Device-Centric Ransomware Detection using Machine Learning-Based Memory Forensics for Smart Inverters

Alycia M. Jenkins
Texas A&M University-Kingsville
alycia.jenkins@students.tamuk.edu

BoHyun An
Texas A&M University-Kingsville
bohyun.ahn@students.tamuk.edu

Alve Akash
Texas A&M University-Kingsville
alve.akash@students.tamuk.edu

Taesic Kim
Texas A&M University-Kingsville
taesic.kim@tamuk.edu

ABSTRACT
Ransomware attacks are the fastest-growing form of cyberattacks worldwide. Recently, ransomware attacks have targeted industrial control systems (ICSs), including power grids. Lessons learned from recent incidents in ICSs show that ransomware groups can deliver ransomware into not only the organization's control servers, but also the operational technology (OT) devices such as smart inverters and smart grid devices. This paper proposes a machine learning (ML)-based memory forensics method enabling the detection of ransomware binaries stored in the memory of a commercial smart inverter. Device firmware binary files are extracted from a Serial Peripheral Interface (SPI) flash memory, and samples of both benign and ransomware binaries are generated by a binary manipulation method and a real-world ransomware encryption, separately. A deep transfer learning (DTL) method is used to retrain a convolutional neural network (CNN)-based ransomware detection algorithm using the generated samples. The experimental result validates that the proposed ML-based memory forensics method can accurately detect ransomware files.

KEYWORDS
Cybersecurity, deep transfer learning, distributed energy resources, machine learning, memory forensics, ransomware, smart inverter.

1. INTRODUCTION
Smart inverters are networked power electronics devices that transfer generated power from renewable energy resources to power grids. Current electric power grids are increasingly transforming into more inverter-dominant power grids as more renewables are connected to the power grids. However, cybersecurity concerns arise due to extensive information exchange among the networked smart inverters, which may also employ seamless firmware update functions. Therefore, the attack surface of the power grid has been significantly expanded by numerous smart inverters. It is predicted that more sophisticated threat actors will directly impact smart inverters with the intention of manipulating critical power infrastructures [1]. With the recent events of the Ukrainian blackouts and the Stuxnet worm, it expresses the immediate threat and importance of combating these attacks [2]. Additionally, failure to detect these malicious malware files within smart inverters could allow the spread of these attacks into the networked grid-level power systems. This leads to many negative issues with grid operation in terms of social impact, market impact, and large-scale blackouts. Even if the malware is found and patched, the once-infected system might cause challenges over time and become a significant threat to future power grids.

Ransomware is malware which encrypts important or credential data and theft of control of a system in demand of a ransom. Recently, ransomware attacks have targeted industrial control systems (ICS) and increased about 500% from 2018 to 2020 [3]. In 2021, the Colonial Pipeline suffered a ransomware attack that impacted computerized equipment managing the pipeline. The company provided 4.4 million dollars in Bitcoin for the decryption tool [4]. It is anticipated that more and more ransomware attackers who are financially motivated will target critical power system infrastructures such as substations, wind/solar farms, and charging infrastructures. This drastic increase alongside the further evolution of the ransomware strains threatens smart grid, with a multitude of negative impacts that include: 1) the leaking and selling of confidential data leading to more cyber-attacks in the future; 2) damage to field device/system control processes and consequently the creation of public safety hazards; 3) substantial disturbances to the power grid operation; and 4) harm in the form of economic loss. Thus, smart grid cybersecurity and engineering teams must prepare now to combat more frequent ransomware attacks with serious real-world consequences in the years ahead.

