2014 IEEE 8th International Symposium on Service Oriented System Engineering

Identifying Malicious Metering Data in Advanced Metering Infrastructure

Euijin Choo*, Younghee Park†, Huzefa Siyamwala†

*North Carolina State University †San Jose State University
*echoo@ncsu.edu, †{younghee.park,huzefa.siyamwala}@sjsu.edu

Abstract

Advanced Metering Infrastructure (AMI) has evolved to measure and control energy usage in communicating through metering devices. However, the development of the AMI network brings with it security issues, including the increasingly serious risk of malware in the new emerging network. Malware is often embedded in the data payloads of legitimate metering data. It is difficult to detect malware in metering devices, which are resource constrained embedded systems, during time-critical communications. This paper describes a method in order to distinguish malware-bearing traffic and legitimate metering data using a disassembler and statistical analysis. Based on the discovered unique characteristic of each data type, the proposed method detects malicious metering data. (i.e. malware-bearing data). The analysis of data payloads is statistically performed while investigating a distribution of instructions in traffic by using a disassembler. Doing so demonstrates that the distribution of instructions in metering data is significantly different from that in malware-bearing data. The proposed approach successfully identifies the two different types of data with complete accuracy, with 0% false positives and 0% false negatives.

***Index Terms*—Advanced Metering Infrastructure, security, malware, smart meters, disassembler, ARM instructions**

I. Introduction

Today, the Advanced Metering Infrastructure (AMI) manages energy consumption and demand response through metering devices (i.e. called smart meters). The AMI provides bi-directional communication services for real-time control and management. However, it also raises a lot of security issues, including privacy invasion and vulnerability to network-borne attacks [16]. To overcome recent problems, research has focused on security solutions in the AMI [8], [16], [15], [9], [18], [22], [13].

In 2010, the Stuxnet worm[4] attacked SCADA(Supervisory Control And Data Acquisition) systems and PLCs(Programmable Logic Controllers) in industrial systems [18], [22]. A McAfee report warned of the ease with which attackers could exploit smart meters and take control of the whole system [15]. In 2009, Mike Davis of Blackhat showed how quickly the smart meters could be compromised by malware and how quickly malware could propagate worms in the meters [9]. In the literature, many defense methods against malware have been proposed[3], [21], [25]. However, existing techniques to defend against malware cannot be directly applied to the AMI because of its unique system specifications, its different network protocols and environments, and resource-constrained metering devices.

In this paper, we first identify the characteristics of metering data and malware in terms of the distribution of instructions when a disassembler is used. Malware is a form of binary executables attached to data payload that pretend to be metering data; however, metering data does not usually include binary executables. Based on this fact, we derive the characteristics of metering data and malware-bearing data through disassembly. The distributions of instructions are statistically analyzed to generate a unique pattern for each data type. This unique pattern is used to identify metering data in the traffic that contains malware.

Malware writers often obfuscate data and hide file structures to avoid detection [1], [19], [25], [12]. Furthermore, metering data often does not have any structure, thus no separated sections are available and the data looks like random sequences of binary values. Disassembling unstructured data is challenging. Therefore, we perform statistical analysis of all of the AMI traffic data instead of

The corresponding author is Younghee Park, an assistant professor in Computer Engineering. The first two authors contributed equally to this work.

978-1-4799-3616-8/14 $31.00 © 2014 IEEE
DOI 10.1109/SOSE.2014.75

focusing on a specific portion of the data. As a result of our disassembly and statistical analysis, we were able to validate that metering data and malware have significantly different instruction distributions. Based on the unique distribution of each data type, the proposed method clusters unique patterns that represent each data type. The clustered patterns will be used to identify unknown data to detect malicious metering data.

Our main contributions are summarized as follows. First, we propose a methodology to distinguish metering data and malware in the AMI based on the distribution of instructions. We extract instruction distributions from metering data and malware-bearing data to show their distinct differences. Second, we propose a method to detect malicious metering data by using unique patterns clustered from the instruction distributions. Lastly, our proposed method is validated with real metering data collected in the AMI and real binaries in the embedded systems. The preliminary results indicate that our method will be feasible for future use in the AMI.

