I. J. Computer Network and Information Security, **2024, 5, 98-112**
Published Online on October 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2024.05.08

IHBOT: An Intelligent and Hybrid Model for Investigation and Classification of IoT Botnet

Umang Garg*
Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, India and Department of Computer Science and Engineering, Graphic Era Hill University, Dehradun, India
E-mail: umangarg@gmail.com
ORCID iD: https://orcid.org/0000-0002-1815-5794
*Corresponding Author

Santosh Kumar
Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, India
E-mail: amu.santosh@gmail.com
ORCID iD: https://orcid.org/0000-0002-1008-0804

Manoj Kumar
Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun, India
E-mail: manoj.kumar.nitj@gmail.com
ORCID iD: https://orcid.org/0000-0002-2252-9605

Received: 20 February 2023; Revised: 19 April 2023; Accepted: 08 June 2023; Published: 08 October 2024

Abstract: The Internet of Things (IoT) is revolutionizing the technological market with exponential growth year wise. This revolution of IoT applications has also brought hackers and malware to gain remote access to IoT devices. The security of IoT systems has become more critical for consumers and businesses because of their inherent heterogenous design and open interfaces. Since the release of Mirai in 2016, IoT malware has gained an exponential growth rate. As IoT system and their infrastructure have become critical resources that triggers IoT malware injected by various shareholders in different settings. The enormous applications cause flooding of insecure packets and commands that fueled threats for IoT applications. IoT botnet is one of the most critical malwares that keeps evolving with the network traffic and may harm the privacy of IoT devices. In this work, we presented several sets of malware analysis mechanisms to understand the behavior of IoT malware. We devise an intelligent and hybrid model (IHBOT) that integrates the malware analysis and distinct machine learning algorithms for the identification and classification of the different IoT malware family based on network traffic. The clustering mechanism is also integrated with the proposed model for the identification of malware families based on similarity index. We have also applied YARA rules for the mitigation of IoT botnet traffic.

Index Terms: IoT Botnet, Malware Classification, Machine Learning, Malware Analysis.

1. Introduction

The most significant digital huge trend, bridging the virtual and physical worlds is the Internet of Things (IoT). People, machines, the Internet, and objects are becoming increasingly connected, resulting in the birth of new strategies and innovative engagements between humans and the world at large. IoT systems have become an alluring target for hackers who begin taking advantage of vulnerabilities authentication, antiquated firmware, as well as malware to jeopardize IoT systems owing to the difficulty of planning and evaluation in both equipment and software, and the lack of security features and competencies. IoT devices will be expected to increased up to 41.6 billion annually by 2025 [1]. With the increased deployment of IoT technology in business, these assaults will continue to proliferate. Botnet is one of the most significant risks for IoT devices. The Mirai botnet family launched one of several biggest and most influential DDoS assaults in recent years on Dyn (a prominent American DNS network service provider) in October 2016 [2]. About 1.2 million IoT sensors and devices were affected, and the bot targeted numerous famous internet services including Google, Facebook, Amazon, and others. As a result, strengthening the protection of IoT devices and sensors is becoming increasingly important for researchers, particularly when contending with IoT botnets. IoT botnet

includes several features, including the ability to undertake DDoS attacks and to monitor the open ports of IoT applications such as File Transfer Protocol, Telnet, or SSH. It also can employ a brute force attack to obtain contact to IoT equipment. According to Vinayakumar et al. [3] the majority of modern botnet is built by duplicating source code from internet instructions or perhaps a variation of the very same dangerous code produced by that the malware developer. There are several IoT botnets developed and implemented in recent time for the hacking of IoT devices and applications.

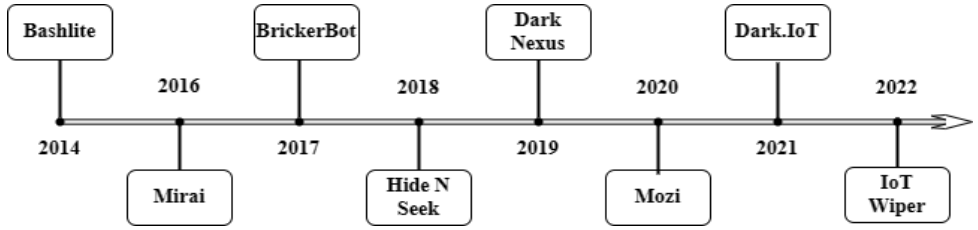

Fig.1. IoT botnet variants year wise evolution

IoT botnet is evolved from the traditional network botnet and grows year-by year. It is targeting several IoT applications such as healthcare, transportation, remote surgeries, etc. There are distinct IoT botnet variants came is existence in the recent time that targets new victims using a variety of designs. Moving forward since 2016 (Figure 1), distinct variants of Mirai has been launched after the release of source code. Therefore, the investigation and detection of IoT botnets becomes a prime concern. This can be majorly done by using static and dynamic analysis techniques. Static analysis is used to analyze the malware without executing the malware. It is used to evaluate the information regarding the functionality, signature, file size, file type, and structure of the malware. The static analysis will transform the binaries of the botnet into a unique structured binary and self-compressed file. Static features typically employed in the research include control flow graphs (CFG), operational codes (opcode), phrases, file headers, gray-scale images, and so on [4]. The efficacy and complexities of IoT malware identification algorithms are strongly influenced by the way these characteristics were retrieved and analyzed. Dynamic analysis-based susceptibility identification, apart from static analysis, necessitates the execution of IoT system software on the genuine software or hardware emulation. The intended firmware should perform based on accurate inputs to the method and encompass practically all conceivable programmed pathways to be efficient. It is a technique to analyze and identify the vulnerabilities of IoT botnets in multiple network protocols. Compared to static analysis, the dynamic analysis offers distinguished benefits. It does not necessitate disassembling the executable files. Since it examines the sample by running the malware therefore it reveals the natural malware behavior that is more resistant to static analysis, and is more potent against the malware. Dynamic analysis is preferable in studying malware due to its efficiency against it. A novel hybrid malware analytics system was devised and deployed in this work to analyze IoT malwares and validates the results. It also utilizes the vast atmosphere containing a sandbox for malware analysis software. The malware analysis technique suggested in this work, is an autonomous analysis technique that may combine approaches of static and dynamic analysis.

Besides, we found a wide gap and significant improvement required that motivate us to investigate, and analyze the IoT botnet samples collected from the IoT network traffic using an integrated hybrid model, and machine learning approaches. The major contribution of the present work is as follows:

- To implement and investigate the IoT botnet malware using a hybrid approach.
- To classify the IoT botnet using a distinct set of machine learning approaches.
- To investigate the behavior of IoT botnet and its characteristics for a better understanding of the IoT botnet malware i.e., the classification of a malware family.
- To build the clusters based on the similarity index for detection and identification of malware family.

The rest of the paper is categorized as follows: section II explored the existing work by several researchers for the analysis of IoT botnets. Section III provides the background details and challenges faced during the IoT botnet analysis. Section IV describes the proposed system architecture and testbed design of the system. Section V shows the experimental results in terms of tables and charts. Section VI concludes this research with the future scope of the work.

2. Literature Review

There are several research papers on IoT device protection. Granjal et al. [5] like many others, concentrated on studying existing protocols and procedures to secure IoT connections. Djamel Eddine Kouicem et al. [6] gave a top-down overview among the most often recommended confidentiality solutions for IoT technology and classified numerous IoT systems to determine security procedures and obstacles, they have also examined classic encryption techniques to cope with secrecy, confidentiality, and accessibility. Authors have classified the IoT botnet by using machine learning classifiers and obtained the results in terms of accuracy.