Existing industry defense strategies of the power grid have focused on tamper proof data backup and network-based security techniques such as network segmentation, encryption (e.g., transport layer security (TLS)), moving target defense, and network intrusion detection system (IDS) to reduce the vulnerabilities of utility and industry communications standards (e.g., SunSpec Modbus, DNP3, and SEP2). However, encrypted ransomware files in TLS network protocols can bypass the firewall security mechanisms and network IDS, and human risks always exist, which threaten admin access to smart grid devices and servers [5]. In general, ransomware detection methods are classified into two categories: static analysis and dynamic analysis methods. Static malware

analysis examines ransomware without executing the actual binary files. Simple static analysis methods utilize static data such as file header information, file hash, and URL. There are open-sources tools/servers providing static malware analysis such as VirusTotal [6]. Although conventional static malware analysis methods are simple to detect known ransomware and easy to use, they are largely ineffective against sophisticated ransomware attacks [7]. Recently, static malware detection methods using machine learning (ML) have been proposed to improve detection accuracy [8]. Since ransomware is evolving, new malware should be detected as well. Dynamic analysis methods detect ransomware attacks using abnormal behavioral data caused by the compiled ransomware or ransomware events by adversaries in the target system. The authors in [9] collected packets and data from network traffic between an infected computer and a command and control (C2) server. Using the network data, a random forest (RF) ML method detected ransomware with over 86% detection accuracy. In [10], a ML combining Navies Bayes (NB) and Support Vector Machine (SVM) is used to detect ransomware attacks by using network data from function virtualization (NFV) and software defined network (SDN). Compared to static malware analysis methods, dynamic methods might provide better capability of detecting sophisticated and unknown ransomware. However, a huge amount of network and event data are required. However, such ransomware detection methods have been studied less in power system applications. Compared to ransomware defense in servers, smart grid device-level defense is challenging due to the limited embedded system resources and lack of firmware ransomware data samples. Recently, malware injected firmware was detected by ML methods where hardware performance counters are used to train the ML [11]. However, defending advanced firmware malware attacks (e.g., MoonBounce [12] implanted to the SPI flash memory, which has not been detected by malware scanning products) must be studied.

This paper explores ransomware attacks targeting smart inverters. It proposes an ML-based memory forensics method to detect ransomware binaries within the memory of a commercial smart inverter. Available variants of ransomware and benign firmware binaries are collected based on the target SPI flash memory chip through reverse engineering. Binary manipulation methods [13] are adopted and used to generate variants. Then, a memory dump from the chip is pulled and decomposed to penetrate the smart inverter files to generate malicious firmware variants with ransomware encryptions. The original and compromised binary files are then extracted and examined under memory forensics. Lastly, the data is moved through a ML preprocessing component before being executed with the deep transfer learning (DTL) method [14]. The DTL model is used to retrain a convolutional neural network-based ransomware detection algorithm using the device firmware files. The experimental result shows that the proposed ML-based memory forensics method achieves above 99% of detection accuracy. The explored potential ransomware detection allows the availability to understand the workings of two different scenarios: 1) ransomware already implemented within a system and 2) non-running ransomware files that could be extracted and experimented on.

2. SEQUCITY THREAT MODELING FOR SMART INVERTERS

Security threat modeling identifies and enumerates a target system's threat points on its system specification. In this paper, a commercial smart inverter-tailored security threat modeling is designed based on hands-on hardware and software reverse engineering.

2.1 System Identification

System identification is the first step of a security threat modeling method to mainly identify: 1) cyber and physical components, 2) intelligence gathering, 3) data modeling and data flow maps, and 4) current security measures of the target system. The system targets a commercial smart inverter architecture [14] as illustrated in Figure 1. A smart inverter mainly consists of three layers 1) network and application layer (*L1*), 2) controller layer (*L2*), and 3) power electronics hardware layer (*L3*). Operators and DER site owners can remotely manage the smart inverters. In addition, device vendors can access their devices to update firmware for bug fixes. The network and process layer consists of a microprocessor unit (MPU) and SPI flash memory. An embedded Linux OS and a bootloader for embedded systems (e.g., U-Boot) are installed in the flash memory to operate the smart inverter system, allowing a bi-directional communication process through LAN ports or WLAN module. Conversely, the controller layer has a relatively lower computational resource with a microcontroller unit (MCU) for signal processing and power stage regulations. Regarding the firmware update, an appropriate firmware image is delivered into the SPI flash memory in the communication layer first, then loaded into the internal memory of the MCU in the controller layer. Therefore, a malicious firmware binary in the image can compromise an inverter function of the power electronics hardware layer. Advanced malware attacks have started compromising SPI flash memory, evading traditional malware detection engines [15]. Therefore, this memory is a key component to protecting target smart inverters against evolving malicious binary attacks.