The remaining parts of this paper are organized as follows. In Section II, we review related work. Section III proposes our method to detect malicious metering data. In Section IV, we validate our method with our research results. Finally, we conclude in Section V with suggestions for future research.

II. Related Work

The AMI provides real-time two-way communications between utility companies and end customers with smart meters. In addition to the smart meters, it consists of smart meters, data concentrated units(DCU), and a Head-end system with other management systems. It aims to support various services, such as automatic meter reading, automated energy distribution, demand response and load management, through interaction between the utility and the customer. The DCU or the Headend system acting as a client aggregates metering data by sending control commands to the meters. The smart meters acting as a server generate responses to the commands to transfer metering data and other information into the DCU or the Headend. There are a lot of communication protocols used for those systems including wireless and wired networks, and unique proprietary protocols are still under investigation.

In spite of the benefits of the AMI, it causes a lot of security issues. Since the smart meters do not usually have security mechanisms [24], compromised smart meters present the most serious potential damage to the AMI [16], [5]. Even though the meters have a security protocol, they are always vulnerable to malware during communications [16], [5].

Attackers often hide malware into the incoming/outgoing traffic in a system while masquerading as benign data. In the alternative, attackers may directly attack vulnerable open services. Previous malware detection methods fall into two categories: static analysis and dynamic analysis. Static analysis utilizes a disassembler to detect malware without its being executed. Through disassembly, these approaches derive specific signatures. The signatures can be extracted by pattern matching[7], instruction flow[17], and data flow[6]. On the other hand, dynamic analysis monitors execution behavior of malware in a virtualized environment [14], [10], [2]. It is impossible to create the virtualized and isolated environment in embedded systems like smart meters. Wei-Jen *et al.*[25] propose a statistical analysis method to detect files with hidden malicious code by using n-gram analysis. Even though the proposed method quickly catches a certain evidence of infected file, it has limitations in that they only analyze a certain portion of a file. However, it is necessary to capture all behavior [23] because malware often changes behavior during execution.

III. The Proposed Approach

This section describes the proposed approach to block malicious metering data in AMI. In subsection III-A, we first give a brief overview of the proposed approach. The details of the proposed approach are followed in subsections III-B and III-C.

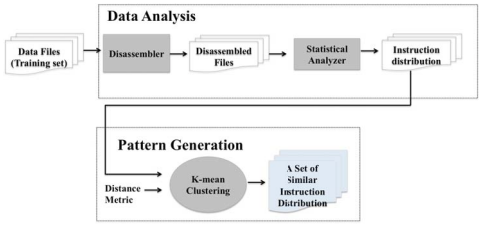
A. Overview

The goal of the proposed approach is to filter unexpected data out of the AMI. While doing so, we perform statistical analysis on the data payload before it is accepted into applications. Fig.1 presents a system architecture of the proposed approach. As shown in Fig.1, there are two functional components in the proposed approach. One is to generate unique patterns for each data type and the other one is to classify unknown data based on generated patterns and to detect malicious metering data. Note that the unique pattern is not for a particular family of malware for which obfuscated malware is easily evaded. Instead, we generate a unique pattern representing all families of same type.

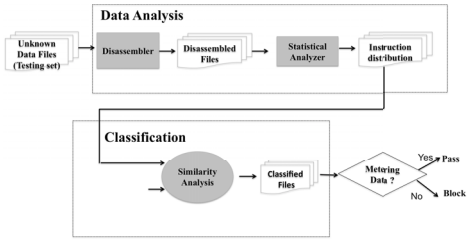
In subsection III-B, we first describe a method to characterize the unique distribution of instructions for different types of data. The details of the proposed method to detect malicious metering data will follow in subsection III-C.

B. A Distribution of Instructions

In order to extract a distribution of instructions, we first need to disassemble data: metering data and malware (i.e.



