; Jadhav, S.; Kim, Y.H.; Kim, J.N. An In-Depth Analysis of the Mirai Botnet. In Proceedings of the 2017 International Conference on Software Security and Assurance (ICSSA), Altoona, PA, USA, 24–25 July 2017; pp. 6–12. [CrossRef]
20. Unit42-Palo Alto Networks. Muhstik Botnet Attacks Tomato Routers to Harvest New IoT Devices. Available online: https://unit42.paloaltonetworks.com/muhstik-botnet-attacks-tomato-routers-to-harvest-new-iot-devices/ (accessed on 19 December 2021).
21. Sicato, J.C.S.; Sharma, P.K.; Loia, V.; Park, J.H. Vpnfilter malware analysis on cyber threat in smart home network. *Appl. Sci.* **2019**, *9*, 2763. [CrossRef]
22. Skuratovich, S. Defeating Sandbox Evasion: How to Increase Successful Emulation Rate in Your Virtualized Environment. *Virus Bull.* **2018**, 1–5. Available online: https://blog.checkpoint.com/wp-content/uploads/2016/10/DefeatingSandBoxEvasion-VB2016_CheckPoint.pdf (accessed on 19 December 2021).
23. Abuse.ch. IoT Botnets Takedown Statistics. Available online: https://urlhaus.abuse.ch/statistics/#avg_takedown (accessed on 19 December 2021).
24. Hoang, D.K.; Tho Nguyen, D.; Vu, D.L. IoT Malware Classification Based on System Calls. In Proceedings of the 2020 RIVF International Conference on Computing and Communication Technologies, RIVF 2020, Ho Chi Minh, Vietnam, 6–7 April 2020; [CrossRef]
25. Cozzi, E.; Vervier, P.A.; Dell'Amico, M.; Shen, Y.; Bilge, L.; Balzarotti, D. The Tangled Genealogy of IoT Malware. In *Annual Computer Security Applications Conference*; ACM: New York, NY, USA, 2020; pp. 1–16. [CrossRef]
26. Lingenfelter, B.; Vakilinia, I.; Sengupta, S. Analyzing Variation among IoT Botnets Using Medium Interaction Honeypots. In Proceedings of the 2020 10th Annual Computing and Communication Workshop and Conference, CCWC 2020, Las Vegas, NV, USA, 6–8 January 2020; pp. 761–767. [CrossRef]
27. Bayer, Ulrich and Comparetti, Paolo and Hlauschek, Clemens and Krügel, Christopher and Kirda, E. Scalable, Behavior-Based Malware Clustering. *NDSS* **2009**, *9*, 8–11.
28. URLhaus. Top Malware Hosting Networks. Available online: https://urlhaus.abuse.ch/statistics/ (accessed on 19 December 2021).
29. SystemTap. Available online: https://sourceware.org/systemtap/ (accessed on 19 December 2021).
30. Strace. Available online: https://man7.org/linux/man-pages/man1/strace.1.html (accessed on 19 December 2021).
31. TCPDump. Available online: https://www.tcpdump.org/manpages/tcpdump.1.html (accessed on 19 December 2021).
32. Church, K.W. Emerging Trends: Word2Vec. *Nat. Lang. Eng.* **2017**, *23*, 155–162. [CrossRef]
33. Thijs, B. Using neural-network based paragraph embeddings for the calculation of within and between document similarities. *Scientometrics* **2020**, *125*, 835–849. [CrossRef]
34. De Boom, C.; Van Canneyt, S.; Bohez, S.; Demeester, T.; Dhoedt, B. Learning Semantic Similarity for Very Short Texts. In Proceedings of the 15th IEEE International Conference on Data Mining Workshop, ICDMW 2015, Atlantic, NJ, USA, 14–17 November 2015; pp. 1229–1234. [CrossRef]
35. Hao, J.; Ho, T.K. Machine Learning Made Easy: A Review of Scikit-learn Package in Python Programming Language. *J. Educ. Behav. Stat.* **2019**, *44*, 348–361. [CrossRef]
36. Angrishi, K. Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV): IoT Botnets. *arXiv* **2017**, arXiv:1702.03681.
37. Scikit-Learn. Mean-Shift Clustering Algorithm. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html (accessed on 19 December 2021).
38. Scikit-Learn. Agglomerative Clustering Algorithm. Available online: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html (accessed on 19 December 2021).
39. Sebastián, M.; Rivera, R.; Kotzias, P.; Caballero, J. AVclass: A Tool for Massive Malware Labeling. *Research in Attacks, Intrusions, and Defenses*; Monrose, F., Dacier, M., Blanc, G., Garcia-Alfaro, J., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 230–253.
40. Virustotal. Available online: https://virustotal.com (accessed on 19 December 2021).