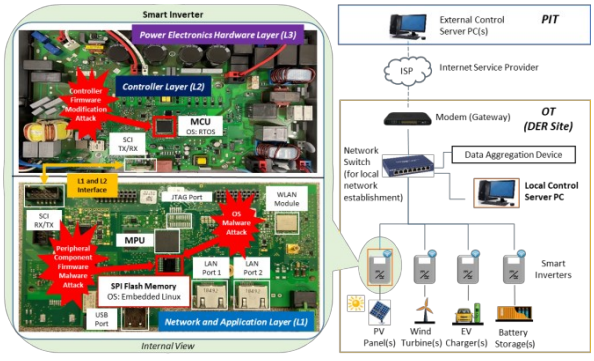

Figure 1: A commercial smart inverter consisting of three layers: Network and Application Layer (*L1*), Controller Layer (*L2*), and Power Electronics Hardware Layer (*L3*).

2.2 Attack Modeling

Attack modeling illustrates potential but practical attack methods from the adversary's point of view.

2.2.1 Physical Memory Access

A threat actor (e.g., apostatic insider) can access SPI flash memory and tamper with firmware images during a system development stage, a supply chain process, or a maintenance process. The memory can be physically accessed through 1) JTAG debugging pins, 2) UART debugging pins, and 3) flash memory chip pins. Even though debugging pin orders are obfuscated by the smart inverter vendor to prevent reverse engineering, several commercial penetration testing tools are available to scan appropriate pin orders. Once the proper pin orders are successfully scanned, the adversary can interrupt a normal boot process and command to open a backdoor, such as using a reverse shell-based Netcat function or unencrypted Telnet protocol. Then, additional malicious payloads can be remotely and seamlessly transferred through this backdoor.

Another third-party vendor designed the flash memory chip on the board of the smart inverter. Therefore, a potential vulnerability can also exist here. Furthermore, most flash memory chip vendors provide pin information on the internet. Thus, a memory access tool can connect to the proper pins to dump and flash a whole firmware image for a nefarious purpose.

2.2.2 Firmware Update

An imperfectly designed or malicious firmware image (by an interception during firmware update [16] or insider) can reactivate the Netcat function or the Telnet protocol, allowing unintended external remote access. If this firmware image includes precise checksum data created by its vendor, an integrity verification process provided by the bootloader will be automatically passed. As a result, the target system-tailored malware can be downloaded to compromise smart inverters.

2.3 Penetration Testing

Once the target binary image stored in SPI flash memory is dumped by a memory access tool and decompressed into system files (e.g., squashfs), the smart inverter could be penetrated by the following techniques:

1) *Firmware Image Tampering:* An executable backdoor file is inserted into the decompressed system files, then an administrative startup script is modified. Therefore, the loaded backdoor is ready to run automatically whenever the smart inverter system turns on.

2) *CRC-32 Checksum Collision*: During the boot process of the smart inverter system, the data integrity of the target image is verified by a cyclic redundancy check-32 (CRC-32) checksum algorithm after scanning the selected either Linux kernel OS image, firmware image, or both. By appending maliciously calculated four consecutive bytes [17] at the end byte offset of the corresponding image, the CRC-32 checksum is bypassed.