(a) Generating patterns with k-mean clustering



(b) Detecting malicious metering data based on the clustered patterns

Fig. 1. System Architecture

Mnemonic	Description	Mnemonic	Description
AND	AND	**ADD**	Add
ADC	Add with carry	**B**	Branch
BIC	Bit Clear	**BL**	Branch with Link
BX	Branch and Exchange	**CDP**	Coprocessor Data Processing
CMN	Compare Negative	**CMP**	Compare
EOR	Exclusive OR	**LDC**	Load coprocessor from memory
LDM	Load multiple registers	**LDR**	Load register from memory
MCR	Move CPU register to coprocessor register	**MLA**	Multiply Accumulate
MOV	Move register or constant	**MRC**	Move from coprocessor register to CPU register
MRS	Move PSR status/flags to register	**MSR**	Move register to PSR status/flags
MUL	Multiply	**MVN**	Move negative register
ORR	OR	**RSB**	Reverse Subtract
RSC	Reverse Subtract with carry	**SBC**	Subtract with carry
STC	Store coprocessor register to memory	**STM**	Store Multiple
STR	Store register to memory	**SUB**	Subtract
SWI	Software Interrupt	**SWP**	Swap register with memory
TEQ	Test bitwise equality	**TST**	Test bits

TABLE I. ARM instruction set

malware-bearing data). By using disassembler [11], we obtain a distribution of instructions for each of the two types of data. The distribution indicates the frequency of an instruction within a given file. Table I shows a list of 34 basic ARM instructions provided by the information center[1].

Fig.2(b) and Fig.2(a) represent distributions of instructions for 255 pieces of metering data and 255 pieces of malware data, respectively. The X-axis represents each instruction, while the Y-axis represents the percentage of each instruction over the total number of instructions in the data. Also, Fig.3 represents the average distribution of instructions for the two different types of data. The X-axis represents each instruction, while the Y-axis represents the

[1]http://infocenter.arm.com

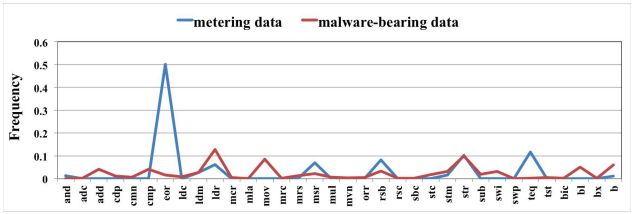


Fig. 3. Average distributions of instructions

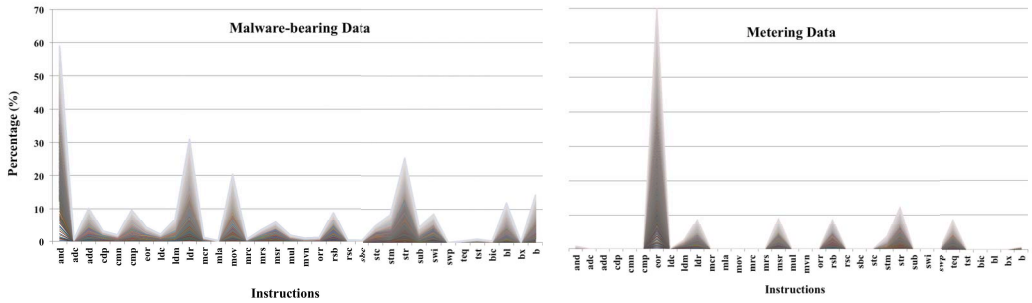
ratio of each instruction compared to the total number of instructions in the data.

As illustrated in Fig.3 and Fig.2, metering data and malware-bearing data have significantly different distributions of instructions. First of all, most instructions are evenly used when data include malware (i.e. binary executables), while a set of only a few instructions are shown in legitimate metering data. Specifically, only a few instructions including eor, ldr, msr, rsb, str, and teq are used in metering data. However, malware-bearing data generally show a lot of control flow instructions. In other words, malware-bearing data have a lot of *b* (jump) and *bl* (call) instructions to change the execution flow in the executables. However, such control flow instructions are never shown in disassembled metering data. Instead, as shown in Fig.2(b), most instructions only relate to data processing and data transfer instructions. In addition, the range of used instructions is dedicated to a specific few instructions, rather than being evenly distributed. These observations are clear and invariant since legitimate metering data do not display the executable characteristics needed to shift control flow.

An interesting finding from the results of disassembly is that we can identify a data type just from the distribution of instructions. This is based on the observation that same data type shows a similar distribution of instructions and similar statistical characteristics, as shown in Fig.2. Furthermore, distributions of the two different types of data represents significantly different statistical characteristics. The following subsection III-C describes a technique to detect malicious metering data (i.e. malware-bearing data) by utilizing these unique instruction distributions.