Costin et al. [7] merely provided a complete review and evaluation of all presently recognized IoT malware types, but they did not explore IoT botnet identification methods. Premised on the kind of method, IoT malware identification approaches may be divided into two categories: a static or dynamic analysis. The dynamic method [8] entails continuously monitoring executable code and identifying anomalous activities. Monitoring operational processes requires a lot of resources, and malware might infect real-world situations in some circumstances. Furthermore, because many varieties of malware demand trigger circumstances to undertake dangerous behaviors, it is impossible to thoroughly monitor the whole of their actions throughout the operation. In contrast to the restrictions of dynamical analysis, the implementation of IoT application programs confronts several challenges, including various architectures (e.g., Malware Information Sharing Platform, Advanced RISC Machines, PowerPC, Sparc) & IoT equipment resource restrictions. As a result, configuring an infrastructure that satisfies the prerequisites for IoT application programs to work effectively and completely is tough. The static technique [9], on the other hand, involves examining and identifying harmful files without running them. The capability to see the architecture of malware has been one of the key benefits of unit testing. In alternative words, researchers can investigate all conceivable execution routes in a malware strain without considering the variety of processor structures, making this technique helpful in addressing the heterogeneous difficulties that IoT devices face. First of all, several researchers [9][10] tried to capture the IoT botnet samples from 2the bot family. Once the samples have been captured for the IoT botnet family, the samples are analyzed by using static or dynamic analysis (shown in Table 1).

Table 1. Analysis review

Authors	Features	Category	Classifier	Acc.
Cheng [11]	Full Native API	B-72, M-18	SVM	94.4
Hashemi [12]	Opcode	B-300, M-300	KNN	91.9
Chia-Yi [13]	Function call graph	M-108k	SVM	98.8
Habibi [14]	CGAN	M-100k, B-95k	MLP	98.9
Galal [15]	API Sequences	M-2K, B-2K	RF, SVM	97.2
Abbasi [16]	PSO-based Wrapper	Ransomware	SVM, RF, DT	97.3
Nguyen [17]	Graph-based	M-5.5k, B-2.7k	SVM, DT, RF	98.1
Giang [18]	System Calls	M-5k, B-3.8K	SVM, DT	99.3

3. Background and Challenges

This section describes the details of the IoT botnet operation, analysis techniques and challenges faced during the analysis, and investigation of the IoT botnet.

3.1. Sequential Flow Diagram for IoT-botnet

An IoT botnet is a compromised network of interconnected devices that are controlled by the cyber criminals to execute malicious activities [17,19]. It may follow centralized architecture where the IoT devices (bots) are orchestrated by a command and control (C&C) server or trying to compromise the network. While in the Peer-to-peer (P2P) architecture, bots are communicated ffwith each other and there are no requirements for any centralized server for control. The bots are instructed to perform a specific action such as crypto-mining, distributed denial of service (DDoS), ransomware, or identity theft, etc.

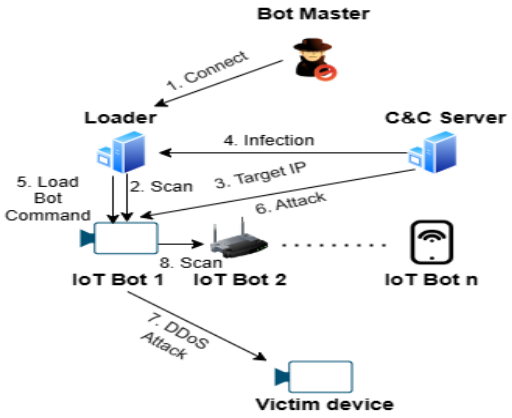

Fig.2. Sequential flow diagram for Iot botnet

The propagation of attacks is possible by using the Internet or local network if devices are connected through the centralized server. These attacks can be possibly done by exploiting the vulnerability of IoT devices which can be

achieved using default credentials or remote code execution. Default credentials are assigned to the devices at the time of manufacturing the devices. While remote code execution is achieved by hackers after the execution of remote code to vulnerable devices [20]. The operations of the IoT botnet can be explained in distinguished steps as depicted in Figure 2 among the following steps:

- First, the botmaster or attacker tries to connect with the loader with default credentials or remote code execution.
- Once, it got access to the loader server, it scans the IoT devices which may have some vulnerable open ports.
- In the next step, a list of targeted IPs sends to the C&C server and used for further propagation of infection.
- An infection command containing the information such as IP address, credentials, and vulnerable ports of the victim devices send to the loader.
- Once the loader downloads and execute the corresponding binaries, the infected devices become part of a botnet and can communicate with the C&C server.
- Further, the botmaster can issue specific types of attacks or commands through the C&C server with attack parameters and duration.
- Finally, distinguished attacks can be performed on the targeted victim devices and remove the binaries for runs in the memory to avoid detection.
- This procedure is now repeated for propagation of attacks to the new vulnerable devices.

3.2. IoT Botnet State Modelling

An IoT device that is referred to as a bot is a compromised device used to receive control commands and try to expand the bot network. Those commands may be propagated or discarded due to outdated or staled. The devices can move from one state to another depending on the degree of connectivity. This strategy will allow us to evaluate the degree of communication and spread boundaries of malware [21]. The set of possible degrees or the capabilities of communication i.e., n can be represented in (Figure 3):

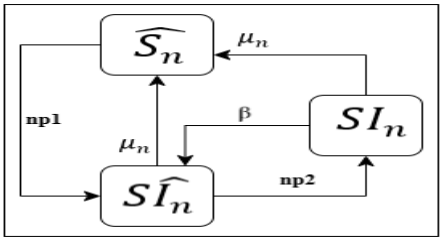

Fig.3. IoT botnet state modelling

- $\widehat{S_n}$ – Represent the uncompromised devices of degree n.
- $\widehat{SI_n}$- Represent the compromised devices with un-informed bot.
- SI_n- Represent the compromised devices with informed bots.

In IoT botnet modeling, the transition between the states can be organized into 3 different states such as uncompromised, compromised with un-informed, and compromised with informed. The transition from an uncompromised state to a compromised un-informed state can possible with degree n, probability p1, or p2. At any instance of time, the constant rate β is used to control commands that are discarded and maintain the recency of information. If the final state is un-compromised that indicates the patched bots. The patching rate indicates the degree-based non-uniform transmission (μ_n). Mathematically, the state's evolution can be represented as (Equations 1-7):

$$\frac{dSI_n(t)}{dt} = np_2\widehat{SI_n}(t) - (\beta + \mu_n)SI_n(t) \quad (1)$$

$$\frac{d\widehat{SI_n}(t)}{dt} = \beta\, SI_n(t) + np_1\widehat{S_n}(t) - (\mu_n + np_2)\widehat{SI_n}(t) \quad (2)$$

$$\frac{d\widehat{S_n}(t)}{dt} = \mu_n\left(1 - \widehat{S_n}(t)\right) - np_1\widehat{S_n}(t) \quad (3)$$

The probability p_1 and p_2 can be expressed in terms of $\sigma 1$ and $\sigma 2$.

$$p_1 = P\partial\gamma_b(1 - \sigma 1) \quad (4)$$

$$p_2 = P\gamma_c\sigma 2) \quad (5)$$

where $\sigma 1\ and\ \sigma 2$ are the probability for a specific link of un-compromised state and informed bot device respectively.

$$\sigma 1 = \sum_{n'} \frac{n' P(n')}{E(n)} \widehat{S_n}\,(\mathrm{t}) \tag{6}$$

$$\sigma 2 = \sum_{n'} \frac{n' P(n')}{E(n)} SI_n(\mathrm{t}) \tag{7}$$

3.3. IoT Botnet Analysis Techniques

In today's era, the discovery, identification, and classification of IoT botnet malware from multiple families of malware are extremely challenging. Although, traditional malware can be detected by using existing secured anti-virus and firewall systems. Yet, there is some modern malware that executes in the kernel and is very difficult to identify as compared to traditional malware. The main advantage of the investigation and analysis of IoT malware is to provide strength to the security firms and their strategies against malware attacks. There are 4 distinguished IoT botnet analysis techniques such as static, dynamic, hybrid, and memory-based.

