JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

New Techniques for Protection of IoT Devices from Malicious Behavior Using Working Set Based System Call Whitelisting and Argument Clustering

Lakshmi Eswari Ponnapu Reddy
Centre For Development Of Advanced Computing (C-Dac)
Hyderabad, India
prleswari@cdac.in

Sarat Chandra Babu Nelaturu
Society For Electronic Transactions And Security (Sets)
Chennai, India
sarat@setsindia.net

Received 2022 March 15; **Revised** 2022 April 20; **Accepted** 2022 May 10.

Abstract - The rapid evolution of Industry 4.0 and the spread of Internet of Things (IoT), is supporting the growth of cyber-physical systems for societal applications. It is challenging to design secure IoT devices, due to constrained computational and storage resources. The vulnerabilities in the deployed IoT devices are exploited by the attackers for carrying out malicious activities. Various anomaly detection approaches are proposed in literature for detecting malicious behavior at runtime. However they are not suitable for resource constrained IoT devices. In this paper, we propose new techniques for detecting runtime intrusions and protecting IoT devices using working set based system call whitelisting and argument clustering. Proposed system call whitelisting technique separates system call whitelists of initialization and service phases of field deployed IoT device, resulting in the reduced attack surface. We evaluated the proposed technique on Tenda AC15 version 15.03.05.19 for Telnet service. The experimental results show that the proposed working set based system call whitelisting successfully reduced 44% of system calls during the initialization phase and 40% of system calls during service phase. In addition to this, we used system call argument clustering technique, to augment the detection of malicious behavior which is injected at runtime through modifying the arguments of whitelisted system calls.
Index Terms - IoT Security, Malicious Behavior, Whitelisting, Working Set, Argument Clustering, Attack Surface

I. INTRODUCTION

The rapid spread of Internet of Things (IoT), is resulting in the phenomenal growth in the design of cyber-physical systems for societal application domains, such as Smart Cities, Healthcare, Smart homes, Transportation, Communication and Agriculture as depicted in Fig. 1.

It is predicted that 152200 IoT devices would be connected every minute by 2025 [1]. These IoT devices vary from home routers, IP cameras to Industrial Control Systems. At the same time, it is becoming difficult to enable secure operation of IoT applications due to unauthorized access by attackers and injected malicious behaviour. Details of IoT security threats are depicted in Fig. 2. As per the report [2], 57% of IoT devices are vulnerable to medium/ high-severity attacks. Further, 41% of attacks exploit IoT device vulnerabilities. Attackers use hardware, firmware, application software and connections of the IoT devices as attack surface. As these devices are connected to the Internet, attacker can scan them through the attack surface looking for open ports, vulnerabilities in the web interface, active network services (Telnet, SSH) and so on. Sophisticated attacks are also launched by exploiting the vulnerabilities in software such as buffer overflow and SQL injection.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

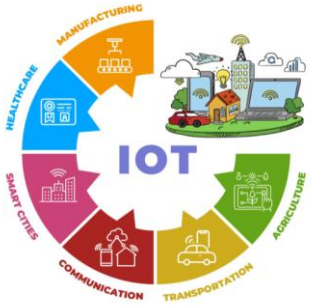

Fig. 1 IoT Application

When the IoT device is compromised and malicious behavior is introduced, it can cause significant impact on the functioning of the device and network. Some of the widely known malware in IoT networks include Mirai, IoT Reaper, PNScan, Gafgyt and Hajime. In Oct 2016, Mirai botnet was used to launch DDoS attack on Dyn, DNS Provider, infecting 1.2 million IoT devices, which brought down GitHub and Twitter and made them inaccessible for hours [3]. IoT Reaper is a botnet which is more dangerous than Mirai, spreading and damaging millions of vulnerable IoT devices. Reaper targets devices with unpatched vulnerabilities [4]. The scale at which botnets that depend on IoT devices are spreading is multifold.

