On the applicability of binary classification to detect memory access attacks in IoT

Sanaa Kerroumi, Damien Couroussé, Florian Pebay-Peyroula, Mohammed Ait Benaoud, and Anca Molnos

Univ. Grenoble Alpes, CEA, Leti, F-38000 Grenoble, France
{firstname.lastname}@cea.fr

Abstract. IoT nodes have faced several attacks aiming to compromise their confidentiality (illegitimately accessing private information), integrity (tampering with the data) and/or availability (denying/ discontinuing the provided service by the node). One way of compromising the node's security is by targeting its memory. Accessing the memory allows the attacker not only to access confidential data but also to have a complete control over the node. In this paper, we investigate the utilisation of nine state of art binary classifiers in detecting suspicious memory accesses (e.g., decision trees, Support Vector Machine SVM, Naive Bayes, Random Forest, Quadratic Discriminant Analysis among others). To evaluate the detection performance of the classifiers, a test use case with different dump attack strategies is developed. The classifiers are compared in terms of their detection accuracy, their complexity and the amount of memory leaked before detection. The results indicate that lightweight binary classifiers were able to flag as suspicious the attacks in their earliest stages.

Keywords: IoT · Detection · Memory access attack · Machine learning · Binary classifiers

1 Introduction

An IoT node can take many forms, from a simple sensor or actuator to an embedded system with its processing unit, a memory and micro-controller. IoT nodes are increasingly under attack especially due to their high exposure and their connectivity. Because of their limited computing and memory resources, the implementation of effective security mechanisms is challenging. Fuses and flash readout protections (RDP) are cost-effective solutions that prevent memory disclosure or alteration of memory contents when the system is deployed in the field. When the RDP is set at its highest security level, the memory is secured completely against logical readout attacks. However, the RDP is often set to a level that permits, to a certain extent, access to memory [13], e.g. because it is desirable to perform post-deployment upgrades. To prevent memory access attacks, two types of solutions can be employed: ensuring the confidentiality of memory contents, e.g., with encryption, or providing the nodes with

the ability to detect and react to attacks. Memory encryption, especially for program confidentiality, is expensive to use in practice for IoT nodes: it either requires dedicated hardware support or software implementations that incur a large performance overhead. As recent studies have demonstrated, memory encryption alone can no longer insure the security of the node [3, 4]. Attacks have found their way around encryption to retrieve sensitive data stored in encrypted memory [12] either by using side channel attacks [2, 15] or by simply physically accessing the memory where the decryption keys are stored [4].

To reinforce the security, attack detection tools can be implemented. Compared to the approaches above, attack detection is more pervasive as it can identify logical attacks, but also physical attacks as long as their manifestation affects some of the features monitored by the detection system. In this paper, we aim at showing that learning approach can be used to implement memory attacks detection tool in IoT nodes. We focus on highly constrained embedded systems, which have small computing resources (e.g. typically fitted with a 32-bit embedded micro-controller unit) and a limited memory capacity (e.g. typically less than 1 MiB of memory). Such systems are classified as Class 0 or Class 1 devices [7], or as Config1 [1]. The challenge that we tackle in this paper is the design of a detection system that is efficient enough to be implemented in such IoT nodes. To do so, our approach is composed of (1) a training phase, offline, which selects the most relevant features and the best detection method to be then implemented in the IoT node for the detection phase, online (2). The training phase can computing resources consuming while the detection phase needs to be light-weight.

Our contribution in this paper is a comparison a comparison between, state of the art, binary classifiers in terms of their detection accuracy, false alarms rate, complexity and the amount of leaked memory before detection.

The remaining part of this paper is as follows: Section 2 present the threat model and some existing detection techniques found in literature are briefly introduced in Section 3. In section 4, we present the use case, the attack strategies, the training and testing process of the classifiers and their performances are detailed. Finally, Section 5 sums up the findings of this work and the planned future work.

2 Threat model

IoT nodes generally have weak or non-existing memory access control exposing them to several types of attacks. Our work covers all the attacks that exploit a logical access to the memory system of the targeted device. Some examples of popular attacks are listed below:

Direct Memory Access Attack is a type of attack where an attacker uses a hardware mechanism called Direct Memory Access (DMA) to easily circumvent protection mechanisms built into the Operating System (OS) by attacking host runtime memory directly [16].