A. Static Analysis

These techniques are used to analyze the malware file without running the portable executable (PE) files [13][22]. The static analysis extracts the necessary information such as URL redirection, malicious intent, linking libraries, and unusual access to system files. Firstly, the PE file needs to be unpacked by using analyzer tools such as EXEinfo and decompressed by using the IDA tool. After that information can be extracted in terms of string signatures, APfI calls, reverse engineering tools, opcodes, control flow graphs (CFG), and n-grams. String signatures are good indicators to reveal malware and semantic information. Windows API calls are used to reveal the information program behavior and are considered for malware detection. Reverse engineering tools are used to rebuild the code and able to detect the infection in the binary file. Opcode is a kind of machine code that identify the operations executed by the CPU. CFG is used to extract the structure of the malware and PE file behavior. N-grams are the sequence of n-subsequences that can be integrated with API calls or opcodes. Machine learning-based classifiers are also applied to existing techniques for the detection or classification of IoT malware.

B. Dynamic Analysis

Dynamic analysis is used to analyze the malware with the execution of suspicious files in a controlled environment. These techniques are used for the behavior analysis of unknown malware, monitoring of information logs, and changes in network activities [23]. For the analysis of malware, there are some techniques associated such as monitoring of system files, network activities, process activities, function calls, tracing of system calls, tracking of information flow, and control flow graph analysis. During the monitoring of system files, a malicious code may request remote code execution, commands, or some specific activities executed on the targeted system. These distinguished activities are recorded in a sequence of events and detected with tools like notify, and fswatcher. The detected events are classified as known (with known signatures) malware and unknown (zero-day) malware.

The monitoring of in-bound and out-bound IoT network traffic is very crucial during the execution of malware in a controlled environment with a tool named Moloch which provides information on malicious commands, and code propagation. For the monitoring of process activities such as inter-process communication, process information, etc., the ELF binaries need to be monitored in a controlled environment and extract the information's suspicious behavior. This technique also analyzes the ptrace system call which is having the information for space, heap, registers, and stack. The functional call is utilized for the abstract view of the code and targeted by the attackers with inject of a malicious piece of code. A hooking method is used to monitor the functional call with API calls. A deep analysis of the functional block is utilized to provide the intended behavior of the attacker.

For accessing the memory and network interface, attackers initiate some sort of system calls for specific operating systems. These system calls need to be traced with the return value, instruction pointer register, and arguments passed to it. The strace and etrace utility are analyzed for the tracing of system calls and kernel modules. If there is an involvement of very large binary files, multiple system calls, and shared library calls then a control flow graph analysis is very effective. This approach is based on the graph which provides detailed information about the parameters passed, return values, API calls, system calls, and functional flow.

C. Hybrid Analysis

It is an integration of static and dynamic analysis techniques that increased the chances of IoT malware detection [24]. This analysis is used to detect known and unknown malware using malware signatures, behavioral patterns, printable string information (PSI), function calls, and API calls. PSI-based features are extracted through static analysis and then apply dynamic analysis for the extraction of API calls to get better results. Jeon et al. [25] proposed a hybrid analysis detection framework named HyMaID based on spatial pyramid pooling network (SPPNET) and Bi-LSTM. It is used to extract the static features of the opcode and API calls dynamically after obfuscation. The accuracy achieved from the proposed framework is 92.5%. The model can be improved by applying anti-debugging techniques on unknown malware. Hamza et al. [26] build a hybrid security analyzer named HSAS-MD extracting the static features

and classifying them with the CNN model. Then verifying the malware behavior with model checking such as CFG, API calls, etc. The major utilization of the proposed model is analyzer ability, ability to detect abnormal behavior, and decision-making ability. The performance accuracy of the system is 95% with 94% precision.

D. Memory-based Analysis

The memory analysis of the IoT malware keeps track of the memory images and analyze distinguished factors such as running programs, states of processor storage, imported libraries, registered activities, and running platform [27]. There are two distinguished steps are involved in this analysis as memory acquisition and memory analysis phase. Tools like Memorize or DumpIt are used to dump the memory of the targeted device to obtain the memory images. While in the second phase, analyze the memory images with tools like Rekall or volatility. Some memory analysis techniques are also used to monitor the hidden processes, API hooking, and dynamic link library (DLL) injection. The hidden processes can be monitored by libraries, dynamic link structure, kernel manipulation, and tracking of the system calls. API hooking is utilized to interrupt the function calls, and flow of API calls, and modify the behavior of API calls. Some of the API hooking techniques are the imported address table (IAT), system service descriptor table (SSDT), and input/output request packet (IRP). DLL injections can be utilized to insert malicious code in an authentic process and transit the normal process into malicious memory space. Some examples of DLL injection include registry manipulation, load-library, and triggering of event-based calls.

3.4. Issues and Challenges

Although, the significant number of research focused on the analysis of IoT malware with different techniques. However, there are some muffled areas or key issues faced during the analysis and detection of IoT malware.

A. Heterogeneity

IoT applications are diverse in terms of processor architecture, libraries, operating systems, devices, and instruction sets. The heterogeneous IoT applications make the analysis system challenging to provide a single solution for all variants.

B. Lack of Standards

There is no pre-defined set of standard protocols, device configuration, and environment for IoT applications. Every manufacturing unit builds the devices with its own set of standards. So, it is very difficult to make a common standard solution that applies to all variants of IoT devices.

C. Availability of Attack Data

The collection of sample data is one of the main challenges faced by the research community. It is because of finding the specific software version for the execution of an attack using a honeypot. Even, there is an authentication problem to pick the data from the available web sources.

D. Insubstantial Environment

It is very challenging to execute the IoT malware on Linux based system with limited resources. Some of the IoT malware may behave indifferently in different conditions and privileges. So, it is difficult to provide root privileges to a malware program for execution.

E. High Computation Infrastructure

IoT devices are having micro-processors and resource-constrained tiny sensors for execution. While the installation of an anti-malware program or detection software required high computation power for implementation.

4. Proposed Methodology and Testbed Design

In this section, proposed methodology of the work is defined that initiate with the collection of malware samples. The samples ae used to analyze the IoT network traffic using filtering. After filtering the data, a signature matching algorithm is applied to recognize the malware signature. If the signature is matched with the pre-defined malware, then it indicates the malware with C & C communication otherwise communication is not established among them. The proposed system architecture is to identify the IP address of the C & C server and the classification of the IoT botnet family (Figure 4). Whenever an IoT bot has established a connection with C & C server, it can be detected by using signature matching or identification of pattern string. Once the pattern or signature is matched, IP addresses of C & C server can be identified. However, it is difficult to differentiate from normal traffic as most of the traffic flows are based on TCP flows.