Furthermore, over the last few years the malware threat landscape in IoT devices has evolved from traditional file-based malware to sophisticated and file-less malware [5]. Traditional file-based malware leverages executable files to carry out malicious activities, whereas file-less malware does not create any entry in the file system and is capable of evading any anti-virus solutions. File-less malware attacks on IoT devices are also increasing [6]. As in today's cyber physical systems the IoT devices are important endpoints, their security has become essential.

Fig. 2 IoT Security Threats

Generally while designing the IoT based applications, focus is on core application functionality due to constrained resources of the devices, short time to market and also looking for affordable end product, ignoring the security aspects. Hence, security by design and incorporating computationally intensive security measures is not a focus and also standardization of IoT device security is not properly established. This leads to serious vulnerabilities in the deployed IoT devices which can be targeted through Internet by attackers. These vulnerabilities in IoT devices are exploited by attackers for carrying out various malicious activities. Considering the constrained resources and also the limitation in incorporating update mechanism in IoT devices, we propose light-weight anomaly detection techniques using system calls. Proposed techniques, working set based system call whitelisting and argument clustering found to be appropriate for detecting runtime intrusions in IoT devices. We evaluated the proposed solution on Tenda AC15 version 15.03.05.19 and the experimental results are promising.

In this paper, Section 2 covers Background and Related Work. Details of Proposed Approach are presented in Section 3. In Section 4, Design and Implementation aspects of the proposed solution are explained. Section 5 covers details of Experimental Evaluation. Conclusions are given in Section 6 and references are provided at the end.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

II. BACKGROUND AND RELATED WORK

Traditionally antivirus solutions are developed to protect endpoints. They make use of Misuse detection approach, where malware is detected and blocked by searching for patterns of malware signatures within the endpoint. Unfortunately many of the IoT devices cannot run the antivirus software or other advanced security solutions used in endpoints, such as desktops, laptops, due to their limited computing and storage resources. Additionally, antivirus solutions require continuous updating of signatures/ rules/ behaviors, which is going to be a challenge in IoT devices.

In addition, application whitelisting approach is used in endpoint security solutions to complement antivirus functionality towards detecting the malware penetrating through unknown files/ executables. Granularity of whitelisting varies based on executable hash, path name, libraries and so on. However these solutions may not be able to protect from intrusions/ attacks launched through vulnerabilities in whitelisted files/ executables at runtime. Malicious behavior can reach the device by exploiting the vulnerabilities in firmware/ applications at runtime. Anomaly based approach is used to detect the deviations from the normal behavior at runtime. Different anomaly based approaches using system calls are proposed in literature for detecting intrusions/ attacks at runtime. However they are not suitable for resource constrained IoT devices.

Furthermore, even though commercial security solutions such as McAfee Embedded Control, Trend Micro OfficeScan, Kaspersky Embedded Systems Security and Symantec Critical System Protection are available for embedded systems, they are not suitable for the resource constrained IoT devices. There are various research initiatives towards using system call based anomaly detection approaches to the resource constrained IoT devices.

Wu et al. [7] proposed a light-weight whitelist based protection titled IoTProtect, which uses whitelist to check and terminate the unknown processes at regular intervals of time. Breitenbacher et al. [8] proposed a host-based anomaly detection system for IoT devices, which provides proactive detection and tamper-proof resistance. This solution is based on whitelisting and utilizes system call interception, performed within the loadable kernel module (LKM), which monitors and stops any unauthorized program before its execution.

Paleari et al. [9] proposed and discussed a method which extracts the behavior of applications through system calls and monitors in emulator. They have also proposed a method to construct recovery procedures by using clustering the behavior of malware approach. Tamiya et al. [10] proposed a method for disinfecting IoT devices by resetting or rebooting the infected devices. However they have not proposed any detection method [10].

WhiteEgret [11] proposed a whitelisting-based execution control technique, which uses bprm_check_security hook and the mmap_file hook to monitor the absolute path of executable files. Harada et al. [12] proposed secure Linux OS by enforcing mandatory access control (MAC) using TOMOYO Linux and SELinux. Kernel space calls are used to check the validity with predefined policies.