C. Detecting Malicious metering Data

1) Pattern Generation: K-mean clustering is a clustering method that partitions observations into k clusters with a distance measure. In k-mean clustering, each observation is assigned to the cluster with the closest centroid. As discussed in subsection III-B, the distribution of instructions that results from disassembly of the two types of data are distinguishable. This means that the variation among



(a) Distributions of binary executables

(b) Distributions of metering data

Fig. 2. Distributions of instructions for the two different data types

k: The number of clusters

K-meanCluster()

1: Randomly pick k data from a training set as the initial k centroids
2: **repeat**
3: **for each** remaining data f in the training set in b **do**
4: Compute distances between f's distribution and k centroids' distributions
5: Assign f to the cluster whose centroid is the closest to f
6: Update the centroid
7: **end for**
8: **until** Centroids stabilize

Fig. 4. K-mean clustering for generating patterns

distributions for the same type of data is likely to be very small. The same type of data will thus be assigned into the same cluster, the centroid of which will be utilized as the signature for each data type to generate a unique pattern. Alternatively, data can be partitioned into a sub-category (i.e. subdata-type), depending on the application or other behavioral factors; however, we build multiple sub-models with k-mean clustering, so that each type can be partitioned into k clusters. Fig.4 describes the k-mean clustering algorithm used in the proposed approach.

In order to compute a distance, we employ Euclidean distance, the definition of which is as follows:

$$\sqrt{\sum_{i=1}^{n}(f_i - c_i)^2}, \tag{1}$$

where f_i is the ith element in the distribution vector of unknown data f and c_i is the ith element in the distribution vector of each centroid.

The result of Fig.4 is a set of k models, each having its centroid distribution as the pattern for the model. The generated unique pattern will be used as a signature to distinguish metering data and malware-bearing data.

2) Classification: In order to detect malicious metering data (i.e. malware-bearing data) with the given patterns, we compute a similarity score between an unknown datas distribution and each centroids distribution. Note that we are interested in the relationship between whole distributions, not in any specific instruction. Therefore, we employ Pearsons correlation coefficient that takes the correlation between data into consideration.

The definition of Pearson's correlation coefficient is as follows:

$$\frac{\sum_{i=1}^{n}(f_i - \overline{f})(c_i - \overline{c})}{\sqrt{\sum_{i=1}^{n}(f_i - \overline{f})^2}\sqrt{\sum_{i=1}^{n}(c_i - \overline{c})^2}}, \tag{2}$$

where f_i is the ith element in the distribution vector of unknown data f, c_i is the ith element in the distribution vector of each centroid, $\overline{f}$ is $\sum_{i=1}^{n} f_i/n$, and $\overline{c}$ is $\sum_{i=1}^{n} c_i/n$.

The coefficient value ranges from -1 to 1, where -1 indicates the complete opposite and +1 indicates the exact same value.

Given unique patterns and unknown data f, we compute Pearson's correlation coefficent between f's distribution and each pattern. The data f will be classified into a model, where the correlation score between the centroid

k(Number of clusters)	Metering(training)-Metering(testing)	Malware(training)-Metering(testing)
k=1	**0.994740821**	**0.037243701**
k=2	**0.994740821**	**0.070434112**
	-0.197042512	0.037243701
k=3	**0.994740821**	**0.037243701**
	-0.05388775	-0.096403763
	0.090942814	-0.101779978
k=4	**0.994740821**	0.037243701
	0.129430554	-0.127191319
	0.112773705	**0.365893756**
	-0.127939407	0.014168807

TABLE II. Similarity scores with a meter testing set when applying different k for clustering

k(Number of clusters)	Metering(training)-Malware(testing)	Malware(training)-Malware(testing)
k=1	**0.038593981**	**0.749101155**
k=2	**0.038593981**	0.030832381
	-0.091239562	**0.749101155**
k=3	**0.038593981**	**0.749101155**
	-0.265149142	-0.24249148
	-0.063116444	0.08251172
k=4	0.038593981	-0.045049109
	0.09276004	0.274175541
	-0.003548511	**0.749101155**
	0.171073048	0.085521288

TABLE III. Similarity scores with a executable testing set when applying different k for clustering

distribution and f distribution is the highest, i.e. the closest to 1.