4.1. Proposed System Architecture

The proposed system architecture is displayed in Figure 4 with the sequence of operations applied. First, we collect

the IoT botnet sample of distinguished families such as Mirai, Bashlite, VPNFilter, and Hajime. The communication traffic is analyzed and thus IP addresses are identified, using them messages are communicated. After the collection of all IP addresses, those IP addresses are filtered out that are involved in the suspicious behavior. Once the suspicious IP address is identified, we inspect the pattern in the behavior of traffic flows, and the signature of malicious patterns are stored. The suspicious behavior of the messages can be detected by using the existing malware signature or similar kinds of repeated messages such as hello flood or any other flood. Besides the identification of IP addresses of C & C server, the classification of the IoT botnet family is also categorized based on the malware behavior. Finally, local sensitive hash (TLSH)-based clustering method is applied to find the similarity index between the malware family.

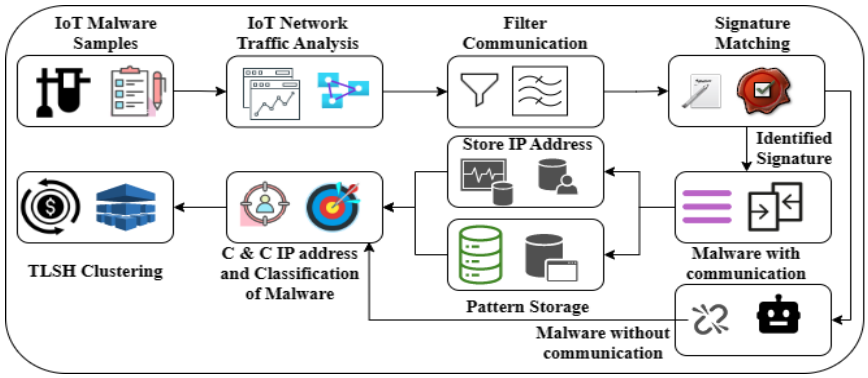

Fig.4. Proposed architectural diagram

After signature matching, there are two different conditions may occur, first, the pattern of the string is matched and the identification of IP address of C & C server, and malware family can be identified. While in the second condition, the signature is not matched with any pre-defined malware family i.e., the communication between the bot and server is not established (Figure 5). In the second scenario, the detection of the IP address together with the malware family is difficult and can be identified by using the characteristics of the malware family. Signatures of the distinguished malware families are stored in the database and then machine learning algorithms are applied for the identification of known behavior. If the signatures of the malware from known behavior are extracted, then malware variant and its family with the IP address of C & C server can be identified. If the behavior of malware is unknown, then repeat the process for better classification and identification.

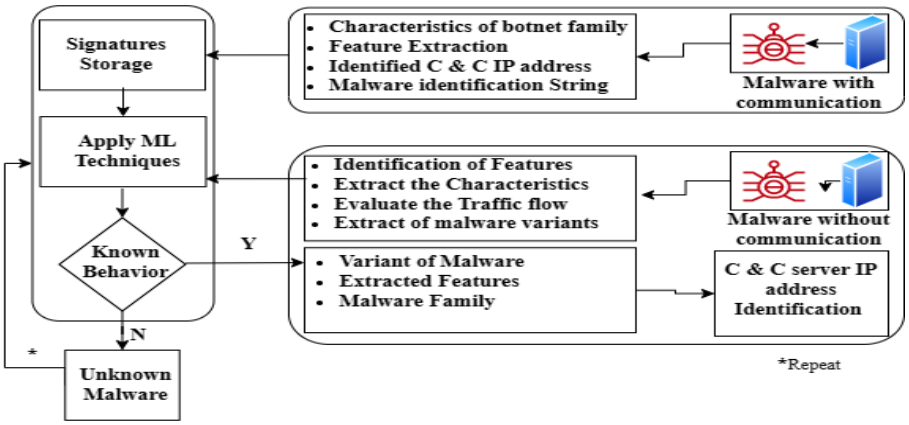

Fig.5. Malware Identification with or without communication

4.2. Testbed Design

The collection of samples is one of the main challenges for this experimental work. Therefore, to collect the samples of IoT malware, we have applied two different approaches:

- Execution of IoT malware using Detux open-source Sandbox in a controlled environment.
- Collect the sample IoT network traffic data provided by existing research works [28].

The experimental testbed has been devised by using a system having 2TB HDD, 16GB RAM, and belongs to a 7 generation Intel i7. A hypervisor VirtualBox is installed on the system as a host for the sandbox. The VM host is a Linux-based operating system with a virtual switch using the bridge and TAP interface. The VM is also used to host a

proxy server for the transformation of HTTPS to HTTP and FTPS to FTP traffic in the simulated Internet. Qemu[1] is used to host IoT devices virtually with different architectures such as I386, PowerPC, and ARM. To implement the IoT-based application, a Linux-based platform is required for each architecture. The auditing or monitoring of these IoT applications is done by using Auditd or TCPdump[2]. Auditd[3] is used to monitor system calls, system activities, and the logging performed by user space. TCPdump is utilized to capture the network traffic files and stored them as PCAP files. A database MongoDB[4] is also used to store and retrieve the results in JSON format. The main script (Diseker) is used for the execution of the sandbox in two modes such as single or interactive execution.

After the implementation of the set-up for the sandbox, two major steps are being performed for the analysis purpose. First, check whether the file is already analyzed or not before updating the database, if not then run the sandbox and update the database. Second, simulate the IoT devices in the sandbox and execute the malware for static and dynamic analysis. In the static analysis, the sandbox executes the simple extractors for retrieving the results and storing them in the database. During dynamic analysis, it is responsible to retrieve the logs by executing and monitoring binary files in the simulated environment. Figure 6 shows the working of the static and dynamic analysis using the sandbox. There are three main modules of the analyzer sandbox such as input/output, control, and simulation module. The first module is used to collect the samples from IoT malware, firmware, and desired parameters which can be processed and requested for the emulated reports. The control module is responsible for orchestration by using different architectures and extracting the logs. It is also used to call the report generator and static analyzer for all compiled processes. In the simulation module, script, configurations, and tools are utilized with a virtual switch and proxy server. The aggregated network traffic is filtered and utilized to generate the syscall logs. The syscall parser provides the information of the log path with EXCVE, PATH, SYSCALL, and SOCKADDR commands. The data is communicated through the internet that may have internal or external network traffic. The network is configured using a group of TAP and bridge interfaces. Sysctl, dhclient, and tun modules are utilized for the bridge interface by using commands such as:

```
$ sudo ip link add et0 type bridge
$ sudo sysctl -w net.ipv4.ip_forward=1
$ sudo tuntap add dev tap0 mode tap
```

The orchestration is responsible for the execution, log generation, and managing the connection with the database. The network traffic or system calls are generated by using parser logs which are conducted by report generator objects. The dynamic analyzer will control the simulation process in the virtual hypervisor and retrieve the logs of malicious scripts.