In [13], authors examined Mirai malware and its propagation in detail, proposed and implemented application whitelisting to effectively combat IoT malware using fanotify feature of Linux kernel. In [14], authors proposed resource clustering method for building model based on systems calls made by an application. This is implemented on Windows using mini filter driver for capturing the application behavior.

We propose a light-weight anomaly detection solution using system calls to protect Linux based IoT devices from malicious behavior. Details of proposed approach are covered in next section.

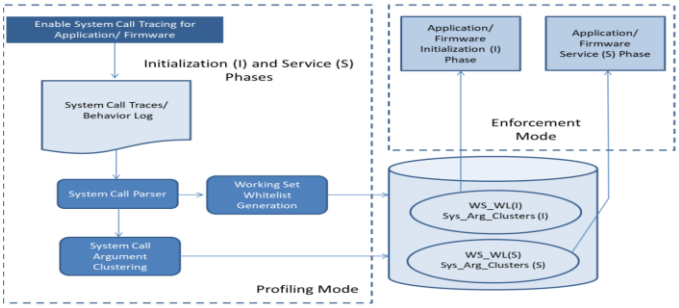

Fig. 3 System Architecture

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

III. PROPOSED APPROACH

When compared with general purpose computing devices such as desktops and workstations, IoT devices have predefined and stable functionality. Considering these aspects of IoT devices and also the growing trends in IoT malware, we propose light-weight anomaly detection techniques. We propose working set based system call Whitelisting and argument clustering. These techniques are proposed for Linux based IoT devices, as it is one of the preferred operating system in those devices. Proposed technique separates the system call whitelists used during initialization and service phases of the IoT device in operation, which reduces the attack surface. System call argument clustering helps to detect the runtime intrusions made by modifying the arguments of whitelisted system calls. Next section covers Design and Implementation details of proposed approach.

IV. DESIGN AND IMPLEMENTATION

System calls are the entry points for applications/ firmware to access the kernel functionality. Even attackers also depends on systems calls to carry out the malicious activities by exploiting the vulnerabilities and injecting the code at runtime. In order to detect intrusions at runtime, various system call based anomaly detection techniques are proposed in literature. We propose light-weight working set based system call whitelisting and system call argument clustering techniques which are found to be appropriate for detecting intrusions at runtime in IoT devices.

A. System Architecture

System architecture of proposed Working Set based System Call Whitelisting and Argument Clustering solution is given in Fig. 3. Every IoT device has two major phases in execution, initialization phase (I) and service phase (S). Initialization phase includes booting of the IoT device and initialization of various services such as telnet and http. Once initialized, during service phase, IoT device offers services based on its predefined functionality, by accepting the requests and responding accordingly. As, IoT device has limited and predefined functionality, system call based whitelisting is a promising approach to detect runtime malicious injections in IoT devices. Further by applying working set based system call whitelisting, system call whitelists can be separated for initialization and service phases, which helps to restrict phase-wise whitelists and reduce the attack surface.

Proposed solution works in two modes, profiling and enforcement modes. During profiling mode of an IoT device, system call traces TI and TS are captured for initialization and service phases respectively. These traces are fed to system call parser, which parses the system call names and arguments and gives the details to working set whitelist generation and System call argument clustering modules. These modules would generate the working set based whitelists (WS_WL(I) and WS_WL(S)) and system call argument clusters (Sys_Arg_Clusters(I) and Sys_Arg_Clusters(S)) for Initialization and Service phases, which are stored in the database. When an IoT device is in the field, it would be enabled in enforcement mode with working set based whitelists and system call argument clusters enforced during the initialization and service phases. Details of Working Set based System Call Whitelisting and System Call Argument Clustering approaches are described in the next sub sections.

B. Working Set based System Call Whitelisting

Considering the constrained resources of an IoT device with predefined functionality, we propose Working Set based System call Whitelisting as depicted in Fig. 4. This technique maintains separate whitelists corresponding to the different phases of execution of an IoT device. Due to the specific functionality of initialization and service phases, there is a requirement of separate system call working set whitelists, WS_WL(I) and WS_WL(S). These are generated and maintained separately, which are enforced during the corresponding phases at runtime. Some of the system calls of initialization phase which are accessed during the booting and initialization of an IoT device, are not required during the service phase of an IoT device. Similarly some of the system calls accessed during the service phase may not be relevant during initialization phase. Phase wise enablement of system call whitelists helps to avoid misuse of system calls, which results in reducing the attack surface.