In **Bus Monitoring Attacks**, an attacker infers where secret data is stored from the CPU's memory patterns or from a specific application's memory accesses (e.g., AES), then tries to access this specific memory region [3].

Memory Dump Attack can be an attack on its own or a step in more sophisticated attacks [8, 6]. Dumping the target's entire memory or specific region is a key step in, including but not limited to, the attacks mentioned above.

3 Related work

To limit the risks of memory access attacks, two types of solutions can be employed: ensuring the confidentiality of memory contents, e.g., with encryption, or providing the nodes with the ability to detect and react to attacks. In this paper, we focus on the learned detection strategies targeted at memory access attacks. Learned detection, in general, relies on machine learning to define a model of normal behaviour of the monitored system. Any deviation from the learned normal behaviour is deemed anomalous. To build the nominal profile, this type of detectors depend on specific properties such as control flow [14], system call probe [19], hardware performance counters [18], instruction mix, among others. This type of solution has the advantage of being able to protect against a board spectrum of attacks. However, existing solutions of this type rely either on inaccessible properties for constrained nodes (e.g. some embedded systems have no implementation of performance counters) or rely on computationally expensive properties which necessitate prohibited amount of storage and computation resources and therefore designed for multi-core nodes (Config3 or higher).

Among the few learned detection solutions that are specifically targeted at memory attacks, we cite the work presented in [20]. Where the authors propose to detect suspicious behaviour of the node by characterizing the memory access of the operating system through a memory heat map. The heat map is combined with a an anomaly detection tool to detect any deviation from the known kernel memory access patterns. This solution however requires a multicore processor with a core entirely dedicated to the detection and a memory big enough to store several images of nominal behaviour (i.e., only compatible with Config3 or higher).

To suit the constrained nature of low-cost IoT node, we investigate the detection performance of light weight binary classifiers coupled with simple characteristics. Artificial Neural Networks (ANN) algorithms were not included in the list of candidate classifiers for their training and prediction time and memory complexity that exceed the allocated resources of a typical IoT node.

4 Comparing binary classifiers performance in detecting memory dump attacks

A learned memory attack detector is defined by a set of features extracted from raw data (ongoing memory access through the bus) and a detection rule learned utilising a classification method..

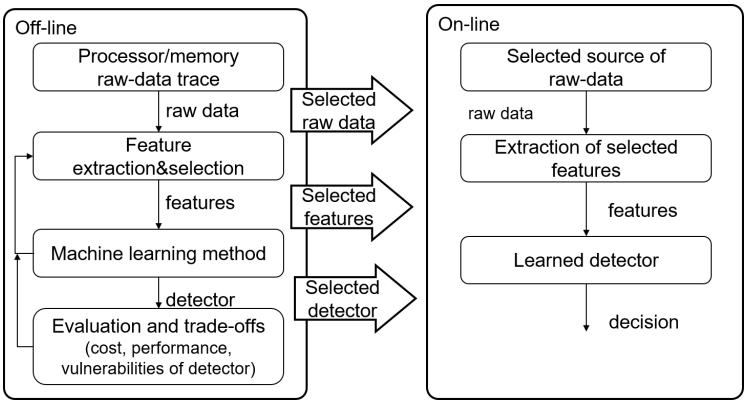

Fig. 1. Memory attack detection overview: training (Offline) and detection (On-line)

Fig. 1 illustrates our approach. The first step in designing the detector starts by choosing the source of information (raw data) to investigate (e.g., system execution control flow, memory access patterns, CPU activity...). In this paper, we choose the memory access as the raw data. The next step is to explore all the possible characteristics (features) to monitor. Different types of features can be used, namely number of load/store instructions, amount of accessed memory per time interval, memory occupancy grid, memory locality, register traffic etc. To choose the most relevant features for memory access detection, we use Recursive Feature Elimination [5]. These features are fed to different classifiers: Support Vector Machine with linear and radial basis function kernel (linear SVM and RBF SVM), Decision Tree (DT), Random Forest (RF), Logistic Regression (LR), K Nearest Neighbour (KNN), Naive Bayes, Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA). Each classifier builds a decision function or boundary that will be used as a detector. The detection performances of the detectors are compared in terms of their classification accuracy, their computation and memory requirements and their ability to detect other variants of attacks.