3) *Byte Order Obfuscation Bypass*: A binary section for the system files is padded by a reverse hexadecimal byte order. To calculate a proper CRC-32 of this section, this order is reverted. A well-known format that could be used as a byte order obfuscation is the big and little endian.

4) *Boot Header Information Tampering*: uImage in U-Boot includes header CRC data, timestamp of image creation, image size, and image CRC data. This information can be tampered with using a hex editor to match the compromised firmware image information.

3. TYPESET TEXT PROPOSED DEVICE-LEVEL MEMORY FORENSICS FOR SMART INVERTERS

3.1 Proposed Memory Forensics Methods

The proposed ML-based memory forensics method is described in Figure 2. After setting up a serial connection to proper pins between the target SPI flash memory chip and a memory interface device, a stored original binary image can be dumped. Then, this image file is transferred to a security module (i.e., Raspberry Pi (R-Pi) in Figure 2) for parsing of its structure. At this point, a blue team reverse engineer performs memory forensics based on parsing information in the PC. Then, multiple non-repetitive benign ML data samples can be created by non-harmful byte edits (i.e., timestamp).

Meanwhile, the original received image is decompressed into a filesystem consisting of multiple directories and system files on the R-Pi device. Then, a red team reverse engineer load and execute ransomware (e.g., Conti ransomware) in every directory, encrypting from a single file to multiple files to create non-repetitive malicious ML data samples. After then, each sample is compressed then each malicious image is recreated. As a result, both benign and malicious ransomware data samples are trained by the CNN model in the PC with higher computation resources. It is implemented using the proposed DTL model on a relatively lower computation-powered security module (e.g., R-Pi).

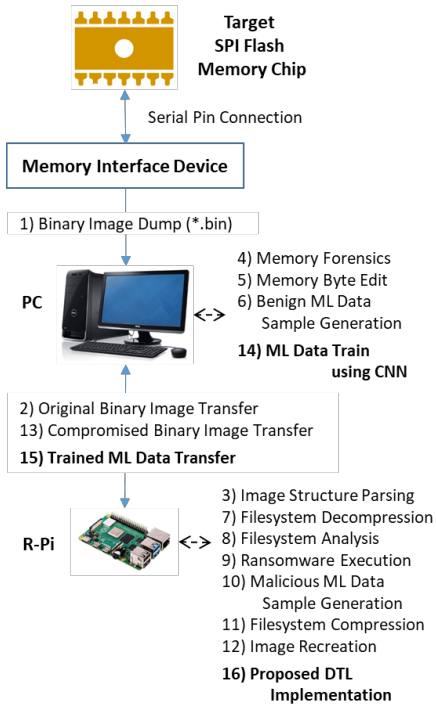

Figure 2: Proposed ML-based memory forensics method

3.2 Binary Files Collection and Binary Artifacts

Benign binary samples: After dumping the original firmware binary image from SPI flash memory, one hundred different benign binary samples are created by a binary manipulation method using a hex editor. To maintain benign binary characteristics, retouching sensitive and critical binary information is avoided. Instead, specific timestamp-related bytes are modified to generate samples with different image creation times.

Ransomware-encrypted binary samples: Using the original dumped firmware binary image, the filesystem binary section (e.g., squashfs) is carved out. After decompressing it into multiple system files, a real-world version of Conti ransomware is executed to encrypt every different file or directory. Then each encrypted outcome is compressed again into the filesystem binary. As a result, non-repetitive one hundred samples are created.

3.3 Deep Transfer Learning

A DTL algorithm is applied to create the smart inverter-centric ransomware detection algorithm by retraining a basis CNN-based ransomware detection model [14] with the generated device-specific samples. The following section proposes a DTL-based ransomware detection method, specifically improving the detection accuracy for device-centric ransomware detection. The DTL utilizes a deep learning model designed for ransomware detection and classification as a basis model. Typically, practitioners would take a pre-trained model from a type of image dataset, freeze a portion of the layers, and fine-tune the last few layers on the newly obtained dataset [14]. The advantages of transfer learning comprise of accelerated training, reduced computational resources, and smaller code strings. The search for deep neural networks and maintaining high classification performance, especially on the relatively smaller-sized dataset, is continued as researchers adopt multiple pre-trained or trainable ML models. In this paper, a CNN model, the VGG 16, was selected as a foundation, which is trained by publicly available ransomware binary files upon being converted into image arrays. Figure 3 illustrates the proposed DTL process, which demonstrates the full workflow of the DTL model.