Note that S is the system call set $\{s_1, s_2, s_3, \ldots, s_n\}$, supported by the operating system kernel (Linux) and n is the number of system calls. For process P, we denote TI and TS as the system call traces captured for Initialization and

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

Service phases. Each trace includes all system calls along with arguments accessed by binary/ executable in that specific phase.

SI and SS are the system call sets derived from the TI and TS traces respectively. SI and SS are the subsets of S and there can be some common system calls present in both SI and SS. SI and SS sets are the working set whitelists, WS_WL(I) and WS_WL(S), for Initialization and Service phases.

During profiling mode, by using dynamic program analysis, system call traces for initialization and service phases of IoT device are captured and Working Set Whitelists WS_WL(I) and WS_WL(S) are generated and stored in the database for runtime enforcement of the whitelists during enforcement mode.

C. System Call Argument Clustering

Working set based system call whitelisting helps to reduce the attack surface, by restricting the system call whitelists specific to phases as mentioned before. However by changing the arguments to whitelisted system calls, mimicry attacks can be launched to carry out malicious activity. For example, new binaries can be introduced and executed using whitelisted *execve* system call. Similarly, it is possible to carry out malicious activity by introducing different arguments to *open* system call for opening new files or change the file contents/ permissions without introducing new files in file system. Therefore the same system call can be part of normal behavior as well as malicious behavior. To complement the working set based whitelisting and address the above mentioned challenges, we propose system call argument based clustering technique for the critical system calls such as *open* and *execve*.

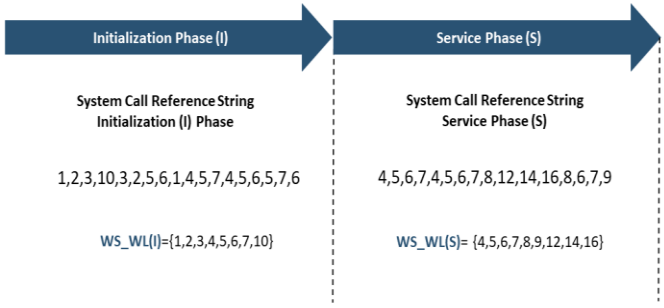

Fig. 3 Working Set based System call Whitelisting

During profiling mode, while generating the working set based system call whitelists (WS_WL(I) and WS_WL(S)), critical system calls such as *open*, *execve* are selected from TI and TS.

The system call argument clusters Sys_Arg_Clusters(I) and Sys_Arg_Clusters(S) are created for those system calls. This involves capturing the arguments of the identified system calls in the entire process flow and generating clusters for the same. Considering the limited and predefined functionality of IoT device, this approach is practical to implement in effectively capturing all system calls and arguments.

We consider that, s_i and s_j from S are identified as critical system calls of a process P for capturing the system call argument clusters.

Initialization Phase

We denote TI_{ci} and TI_{cj} defined as $TI_{ci} = \{Ici_1, Ici_2, ….., Ici_{m1}\}$ $TI_{cj} = \{Icj_1, Icj_2, …., Icj_{m2}\}$ as the clusters for s_i and s_j system calls generated during Initialization Phase, which includes all the references of s_i and s_j in TI. *m1* and *m2* are the number of references to s_i and s_j system calls in TI.

Next $Ici_j = <arg_1, arg_2,…, arg_m>$, Ici_j gives the details of jth cluster of s_i in TI and *m* is the number of arguments of s_i system call.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

During Initialization Phase, Sys_Arg_Clusters(I) defined Sys_Arg_Clusters(I)= $\{TI_{ci},TI_{cj},\ldots,TI_{cp}\}$, is generated which includes clusters for all the identified system calls. These are captured during the Initialization phase in profiling mode for all identified critical system calls.