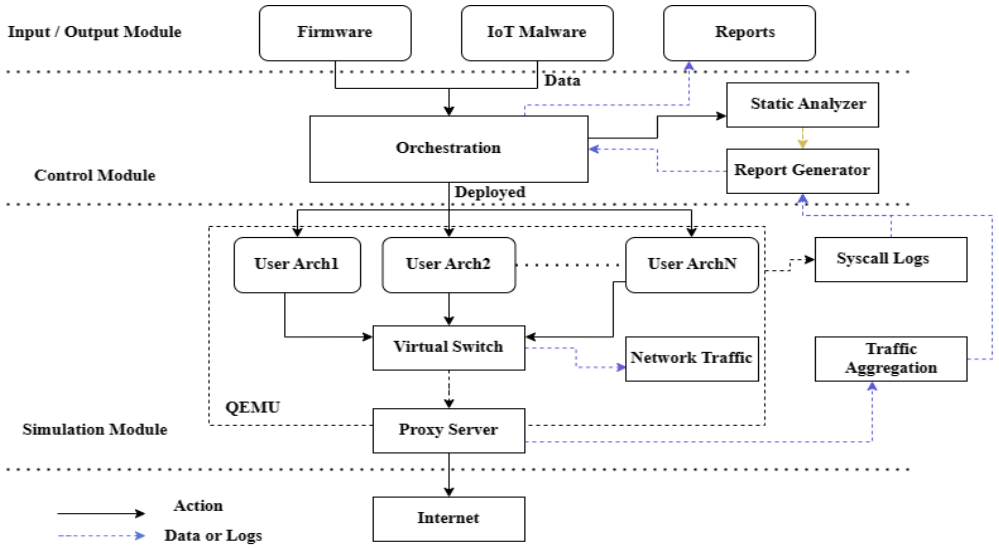

Fig.6. Static and dynamic analyzer architecture and working

5. Result Analysis

After collection of IoT malware samples having count 5747, have been analyzed in this article. The first approach is to distinguish whether the malware based on communication between C & C server and bots has been established or not. The collected PCAP data is segregated as not communicated with public IPs (40% or 2298) intentionally or

[1] https://www.qemu.org/
[2] https://www.tcpdump.org/
[3] https://www.thegeekdiary.com/
[4] https://www.mongodb.com/

unintentionally, and communicated with public IPs (60% or 3449). The PCAP files do not communicate with public IPs therefore we do not consider that data for analysis. While remaining data is considered for analysis purposes that are segregated as communication with the C&C server (34% or 1965) and not in communication with C & C server (26% or 1484) represented in Figure 7. The dataset contains several CPU architectures such as MIPS, ARM, M68K, etc. However, we considered the most used architecture like ARM, X86, and MIPS for comparison with existing malware signatures (Figure 8).

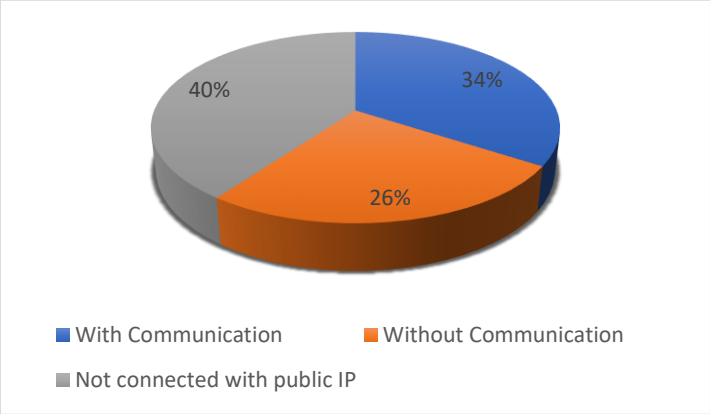

Fig.7. Distribution of sample data

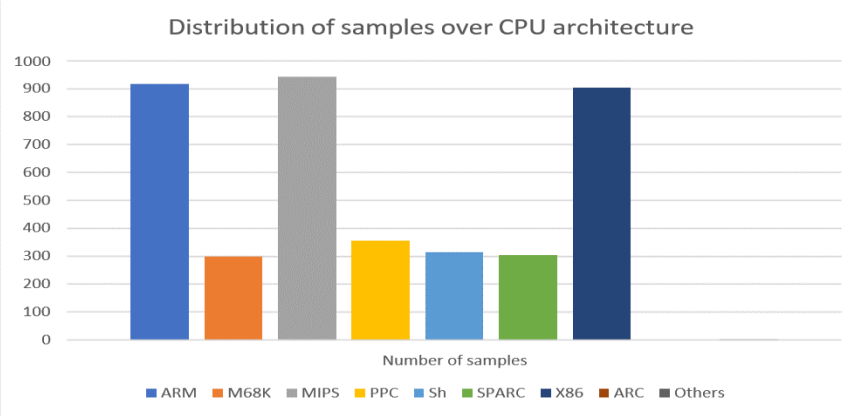

Fig.8. Dataset sample distribution among CPU architecture

Table 2. Some major samples collected from sandbox

SN	Hash File (MD5)	Malware	Linkage	CPU Architecture	Library (C)	Found (%)	IP Found
1	3ee5f6b919203c48f4512ae26a7dfc3f	Router	Dynamic	ARM–32bits	uClibc	5	128.X.X.X
2	dda3c47921bba43b4f33bf0ab27faa13	Dongs	Dynamic	X86–32bits	uClibc	9	193.X.X.X
3	b2b0c9f6cd2a5c9c7d367667739a2744	Razer	Dynamic	ARM-32bits	uClibc	2	80.X.X.X
4	76b40918b492402a696f9c4ac760df31	Get Wrecked	Dynamic	ARM-32bits	uClibc	2	192.X.X.X
5	3951bc82b1e4487a85eaa3986b829c80	Gaygft	Dynamic	X86–32bits	Glibc	4	46.X.X.X
6	11166712561c5b463c08f49d5213c1e0	Prometheus	Static	X86–32bits	Glibc	2	185.X.X.X
7	868788ed5f594fb1de6dac82ae70a700	Pongdupmove	Static	ARM-32bits	Glibc	1	154.X.X.X
8	62379511e6848cbd920c40dc4495b0ce	IoT Wiper	Dynamic	X86-32bits	Glibc	3	113.X.X.X
9	88a9ed5408f20300ea79dd9c9b219379	Reaper	Static	ARM-32bits	Glibc	2	192.X.X.X
10	b0efff0dafe3b234c177b28012c1161d	Kaiji	Static	ARM-32bits	Glibc	1	80.X.X.X
11	7d07e6669ae4f63d08a34a2b3edcd72f	Kinsing	Static	X86-64bits	Glibc	3	128.X.X.X

For analyzing the IoT malware samples, network traffic is captured along linkage, malware family, occurrence percentage, and library users. Table 2 shows the IoT malware family according to their occurrence percentage out of 34%, rest 26% of malware are not identifiable due to the absence of communication between IoT devices and the C&C server. Dongs' family was found the highest in numbers around 188 times from all samples followed by Router and Gaygft. This section analyzes the IoT malware with static and dynamic analysis based on the communication

establishment and in without communication of the C&C server.

5.1. With Established Communication

Once the IoT devices can communicate with the C&C server, the network traffic can be captured by using TCPDump and categorized based on the malware behavior. The Auditd tool is utilized to monitor the system calls and extract the features of IoT network traffic. These features are utilized to generate the behavior of the malware family with existing signatures. This analysis technique is known as static analysis of the malware. Here, in this article, we have considered several malware families. however, we are mainly focused on the Dongs family due to its highest occurrence in the sample. The static analysis for the Dongs family can be done in the following steps:

File name	Name
Size	50 MB
Type	ELF
Architecture	ARM
Hash MD5	65465adfad
Hash SHA1	a65s4df65a4sd6f54as65d
Hash SHA256	wer654a65sd4fa65v64s5d4fa56s4df46awe45f

Fig.9. Details of the dongs family file

Details of the file contents including MD5, Hash1, size, and Hash5 are represented in Figure 9. Entropy is also providing important information regarding the strings extracted from the binary file. There is an entropy dipped for ini_array, .eh_fram, and init_array depict in Figure 10.