3.3.1 DTL Experimental Setup

The structure of the deep learning model (DL) is developed and trained on a native computing platform, Microsoft Visual Studio, using the datasets described in Section III. Firstly, the DTL model experiment was executed using a Macintosh Operating System with the Apple M1 SOC, 7-Core GPU, and 8 GB of RAM. A total of 100 benign firmware files (Goodware) and 100 ransomware-encrypted files (Ransomware) are used for training (60%) and validation (40%). The model runs for more than two minutes to process the three main algorithms: Binary to Grayscale, Training & Validation, and Prediction. Then, the proposed DTL model is implemented in an IoT device (R-Pi with 64-bit ARM OS, internal GPU (896 MB), and 8GB RAM) using the test image files, which hold randomized ransomware and goodware images. Finally, the result is compared with VirusTotal and a VGG 16 model (i.e., a DL basis model in this paper) trained by publicly available ransomware

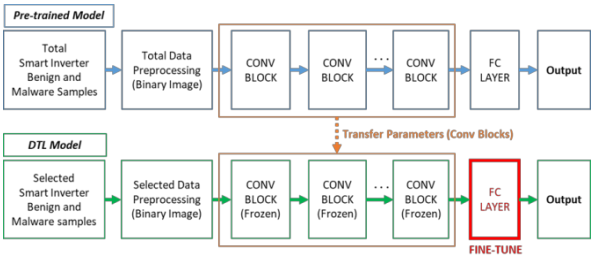

Figure 3. Deep transfer learning [14]

files. The benign and malicious binary files are represented as grayscale images and are used to train the last FC layer in the VGG16 model. Furthermore, other faster implementations can be realized using purely ML-based workstations, which are GPU-assisted and compute with added accuracy.

3.3.2 Data Preprocessing for DTL

The benign and ransomware-encrypted binary files were labeled "Goodware" and "Ransomware" respectively, as they were passed through a conversion algorithm to extract binary data and convert them into image arrays. The RGB image array is then converted to training data. Since the transfer learning model was trained over grayscale training images, converted using the above-mentioned method, the RGB images were imported as grayscale images to better initiate the model. The static binary file was mapped to an array of integers between 0 and 255. Hence, each binary is converted into a one-dimensional array [0; 255]. Then the array is normalized to [0, 1] by dividing by 255. The normalized array is reshaped into a two-dimensional array. The height of the file is the total length of the one-dimensional array divided by the width. Interested readers may refer to [18] for more details of the data preprocessing. Figure 4 shows the transition of data as it is being processed to be fit into the CNN-based DTL. The initial model was trained using the same *.png formatted files except for the file labels. Different file labeling formats were used to maintain proficiency. For randomized testing, we propose the use of a total of 100 .bin files to be converted to images and then fit into the final saved model to be trained in any local system or the cloud. The randomized dataset was set in a different directory and the