Service Phase
We also denote TS_{ci} and TS_{cj} defined as $TS_{ci} = \{Sci_1, Sci_2,\ldots.., Sci_{m1}\}$ $TS_{cj} = \{Scj_1, Scj_2,\ldots., Scj_{m2}\}$

TS_{ci} and TS_{cj} are the clusters for s_i and s_j system calls generated during Service Phase which includes all the references of s_i and s_j in TS. *m1* and *m2* are the number of references to s_i and s_j system calls in TS.

$Sci_j = <arg_1,arg_2,\ldots.arg_m>$, Sci_j gives the details of jth cluster of s_i in TS and *m* is the number of arguments of s_i system call.

During Service phase, Sys_Arg_Clusters(S) defined as Sys_Arg_Clusters(S)= $\{TS_{ci}, TS_{cj},\ldots, TS_{cp}\}$, is generated which includes clusters for all the identified system calls. These are captured during the Service phase in profiling mode for all identified critical system calls.

Enforcement Mode
System call arguments clusters for critical system calls which are captured during profiling mode are later enforced during the enforcement mode. In case of any new cluster traced during runtime with different file name or change in permissions, which is not there in Sys_Arg_Clusters(I) and Sys_Arg_Clusters(S), it would be detected and blocked.

D. Implementation

We have applied the proposed techniques on system call traces captured using strace utility on Linux to detect the intrusions at runtime. However, to implement the proposed techniques in enforcement mode, we can use Seccomp-BPF of Linux to restrict the system calls allowed for the firmware/ application of IoT device during Initialization and Service phases. This helps to prevent the malware behavior at runtime. In case if the proposed techniques have to be used for detecting the deviations, we can depend on Falco and AuditD functionality integrated with Linux. Both depend on system call based policies for detection on intrusions.

Another approach for designing the solution is using Edge based detection of intrusions. Working Set Whitelists and System call argument clusters of Initialization and Service phases of IoT devices are maintained at Edge Server. Behavior profiles of IoT devices in terms of system calls are logged and periodically sent to Edge Server. This helps to offload all the computationally intensive activities to edge server towards detection of runtime intrusions in IoT devices.

V. EXPERIMENTATION EVALUATION

Experimentation is carried out on Tenda AC15 router with firmware version 15.03.05.19 which is based on Linux. To avoid the recompilation of the firmware with the required software for capturing the system calls, ARMX (currently EMUX) an emulation framework is used. For working with the firmware images, one has to make device entry in the ARMX framework. Advantage of ARMX is that files and binaries can be added to the filesystem without recompilation. strace utility compiled for ARM platform is added to the Tenda AC15 firmware image through ARMX and system call traces of telnet initialization and service are captured.

The working set based system call whitelist for telnet initialization phase in Tenda AC15 version 15.03.05.19 is as follows

WS_WL(telnet_init) = {execve, mmap2, open, listen, fstat, stat, read, close, munmap, mprotect, ioctl, getuid32, fork, exit}

Similarly the working set based system call whitelist for telnet service phase in tenda AC15 version 15.03.05.19 is given as

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

WS_WL(telnet_service) = {select, accept, fcntl64, brk, open, ioctl, setsockopt, write, vfork, read, access, fcntl, gettimeofday, lseek, close}

Working Set based Whitelists of Telnet is given in Fig. 5. From the Whitelists of Telnet, WS_WL(telnet_init) WS_WL(telnet_service) obtained during the Initialization and Service phases, it is very clear that system calls common to both phases are limited. By separating the system call whitelists of initialization and service phases, attack surface is reduced. System calls relevant to only the initialization phase can be removed in service phase and vice versa, thus reducing the risk as these system calls can be misused by attackers in other phase.