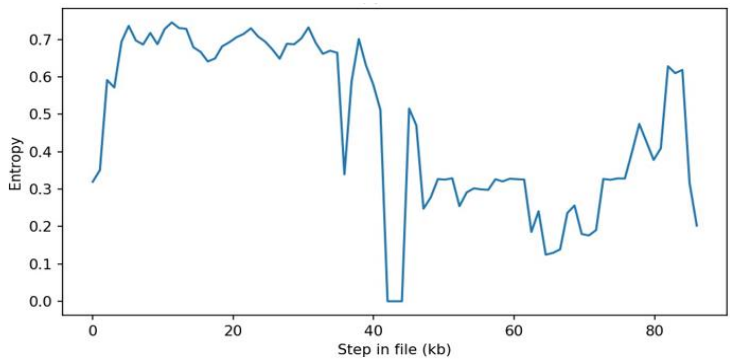

Fig.10. Dongs sample entropy

This string is extracted from the binary file by using the static analyzer. The string is useful to extract the function name including the number of interactions with the function as shown in Figure 11.

134526604	sym.socket_connect	267
134526871	sym.echoLoader	220
134527091	sym.StartTheLelz	5674
134532765	sym.sendSTD	243
134533008	sym.sendUDP	1219
134534227	sym.spoofTest	690
134534917	sym.sendTCP	1193
134536110	sym.sendHTTP	504
134536614	sym.sendHTTP2	526
134537140	sym.sendCNC	165
134537305	sym.processCmd	4893

Fig.11. Function names extracted by static analyzer

The dynamic analysis for this malware can extract the all-possible behavior shown in Figure 12. The malware is executed with PID 268, and PID 270 is generated which is a child process and captures the suspicious behavior.

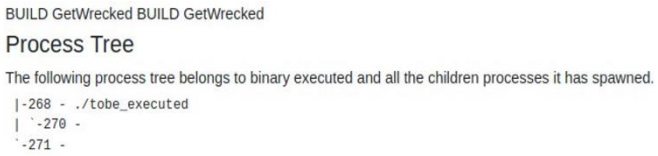

Fig.12. Dynamic Analyzer process tree

After the execution of malware, PID 271 is added to the processes that shows the suspicious behavior.

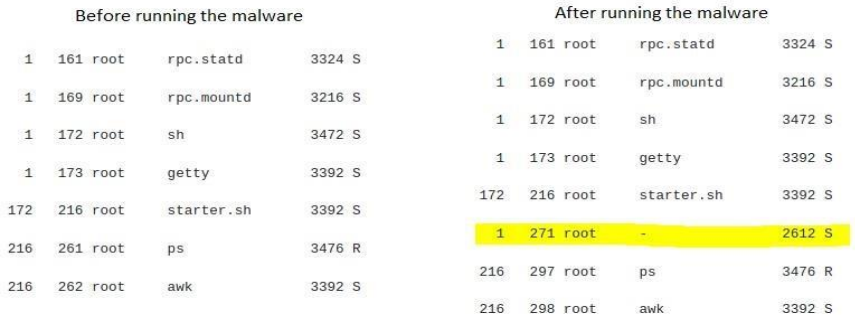

Fig.13. Process states before and after execution of malware

The system calls are captured for PID 268, 270, and 271 for process detection, and found that PID 268 is trying to read the files such as "/etc/rc.d/rc.local" and "/etc/rc.conf", and a successful established communication with IP 193.X.X.X over port 1665. After cloning the file, named "/proc/net/route" is generates a child PID 271 attempting to establish communication with the C&C server (Figure 14).

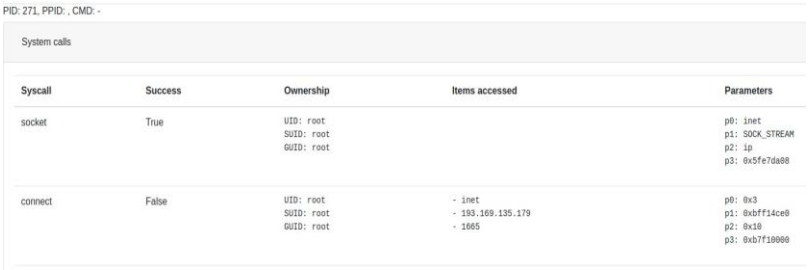

Fig.14. PID 271 system calls

5.2. Without Established Communication

In this section, we focused on the set of malware families which does not have any communication with the C&C server. First, the database is built based on the signature and behavior of the distinguished malware families. Several parameters are considered for the segregation of malware family such as IP address, the total transmission of packets, bit rate, average packets, etc. A few parameters are measured and compared with Dongs' family parameters shown in green color (Figure 15). Then, a set of machine learning algorithms is applied that works to classify the behavior such as KNN, Decision tree, SVM, and Logistic regression.

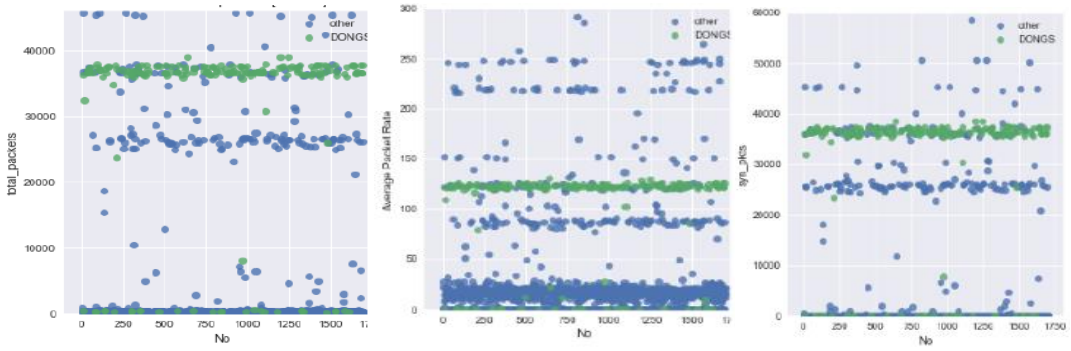

Fig.15. Packets distributions for dongs family

Accuracy is one of the most common parameters for effective evaluation of the classification. Figure 16 shows the accuracy results for the classification of the Dongs family by using distinguished ML algorithms.

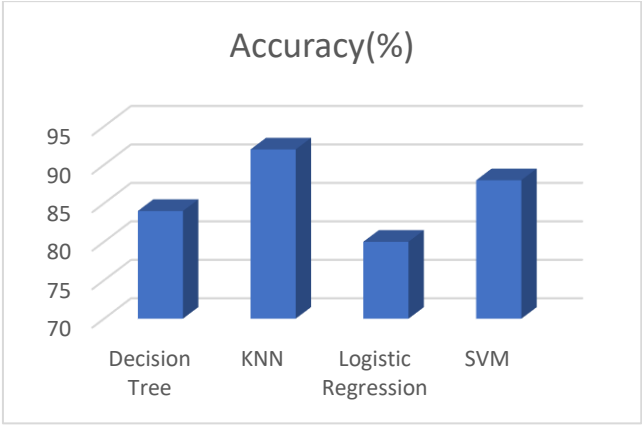

Fig.16. Accuracy comparison of Dongs family

In the case of the KNN algorithm, we acquire better accuracy and less error rate where the value is nearly 10 (Figures 17 and 18).