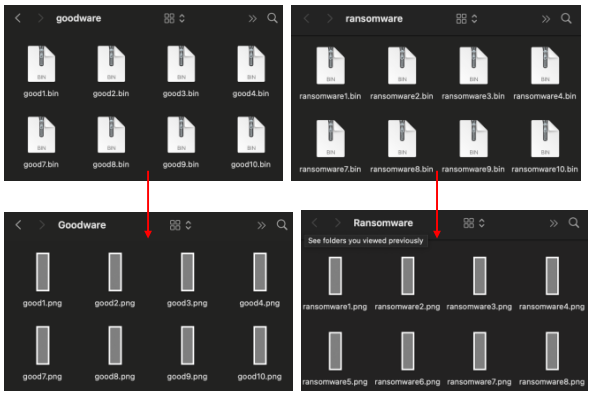

Figure 4: Binary to image conversion

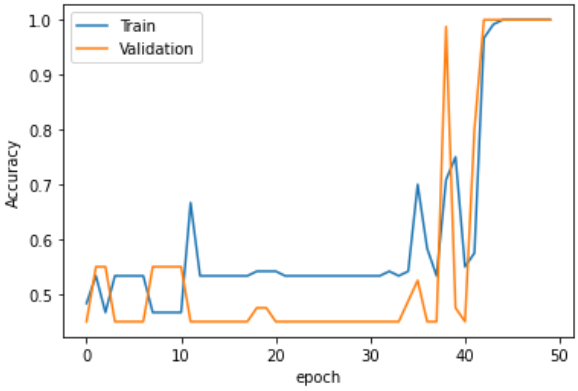

Figure 5: Model training and validation results

files undergo full algorithmic transformation to maintain validity.

3.3.3 DTL Model Training and Validation

Training and validation accuracy values converge to 99% and 98%, respectively. However, the VGG16 basis model shows 91.31% of ransomware detection accuracy for the device-specified ransomware samples, although the basis ransomware detection algorithm provides high detection accuracy for the general ransomware files (> 96%). The proposed DTL method uses three convolutional 2D layers and three maxpooling 2D layers fitted over 640 parameters. The model comprises a total of 132,097 trainable parameters. The model executes and predicts the ransomware and goodware in 1 minute and 44.5 seconds while being run on the Apple M1 architecture. The model was trained for 50 epochs, where it converged to over 98% accuracy in the last 10 epochs. as shown in Figure 5. Once the learned weights of the model are saved and trained to predict, separate randomized ransomware and goodware files are created, containing randomized files for effective prediction. As the accuracy was almost nearing 99%, it predicted all the files accurately as mentioned in the file name.

4. EXPERIMENTAL VALIDATION

4.1 Experimental Setup

To validate the proposed method, an ML-based memory forensics testbed has been constructed, as shown in Figure 6. During the firmware binary image dumping process, an interface cable between the network and application layer (*L1*) and the controller layer (*L2*) has been disconnected to prevent potential physical damage.

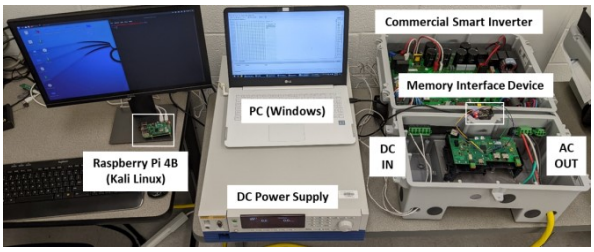

Figure 6: Experimental setup

4.2 Ransomware Attack

Compromising firmware image is performed using a hex editor. For a clear firmware modification, the filesystem (sqaushfs) section of both images is carved out first. The timestamp (firmware image creation date and time) and the image size are compared in Figure 7. The implementation of the reverse hexbyte order method has been applied for data obfuscation by a vendor developer. The implementation of this method can be seen in Figure 7(a) by the byte order of '5C D3 DE 5C' is reversely converted into '5C DE D3 DC'. These hex bytes are converted into a human-readable date, resulting in 'Friday, May 17, 2019, 03:29:32.000 PM (GMT)'. Following the same procedure, '63 28 B6 84' is converted into 'Monday, Sep 19, 2022, 06:35:48.000 PM (GMT)' in Figure 8(b). When ransomware successfully encrypts one or more targeted directories, a text file (e.g., readme.txt) is automatically created and presented to the user. Within the readme.txt file, there are specific instructions on how to pay the malicious user via a Dark web link for future decryption. Therefore, depending on the number of encrypted directories, the same number of test files will be created, increasing the total image size. To validate this case, the image size is calculated and then compared between the two filesystems. Figures 8(a) and 8(b) show that the compromised filesystem (11583708 [bytes] ='00 B0 C0 DC') was a little bit larger than the original filesystem (11580336 [bytes] = '00 B0 B3 B0').