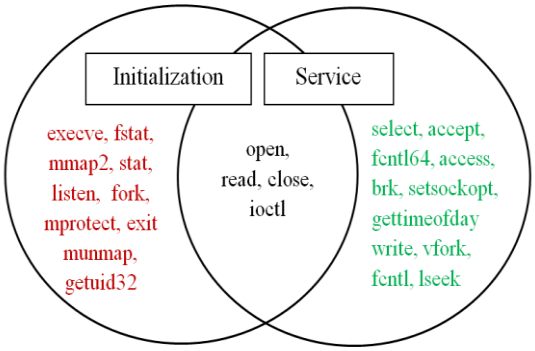

Fig. 4 Working Set based Whitelists of Telnet

From the experimental results on Tenda AC15 version 15.03.05.19 for telnet service, the system calls accessed during initialization phase are 14. Similarly the system calls accessed during service phase are 15. System calls common to both the phases are 4. From these results, it is observed that the proposed approach successfully reduced 44% of system calls during the initialization phase and 40% of system calls during service phase.

From the telnet results, *execve* system call is accessed only during Initialization Phase. Note that if *execve* system call is allowed during Service Phase, it can be misused by attackers to carry out malicious activities, such as creating a shell. Similarly, *listen* system call used to open a network service during initialization phase can be misused to open unauthorized network service if allowed in the whitelist of service phase. A hard-coded telnet credential in the tenda_login binary of Tenda AC15 AC1900 version 15.03.05.19 allowed unauthenticated remote users to restart telnetd during service phase, which is detected through the working set whitelisting technique. Further, execution of any malicious binaries such as bots dropped during the service phase can be detected. From the experimental results of telnet, it is very clear that by separating the whitelists for initialization and service phases, attack surface is reduced, thus avoiding the exploitation of unused system calls.

However by using the separate whitelists, it is still possible to carry out attacks through whitelisted system calls of that phase, therefore System call argument clusters for system calls such as *execve* and *open* is proposed where argument clusters would be whitelisted and maintained in the database.

Sys_Arg_Clusters(telnet_init) for *execve* and *open* system calls are as follows:

Clusters_execve – Cluster1</usr/sbin/telnetd>

Clusters_open – Cluster1</.armx/tenda_hooks.so, O_RDONLY>, Cluster2</.armx/libnvram-armx.so, O_RDONLY>, Cluster3</lib/libcrypt.so.0, O_RDONLY>, Cluster4</lib/libm.so.0, O_RDONLY>, Cluster5</lib/libc.so.0, O_RDONLY>, Cluster6</dev/urandom, O_RDONLY >, Cluster7</dev/null, O_RDWR|O_LARGEFILE>

Sys_Arg_Clusters(telnet_service) for *execve* and *open* system calls are as follows:

From the experimental results of telnet, there are no clusters for execve in service phase.

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

Clusters_open - Cluster1/dev/ptmx, O_RDWR|O_LARGEFILE>, Cluster2</var/run/utmp, O_RDWR|O_CLOEXEC >, Cluster3</var/log/wtmp, O_WRONLY|O_APPEND>, Cluster4</dev/ptmx, O_RDWR|O_LARGEFILE>, Cluster5</var/run/utmp, R_OK|W_OK>, Cluster6<var/run/utmp,O_RDWR|O_CLOEXEC>, Cluster7</var/log/wtmp, O_WRONLY|O_APPEND >

By using the system call argument clusters, runtime intrusions through whitelisted system calls are detected. Opening or execution of any files/ binaries which are deviating from the system call argument clusters identified during the profiling mode are detected in the field during the service phase of IoT device in enforcement mode.

VI. CONCLUSIONS

In this paper, we proposed light-weight working set based system call whitelisting and argument clustering techniques, which are found to be appropriate for protecting IoT devices from malicious behavior. By separating the system call whitelists of initialization and service phases, we can reduce the attack surface. We evaluated the proposed solution on tenda AC15 version 15.03.05.19 for telnet service by using the strace utility for capturing the system calls. Experimental results show that the proposed solution successfully reduced 44% of system calls during the initialization phase and 40% of system calls during service phase. System call argument clustering technique supplemented the working set based whitelisting. As IoT devices are having limited functionality and implementing update mechanism is challenging, proposed techniques help to effectively restrict the attacks on IoT devices in the field.

REFERENCES

[1] Newsroom, “Gartner Identifies Top 10 Strategic IoT Technologies and Trends,” Gartner, Barcelona, Spain, November 7, 2018. Accessed on: December 10, 2021. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends.