IV. Experimental Results and Analysis

This section shows experimental results with analysis. Note that both metering data and malware can have different types of data so that sub-types can be determined by k-mean clustering. There are two types of data: metering data and malware(i.e. executables). Metering data is collected in the TCIPG testbed [2] as described in [20]. The executables are extracted from an embedded board based on an ARM processor. When traffic contains malware instead of metering data, it must include a form of an executable to produce malicious behavior. A type of executable is a predictable and unchanged feature of malware. Therefore, we redefine the executables as malware-bearing data, simply put, as malware.

We conducted experiments in two different settings. In the first setting, we used 1,428 randomly selected pieces of metering data for the training set, and 357 pieces of metering data and 357 pairs of binary malware for testing sets. In the second setting, we used 285 randomly selected binary executables for the training set, and 72 pieces of metering data and 72 binary executables for the testing sets.

Table II presents the averages of the similarity scores obtained with a meter testing set when applying different k for clustering from two training sets: metering data and malware. As described in Table II, the highest similarity score between distributions of meter testing data and distributions of meter training sets' centroids is 0.994740821, regardless of k. In addition, the highest similarity score between distributions of meter testing data and distributions of executable training sets' centroids is 0.037243701 for k=1,3, 0.070434112 for k=2, and 0.365893756 for k=4. Since 0.994740821 is bigger than 0.037243701, 0.070434112, or 0.365893756, metering data is always correctly classified into metering data using the proposed approach, regardless of k, which means there are 0% false negatives. Note that in our experiments, both metering data and malware included only one sub-type of data, respectively. Consequently, there is one cluster for each type of data that includes most of the data in the training sets and the other clusters include a few outliers. Accordingly, the similarity score will be very low.

On the other hand, Table II presents averages of similarity scores with an executable testing set when applying different k for clustering from two training sets: metering data and malware.

As described in Table III, the highest similarity score between distributions of executable testing data and distributions of meter training set centroids is 0.038593981 for k=1,2,3 and 0.171073048 for k=4. In addition, the highest similarity score between distributions of meter testing data and distributions of executable training set centroids is 0.749101155, regardless of k. Since 0.749101155 is bigger than 0.038593981, executables are always correctly classified as executables rather than as metering data using the proposed approach, regardless of k, which means there is a rate of 0In the same context, the similarity between metering data and malware will be very low as demonstrated in Table III.

	Itron (smart meter)	Raspberry Pi	Linux System
Processor	ARM-M3	ARM1176JZ-F	Intel Core i3
RAM	256Kb	512Mb SDRAM	3GB
Flash	512KB	4GB	30GB
Cache	No cache	32k-64k	1GB

Fig. 5. Different System Specifications

Compared to the general desktop, a disassembly is a heavy operation to perform in a smart meter because the embedded system has limited resources. Table 5 shows a comparison of various system specifications: the system specification of smart meter is provided by Itron company; the Raspberry PI is a popular embedded system; and the last system is a general desktop on Linux. Fig. 6 shows the

[2]http://tcipg.org

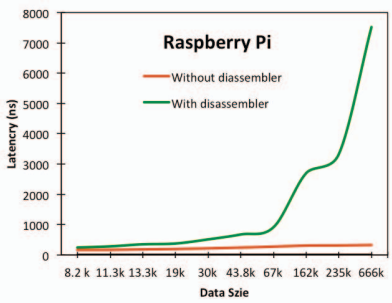


Fig. 6. Performance overhead on Raspberry Pi.

results of performance overhead when the disassembler is used in these different systems. The overhead is generated by the round-trip time when requests and responses are transmitted during the communication of a server and a client. The average round-trip time without the disassembler was around $160ns$, and with the disassembler was around $208ns$. This shows the small overhead when the disassembler is used. However, as the file size to be disassembled increases, the performance overhead also increases.