An encryption of a CONTI strain can be seen in Figure 8 (c). In these forensics, a whole compromised firmware image (i.e., ransomware100.bin) instead of carving out the filesystem to clearly demonstrate the current method can detect a ransomware attack indication.

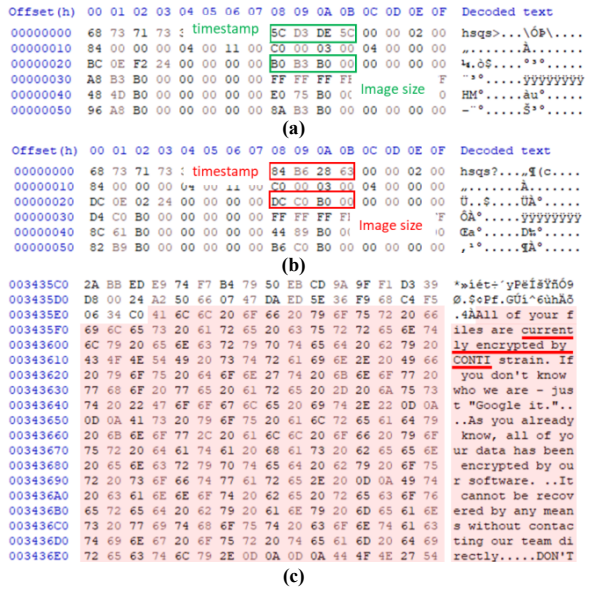

Figure 7: Ransomware Attack Results: (a) original firmware data from the SPI flash memory, (b) compromised firmware data by ransomware, and (c) ransomware attack indication on the compromised firmware

```
In [12]: prediction = model.predict(X, batch_size=1)
         print(prediction,'\n')  # will be a list in a list.

         for x in prediction:
             print(class_category[int(x)])

[[1.]
 [1.]
 [1.]
 [0.]
 [1.]]

Ransomware
Ransomware
Ransomware
Goodware
Ransomware
```

Figure 8: Experimental results of ransomware detection

4.3 Ransomware Impacts on a Smart Inverter

Due to ransomware encryption on the filesystem in memory, an operator could not remotely access the smart inverter for monitoring and maintenance. However, the controller layer (*L2*) and the power electronics hardware layer (*L3*) have not been impacted yet with proper DC input and AC output values.

4.4 Ransomware Attack Detection Results

The trained model is installed in an R-Pi (i.e., security module in a smart inverter). Variants of 4 ransomware and 1 goodware files are additionally created to validate a real-time ransomware attack detection using the proposed method. Figure 8 shows the experimental result of the real-time ransomware attack detection. The detection accuracy was almost nearing 100%, it predicted all the files accurately as mentioned in the file name.

5. CONCLUSION

This paper provides ransomware threat modeling for smart inverters and proposes an ML-based memory forensics method. The proposed ML-implemented memory forensics method can successfully detect malicious ransomware attacks targeting the smart inverter with high accuracy. This method can be implemented in the smart inverter with additional security processor (i.e., security module) enabling device-centric ransomware/malware defense. Future works include: 1) proactive detection using prediction of ransomware/malware variants and deobfuscation techniques, 2) ransomware key detection and recovery, 3) developing recovery techniques for the smart inverters, and 4) cyber-resilient control for mitigating impacts of powerbots implementing automatic detection for the files.