The detector to be implemented in the node will monitor the selected features using the detection rule learnt by the classifier that combines good detection performance and low computation/memory complexity.

4.1 Presentation of the use case

To evaluate the detection performance of the classifiers a test use case was developed (see Fig. 2). The thermostat use case reproduces a system on chip with a core and peripherals. This use case reproduces two periodic tasks and a random one to create the peripherals and user interactions. The first task simulates an interruption every ten seconds which reads the temperature value on a peripheral register. The second task, called each minute, computes the regulation loop

and sends orders to the heating by writing a radio frame on a register peripheral. The last task is called randomly. It simulates the user behaviour by pushing the plus or minus button of the thermostat to change the desired room temperature. The simulator generates a trace of all ongoing memory accesses through the bus. Each memory access is logged according to their timestamp, address, type (read or write), and a label to indicate whether the transaction is part of a nominal behaviour or not.

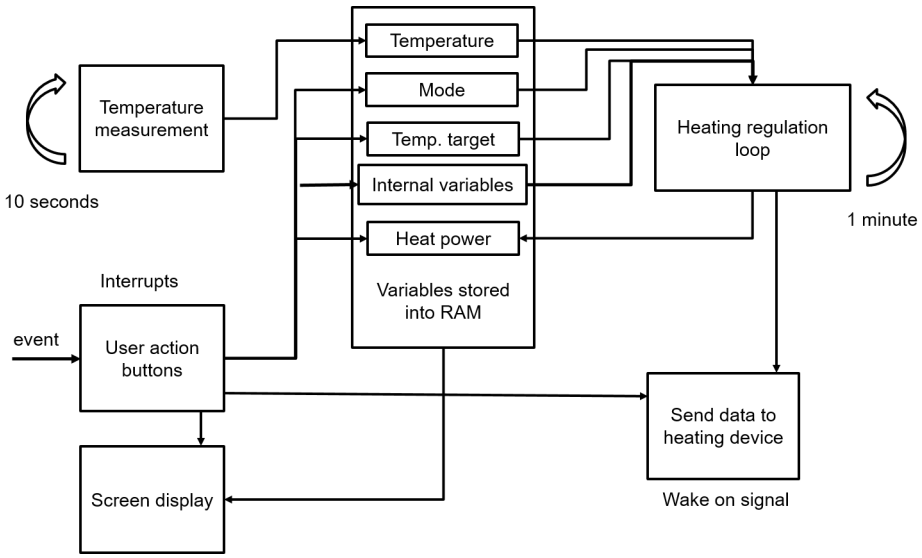

Fig. 2. Functional schema of the thermostat

4.2 Training and testing datasets

To obtain a training/testing dataset, we first ran the thermostat and collected observations of the nominal behaviour, each of which is extracted during a time of interval of 15 seconds. Every observation is represented by five extracted features (number of memory reads (NRead), number of memory accesses (NAcc), number of contiguous address increments, number of unknown accessed address and average time between two consecutive reads.

Seven variants of memory dump were launched on the thermostat: classic dump (CD); the attacker reads the entire memory in one go and in a contiguous way. The memory reads are spaced regularly in time. Dumping in bursts (DB); the memory is read in bursts, the accessed addresses are contiguous and the time step between two consecutive illegitimate memory reads is incremented either linearly (BD(lin)), randomly (BD(rand) or regularly BD(cts). In non-contiguous dump (NG), the attacker changes the way he/she sweeps the memory, the address

Use case	Training dataset	Testing dataset
Classic	(Nom + CD)	6 testing datasets each contains (Nom + DB) or (Nom + NG)
Mixed	(1) (Nom + (CD + NG + DB)) (2) (Nom + (CD + NG) (3) (Nom + (CD + DB))	(1) (Nom + (NG + CD + DB)) (1) (Nom + DB) (3) (Nom + NG)

Table 1. Training and testing datasets. Each training and testing data sets contains, in this order 1200 and 600 instances divided equally between the two classes and equally between different attack datasets

increment between two consecutive illegitimate memory reads is incremented either linearly, randomly or regularly.

During each memory dump attack variant, observations are extracted and a record of the amount of leaked memory during each time interval is kept. The record is used during the evaluation of the detectors.