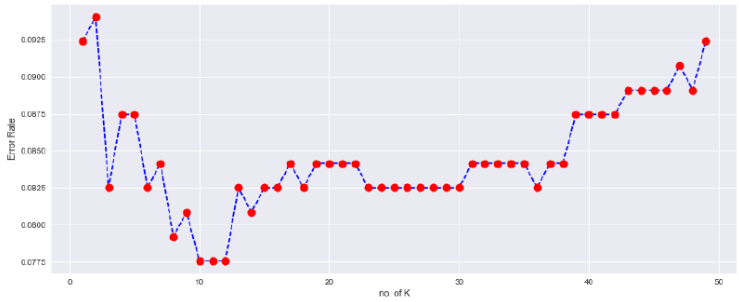

Fig.17. Error rate for KNN

Fig.18. K-value for KNN

5.3. Clustering based on Malware Similarity

A local sensitive hash (TLSH) fuzzy matching program is utilized to cluster the malware based on similarity index. The TLSH digest is measured by using binary entropy evaluation and yet another recursive acronym (YARA) rules. The binary entropy is evaluated by using the Shannon formula $H(x) = -\sum_{i=1}^{n} P(i) \log_2 P(i)$. Figure 19 shows the average distribution of entropy of the selected samples.

There is one of the most critical disadvantages of the entropy evaluation is lower entropy for low entropy bytes. To deal with this limitation, YARA rules are fully customizable malware detection and mitigation patterns. There are three major sections in YARA rules such as string, metadata, and conditions. The string section considers defining strings and variables, while metadata considers storing conditions, metadata, or key-value pair. The YARA rule conditions for the Dongs family are shown in Figure 20.

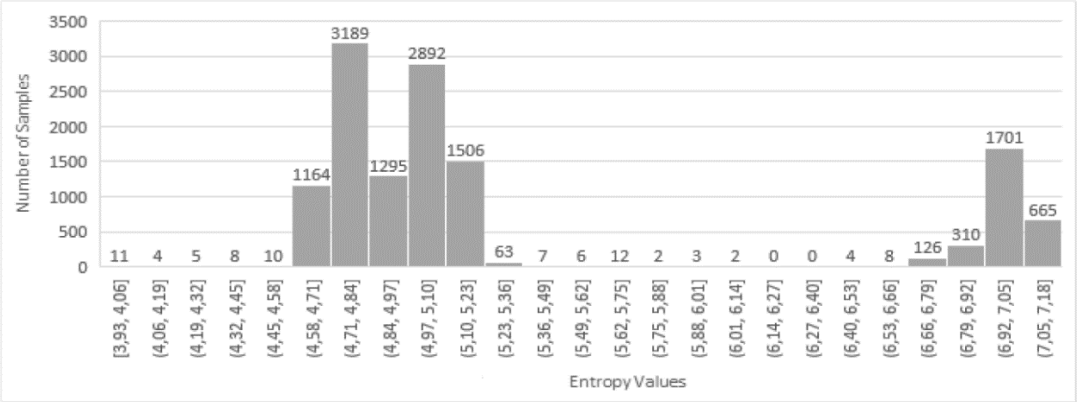

Fig.19. Average Entropy distribution of malware sample

```
condition:
    uint16(0) == 0x5A4D
    and pe.version_info["FileDescription"] contains "Versium Research 5 Installation"
    and pe.imports["FDIDestroy", "FDICopy", "FDICreate"]
    and pe.sections[0].name == "CODE"
    and pe.sections[2].name == "BSS"
    and (($a*) or 3 of ($b*) or 1 of ($c*)
    and filesize < 50000
```

Fig.20. YARA conditions rule for dongs family

Once entropy and YARA rules are applied to the binary files, the k-medoid algorithm is applied. The major aim of this algorithm is to find the value of k similar objects among distinguished objects of datasets. We have considered the TLSH difference or TLSH threshold value lower than 48 with k=17. Table 3 depict that the mean diameter of the cluster is 581.692 while number of clusters are 649.5 with 17 clusters.

Table 3. Parameters measurement for clusters k=17

Parameters	Value
Number of Clusters	17
Mean Diameter	581.692
Maximum Clusters	1258
Minimum Clusters	41
Mean Cluster	649.5

6. Conclusions

The IoT botnet is one of the crucial threats that can control and hack IoT devices or applications. The samples are collected by using an experimental testbed implemented by using sandbox Detux and network traffic analyzer TCPDump tool. After the collection of samples, the segregation of traffic is done based on the communication established with the C&C server and without establishing the communication. The analysis of IoT botnets can be done by using several techniques such as static, dynamic, memory, and hybrid analysis. Static analysis is utilized to obtain the hash keys, functions, and IP addresses of the C&C server. While the dynamic analysis is providing the process states, system calls, and malware features. This article presents a set of IoT botnet analysis techniques for the identification of malware families based on the extracted features from IoT network traffic. We also implemented an intelligent and hybrid model (IHBOT) that is further integrated with machine learning and analysis mechanisms by clustering of similar extracted features. After taking through hybrid analysis, malware is further clustered based on the similarity index that provides better accuracy in the case of k=17. Once the malware families are identified, it can be prevented or removed from the system by alerting the users or applications. The advantage of this hybrid model is to combine the static and dynamic modeling integrated with machine learning to recognition and cluster formation of the IoT botnet traffic. The major limitation of this work is that proposed mechanism cannot identify all possible malware families at the same time. So, in the future, the process can be improvised by classifying and identifying more malware families at a time.

Funding

Not Applicable.

Conflict of Interest

The authors declare that they have no conflict of interest.

References

[1] Informatica, "IoT Data Management: The Rise of Industrial IoT and Machine Learning," 2020. https://www.informatica.com/in/resources/articles/iot-data-management-and-industrial-iot.html (accessed Aug. 10, 2021).

[2] M. Farhoumandi, Q. Zhou, and M. Shahidehpour, "A review of machine learning applications in IoT-integrated modern power systems," *Electr. J.*, vol. 34, no. 1, p. 106879, 2021.

[3] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. V. Pham, S. K. Padannayil, and K. Simran, "A Visualized Botnet Detection System Based Deep Learning for the Internet of Things Networks of Smart Cities," *IEEE Trans. Ind. Appl.*, vol. 56, no. 4, pp. 4436–4456, 2020, doi: 10.1109/TIA.2020.2971952.

[4] Q. D. Ngo, H. T. Nguyen, V. H. Le, and D. H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Express*, vol. 6, no. 4, pp. 280–286, 2020, doi: 10.1016/j.icte.2020.04.005.

[5] M. Malik, M. Dutta, and J. Granjal, "A Survey of Key Bootstrapping Protocols Based on Public Key Cryptography in the Internet of Things," *IEEE Access*, vol. 7, pp. 27443–27464, 2019, doi: 10.1109/ACCESS.2019.2900957.

[6] D. E. Kouicem, Y. Imine, A. Bouabdallah, and H. Lakhlef, "Decentralized Blockchain-Based Trust Management Protocol for the Internet of Things," *IEEE Trans. Dependable Secur. Comput.*, vol. 19, no. 2, pp. 1292–1306, 2022, doi: 10.1109/TDSC.2020.3003232.

[7] A. Costin and J. Zaddach, "IoT Malware: Comprehensive Survey, Analysis Framework and Case Studies," *BlackHat USA*, pp. 1--7, 2018.

[8] M. Gopinath, S. Chakkaravarthy, and S. Ph, "A comprehensive survey on deep learning based malware detection techniques," *Comput. Sci. Rev.*, vol. 47, p. 100529, 2023, doi: 10.1016/j.cosrev.2022.100529.

[9] R. Tanabe *et al.*, "Disposable Botnets: Long-term Analysis of IoT Botnet Infrastructure," *J. Inf. Process.*, vol. 30, no. May 2019, pp. 577–590, 2022, doi: 10.2197/IPSJJIP.30.577.