ACKNOWLEDGMENT

This work was supported in part by the Department of Energy (DoE) under award No. DE-EE0009026, National Science Foundation (NSF) under award No. EEC-1359414 and CNS-2219733.

REFERENCES

[1] J. Qi, A. Hahn, X. Lu, J. Wang, and C-C. Liu, "Cybersecurity for distributed energy resource and smart inverters," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 1, no. 1, pp. 28–39, Dec. 2016.

[2] Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), "Cyber-Attack Against Ukrainian Critical Infrastructure", Incident Report, February 25, 2016.

[3] S. Larson and C. Singleton, "Ransomware in ICS environments." [Online]. Available: https: //www.dragos.com/resource/ransomware-in-ics-environments/

[4] M. Novinson, "Colonial pipeline hacked via inactive account without mfa." [Online]. Available: https://www.crn.com/news/security/colonial-pipeline-hacked-via-inactive-account-without-mfa

[5] J. Ye *et al.*, "A review of cyber-physical security for photovoltaic systems," *IEEE Journal of Emerging and Selective Topics in Power Electronics*, vol. 10, no. 4, pp. 4879-4901, Aug. 2022.

[6] VirusTotal.com, [Online]. Available: https://www.virustotal.com/gui/home/upload

[7] A. Kharraz *et al.*, "Cutting the gordian knot: A look under the hood of ransomware attacks," in *Proc. Int. Conf. Detection of Intrusions and Malware and Vulnerability Assessment* Springer, 2015, pp. 3–24.

[8] M. Khammas, "Ransomware detection using random forest technique," *ICT Express*, vol. 6, no. 4, pp. 325–331, 2020.

[9] O. M. G. Cusack and E. Keller, "Machine learning-based detection of ransomware using sdn," in *Proc. the 2018 ACM International Workshop on Security in Software Defined Network Network Funciton Virtualization, IEEE*, 2018, pp. 1–6.

[10] F. Maimo *et al.*, "Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments," *Sensors*, vol. 19, p. 1114, 2019.

[11] A. P. Kuruvila S. Kundo, and K. Basu, "Defending hardware-based malware detectors against adversarial attacks," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 9, pp. 1727–1739, 2021.

[12] D. L. M. Lechtic, V. Berdnikov and I. Borisov, "Moonbounce: the dark side of uefi firmware," [Online]. Available: https://securelist.com/moonbounce-the-dark-side-of-uefi-firmware/105468/

[13] A. Abusnaina, A. Anwar, S. Alshamrani, A. Alabduljabbar, R. Jang, D. Nyang, and D. Mohaisen, "Systemically evaluating the robustness of ML-based IoT malware detectors," in *Proc. 2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, Taipei, Taiwan, June 21- 24, 2021, pp. 3-4.

[14] S. Alvee, B. Ahn, S. Ahmad, K. Kim, T. Kim, and J. Zeng, "Device-centric firmware malware detection for smart inverters using deep transfer learning," in *Proc. 2022*

IEEE Design Methodologies for Power Electronics, Bath U.K., Sep. 1-2, 2022, pp. 1-5.

[15] [Online]. Available: https://eclypsium.com/2022/06/02/conti-targets-critical-firmware/

[16] S. Falas, C. Konstantinou and M. K. Michael, "A Hardware-based Framework for Secure Firmware Updates on Embedded Systems," in *Proc. 2019 IFIP/IEEE 27th International Conference on Very Large-Scale Integration (VLSI-SoC)*, 2019, pp. 198-203.

[17] [Online]. Available: https://www.nayuki.io/page/forcing-a-files-crc-to-any-value

[18] S. Alvee, B. Ahn, T. Kim, Y. Su, Y-W. Yoon, and M-H. Ryu, "Ransomware attack modeling and artificial intelligence-based ransomware detection for digital substations," in *Proc. 2021 6th IEEE Workshop on Electronic Grid (eGrid)*, New Orleans, LA, Nov. 8-10, 2021, pp.1-5.