[2] Unit 42, “IoT Threat Report,” Paloalto Networks, March 10, 2020. Accessed on: December 10, 2021. [Online]. Available: https://unit42.paloaltonetworks.com/iot-threat-report-2020/ .

[3] R. Hallman, J. Bryan, G. Palavicini, J. Divita, and J. Romero-Mariona, “IoDDoS - The Internet of Distributed Denial of Sevice Attacks - A Case Study of the Mirai Malware and IoT-Based Botnets,” 2nd International Conference on Internet of Things, Big Data and Security (IoTBDS 2017), Pages 47-58.

[4] Joe Curtis, “IoT Reaper 'will be worse than Mirai'," ITPro, October 23, 2017. Accessed on: December 10, 2021. [Online]. Available: https://www.itpro.co.uk/malware/29783/iot-reaper-will-be-worse-than-mirai.

[5] Avira Protection Labs, “Malware Threat Report: Q3 2020 Statistics and Trends,” 10 November 2020. Accessed on: December 12, 2021. [Online]. Available: https://www.avira.com/en/blog/malware-threat-report-q3-2020-statistics-and-trends.

[6] Fan Dang, Zhenhua Li, Yunhao Liu, Ennan Zhai, Qi Alfred Chen, Tianyin Xu, Yan Chen, and Jingyu Yang, “Understanding Fileless Attacks on Linux-based IoT Devices with HoneyCloud,” in proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services, June 2019 Pages 482–493.

[7] Chun-Jung Wu, Ying Tie, Santoshi Hara, Kazuki Tamiya, Akira Fujita, Katsunari Yoshioka, and Tsutomu Matsumoto, “IoT Protect: Highly Deployable Whitelist-based Protection for Low-cost Internet-of-Things Devices,” Journal of Information Processing, Vol-26, pp 662-672, Sep. 2018.

[8] Dominik Breitenbacher, Ivan Homoliak, Yan Lin Aung, Nils Ole Tippenhauer, and Yuval Elovici, “HADES-IoT: A Practical Host-Based Anomaly Detection System for IoT Devices (Extended Version),” in proceedings of the 2019 ACM Asia Conference on Computer and Communications Security., July 2019, Pages. 479–484.

[9] R. Paleari, L. Martignoni, E. Passerini, D. Davidson, M. Fredrikson, J.T. Giffin, and S. Jha, “Automatic Generation of Remediation Procedures for Malware Infections,” USENIX Security Symposium., pp.419-434 (2010).

[10] K. Tamiya, S. Nakayama, Y. Ezawa, Y. Tie, C. Wu, D. Yang, K. Yoshioka, and T. Matsumoto, “Experiment on removal and Prevention of IoT malware using real devices,” Symposium on Cryptography and Information Security 2017, Session 3E1-5 (2017).

[11] M. Kioke, N. Ogura, S. Takumi, Y. Hanatani, and H. Haruki, “Development of WhiteEgret: A Whitelisting-type Execution Control on Linux,” Computer Security Symposium 2017, Session 3D3-4 (2017).

JOURNAL OF ALGEBRAIC STATISTICS
Volume 13, No. 1, 2022, p. 178-186
https://publishoa.com
ISSN: 1309-3452

[12] T. Harada, T. Horie, and K. Tanaka, "Task Oriented Management Obviates Your Onus on Linux," Linux Conference, Vol.3, p.23 (2004).

[13] Tatikayala Sai Gopal, Meerolla Mallesh, G Jyostna, P Reddy Lakshmi Eswari and E Magesh, "Mitigating Mirai Malware Spreading in IoT Environment," International Conference on Advances in Computing, Communications and Informatics (ICACCI), Sep 2018.

[14] Grandhi Jyostna, Himanshu Pareek, and P. R. L. Eswari, "Detecting Anomalous Application Behaviors using a System call clustering Method over Critical Resources," Advances in Network Security and Applications. CNSA 2011. Communications in Computer and Information Science, Vol 196. Springer.