[10] T. Trajanovski and N. Zhang, "An Automated and Comprehensive Framework for IoT Botnet Detection and Analysis (IoT-BDA)," *IEEE Access*, vol. 9, pp. 124360–124383, 2021, doi: 10.1109/ACCESS.2021.3110188.

[11] Y. Cheng, W. Fan, W. Huang, and J. An, "A Shellcode Detection Method Based on Full Native API Sequence and Support Vector Machine," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 242, no. 1, 2017, doi: 10.1088/1757-899X/242/1/012124.

[12] H. Hashemi and A. Hamzeh, "Visual malware detection using local malicious pattern," *J. Comput. Virol. Hacking Tech.*, vol. 15, no. 1, pp. 1–14, 2019, doi: 10.1007/s11416-018-0314-1.

[13] C.-Y. Wu, T. Ban, S.-M. Cheng, T. Takahashi, and D. Inoue, "IoT malware classification based on reinterpreted function-call graphs," *Comput. Secur.*, vol. 125, p. 103060, 2023, doi: https://doi.org/10.1016/j.cose.2022.103060.

[14] O. Habibi, M. Chemmakha, and M. Lazaar, "Imbalanced tabular data modelization using CTGAN and machine learning to improve IoT Botnet attacks detection," *Eng. Appl. Artif. Intell.*, vol. 118, p. 105669, 2023, doi: https://doi.org/10.1016/j.engappai.2022.105669.

[15] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *J. Comput. Virol. Hacking Tech.*, vol. 12, no. 2, pp. 59–67, 2016, doi: 10.1007/s11416-015-0244-0.

[16] M. S. Abbasi, H. Al-Sahaf, M. Mansoori, and I. Welch, "Behavior-based ransomware classification: A particle swarm optimization wrapper-based approach for feature selection," *Appl. Soft Comput.*, vol. 121, p. 108744, 2022, doi: 10.1016/j.asoc.2022.108744.

[17] T. N. Nguyen, Q. D. Ngo, H. T. Nguyen, and G. L. Nguyen, "An Advanced Computing Approach for IoT-Botnet Detection in Industrial Internet of Things," *IEEE Trans. Ind. Informatics*, vol. 18, no. 11, pp. 8298–8306, 2022, doi: 10.1109/TII.2022.3152814.

[18] G. L. Nguyen, B. Dumba, Q.-D. Ngo, H.-V. Le, and T. N. Nguyen, "A collaborative approach to early detection of IoT Botnet," *Comput. Electr. Eng.*, vol. 97, p. 107525, 2022, doi: https://doi.org/10.1016/j.compeleceng.2021.107525.

[19] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer (Long. Beach. Calif).*, vol. 50, no. 7, pp. 80–84, 2017, doi: 10.1109/MC.2017.201.

[20] M. Catillo, A. Pecchia, and U. Villano, "A Deep Learning Method for Lightweight and Cross-Device IoT Botnet Detection †," *Appl. Sci.*, vol. 13, no. 2, 2023, doi: 10.3390/app13020837.

[21] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, "Epidemic processes in complex networks," *Rev. Mod. Phys.*, vol. 87, no. 3, pp. 1–62, 2015, doi: 10.1103/RevModPhys.87.925.

[22] B. Bojarajulu, S. Tanwar, and T. P. Singh, "Intelligent IoT-BOTNET attack detection model with optimized hybrid classification model," *Comput. Secur.*, vol. 126, p. 103064, 2023, doi: https://doi.org/10.1016/j.cose.2022.103064.

[23] R. Tanabe *et al.*, "Disposable botnets: Examining the anatomy of IoT botnet infrastructure," *ACM Int. Conf. Proceeding Ser.*, no. August, 2020, doi: 10.1145/3407023.3409177.

[24] A. Karim, V. Chang, and A. Firdaus, "Android botnets: A proof-of-concept using hybrid analysis approach," *J. Organ. End User Comput.*, vol. 32, no. 3, pp. 50–67, 2020, doi: 10.4018/JOEUC.2020070105.

[25] J. Jeon, B. Jeong, S. Baek, and Y. S. Jeong, "Hybrid Malware Detection Based on Bi-LSTM and SPP-Net for Smart IoT," *IEEE Trans. Ind. Informatics*, vol. 18, no. 7, pp. 4830–4837, 2022, doi: 10.1109/TII.2021.3119778.

[26] A. A. Hamza, I. T. Abdel Halim, M. A. Sobh, and A. M. Bahaa-Eldin, "HSAS-MD Analyzer: A Hybrid Security Analysis System Using Model-Checking Technique and Deep Learning for Malware Detection in IoT Apps," *Sensors*, vol. 22, no. 3, 2022, doi: 10.3390/s22031079.

[27] M. M. Alani, "BotStop : Packet-based efficient and explainable IoT botnet detection using machine learning," *Comput. Commun.*, vol. 193, pp. 53–62, 2022, doi: https://doi.org/10.1016/j.comcom.2022.06.039.

[28] T. Trajanovski and N. Zhang, "An automated and comprehensive framework for IoT botnet detection and analysis (IoT-BDA)," *IEEE Access*, vol. 9, pp. 124360–124383, 2021.

Authors' Profiles

Umang Garg is currently pursuing Ph.D. from Graphic Era Deemed to be university, Dehradun, India in the area of IoT security. He is working as an assistant professor in Graphic Era Hill University, Dehradun, India since 2017. The work has been successfully performed under the supervision of Dr. Santosh Kumar. His area of interest includes the IoT security, IoT botnet analysis, Machine leaning, and deep learning.

Santosh Kumar earned his Ph.D. in 2012 from the India Institute of Technology in Roorkee, India, his M. Tech. in Computer Science and Engineering in 2007 from Aligarh Muslim University in Aligarh, India, and his B.E. (IT) in 2003 from C.C.S. University in Meerut, India. He has more than 75 articles, 10 book chapters and authored 01 book titled "Component based Software Engineering" with Taylor and Fanscis, November 2020. He has been served on the editorial/review boards of several National and International Journals and Conferences. He is a Senior Member of ACM, IEEE, IAENG, ACEEE, and ISOC (USA) and has contributed over 75 research papers in national and international journals/conferences on Wireless Communication Networks, WSN, IoT, Artificial Intelligence, Grid Computing, and Software Engineering. Currently, he is a Professor and Chairman, DRC of Doctoral (PhD) programme in the Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun (India).

Dr. Manoj Kumar is working as an Associate Professor in the Department of Computer Science and Engineering at Graphic Era Deemed to be University, Dehradun, Uttarakhand, India-248802. He has almost 14 years of experience in academics and mostly on an Industrial scale in the field of Cyber Security. He has B.Tech (CSE) from Cochin University of Science and Technology, MTech(CSE), and Ph.D. (CSE) with specialization in Cyber Security from the National Institute of Technology Jalandhar (Punjab)-144011. He has published several research articles in reputed journals. He is an active reviewer board member in various national/International Journals and Conferences. He has memberships in ACM (Senior Member), IAENG, ACEEE, and ISOC (USA). His research interests include Visual Cryptography, Identity-based encryption, ID-based signature, Elliptical curve cryptography, Zero Knowledge Proof based authentication, and Signcryption Techniques.

How to cite this paper: Umang Garg, Santosh Kumar, Manoj Kumar, "IHBOT: An Intelligent and Hybrid Model for Investigation and Classification of IoT Botnet", International Journal of Computer Network and Information Security(IJCNIS), Vol.16, No.5, pp.98-112, 2024. DOI:10.5815/ijcnis.2024.05.08