V. Discussion and Future Work

A smart meter is an attractive target for malicious purposes of attackers. In order to quickly and automatically detect and prevent malware propagation in the AMI, we proposed a methodology to differentiate metering data and malware-bearing data in this paper. In the AMI, malware often takes the form of binary executables, attached to data payload while pretending to be metering data, but metering data does not usually include binary executables. Therefore, the proposed approach focused mainly on the differentiation of legitimate metering data and binary executables. To do so, we performed statistical analysis to derive the unique statistical characteristics of metering data versus malware. The distributions of instructions in each type of data are significantly different from each other. Based on this fact, the proposed approach generated a unique pattern for each type of data in terms of instruction distribution.

Our research on malware detection in the AMI will continue in several future directions. Even though we only used datasets of metering data and binary executables without differentiating sub-types, we will consider various different subtypes of data in the future.

References

[1] M. Baig, P. Zavarsky, R. Ruhl, and D. Lindskog. The study of evasion of packed pe from static detection. In *Internet Security (WorldCIS), 2012 World Congress on*, pages 99–104. IEEE, 2012.

[2] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda. Scalable, behavior-based malware clustering. In *NDSS*, volume 9, pages 8–11. Citeseer, 2009.

[3] R. Berthier and W. H. Sanders. Specification-based intrusion detection for advanced metering infrastructures. In *Dependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on*, pages 184–193. IEEE, 2011.

[4] E. Byres, A. Ginter, and J. Langill. How stuxnet spreads–a study of infection paths in best practice systems. *Tofino Security, White paper*, 2011.

[5] M. Carpenter, T. Goodspeed, B. Singletary, E. Skoudis, and J. Wright. Advanced metering infrastructure attack methodology. *InGuardians white paper*, 2009.

[6] D. M. Chess and S. R. White. An undetectable computer virus. In *Proceedings of Virus Bulletin Conference*, volume 5, 2000.

[7] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In *Proceedings of the 12th conference on USENIX Security Symposium - Volume 12*, SSYM'03, pages 12–12. USENIX Association, 2003.

[8] A. Cui, M. Costello, and S. J. Stolfo. When firmware modifications attack: A case study of embedded exploitation.

[9] M. Davis. Smartgrid device security: Adventures in a new medium. *Black Hat USA*, 2009.

[10] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. X. Song. Dynamic spyware analysis. In *Usenix Annual Technical Conference*, pages 233–246, 2007.

[11] C. Giesselink. a disassembler for "arm instruction set".

[12] G. Jeong, E. Choo, J. Lee, M. Bat-Erdene, and H. Lee. Generic unpacking using entropy analysis. In *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, pages 98–105. IEEE, 2010.

[13] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke. Smartgrid security issues. *Security & Privacy, IEEE*, 8(1):81–85, 2010.

[14] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. Kemmerer. Behavior-based spyware detection. In *Proceedings of the 15th USENIX Security Symposium*, pages 19–19, 2006.

[15] McAfee. Smarter protection for the smart grid. *Technical Report*.

[16] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *Security & Privacy, IEEE*, 7(3):75–77, 2009.

[17] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pages 231–245. IEEE, 2007.

[18] D. Nicol. Hacking the lights out. *Scientific American Magazine*, 305(1):70–75, 2011.

[19] P. O'Kane, S. Sezer, and K. McLaughlin. Obfuscation: The hidden malware. *Security & Privacy, IEEE*, 9(5):41–47, 2011.

[20] Y. Park, D. M. Nicol, and et. al. Prevention of malware propagation in ami. In *Proceedings of the IEEE International Conference on Smart Grid Communications*, SmartGridComm, 2013.

[21] V. Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.

[22] J. Pollet. Electricity for free? the dirty underbelly of scada and smart meters. *Proceedings of Black Hat USA*, 2010, 2010.

[23] S. J. Prowell, M. Pleszkoch, K. D. Sayre, and R. Linger. Automated vulnerability detection for compiled smart grid software. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–5. IEEE, 2012.

[24] I. Rouf, H. Mustafa, M. Xu, W. Xu, R. Miller, and M. Gruteser. Neighborhood watch: security and privacy analysis of automatic meter reading systems. In *Proceedings of the ACM conference on Computer and communications security(CCS)*, Raleigh, NC USA, 2012.

[25] S. J. Stolfo, K. Wang, and W.-J. Li. Fileprint analysis for malware detection. *ACM CCS WORM*, 2005.