The collected observations are used to build training and testing datasets as shown in Tab. 1.

4.3 Detection performance of classifiers trained on classic dump

The classifiers were trained on the Classic training set to learn the parameters of the decision boundaries. In order to tune the hyper-parameters of the classifiers and to increase their accuracy, we performed the holdout cross-validation [9]. Fig. 3 shows the resulting decision boundaries obtained after the training, and Fig. 4 shows the application of the learnt decision boundaries on the dump attack variant DB(rand). We emphasize on the fact that these two figures represent a projection of our 5-D dataset on a 2-D representation space.

Fig. 3 illustrates that observations of the two classes are separable. SVM-rbf, QDA, Naive Bayes and RF draw a non-linear boundary to separate the two classes, while KNN, decision tree, linear SVM and logistic regression draw a linear boundary to separate the two. For this dataset, a linear boundary is more appropriate as it has a low classification complexity and a high training accuracy.

However, a detector must be able to detect different attacks from what is used during the training. Attacks that can be manifested differently in the input space. For this purpose, we tested the learnt decision boundaries on the other dump attack variants (Fig. 4).

The position of the observations extracted during the attack variant BD(rand) in the input space (as can be see in Fig. 4) is not only closer to nominal class than what observed in Fig. 3 and placed in a region that most classifiers (especially the linear ones) defined as nominal. But also, made the attack and nominal classes no longer linearly separable.

The exact detection accuracy of the classifiers as well as the amount of leaked memory before detection are shown in Fig. 5. As can be read on the figure, the linear boundaries accuracy, as expected, scored poorly on the six test datasets.

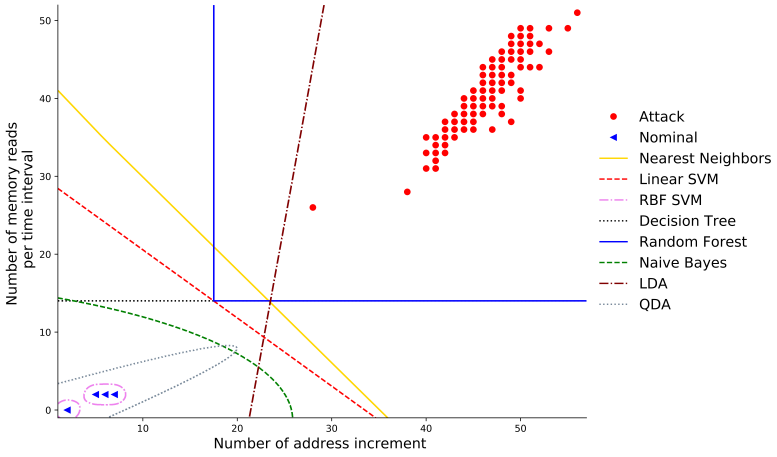

Fig. 3. Decision boundaries built by different binary classifiers trained on classic dataset. Projection of the 5-D dataset on the 2-D space *number of memory reads per time interval* vs. *Number of address increment*. The time interval is set to 15 s.

The SVM-rbf are QDA were able to perform better (more than 85% of observations were correctly classified and the corresponding average memory leakage was under 100 bytes in the worst case (less than 60 bytes for SVM-rbf). This better performance can be explained by the fact that the decision boundaries of these two classifiers are created in higher dimensional space mapped by a kernel function. In that higher dimensional space, the observations of the new attacks can still be separable. It should be noted, that both SVM-rbf and QDA were able to flag the observations extracted during the time window of the start of attacks as suspicious. However, as the detection verdict is announced at the end of the time window, a memory leakage is observed. The amount of memory leakage can be reduced by shortening the time window.

4.4 Detection performance of classifiers trained on several attacks

The classes in the mixed datasets are not linearly separable in the input space. So only classifiers that can learn non-linear boundaries are compared in this section.

By training on mixed datasets, the performance of the classifiers has increased measurably (see Fig. 6). For example, the average accuracy of DT with the Classic dataset was 60% while on the Mixed dataset it reached more than 94%. SVM-rbf, NB and QDA continued to score the highest overall accuracy among the classifiers, with average accuracies respectively of 96%, 94% and 95%. The increase in classification accuracy is accompanied with an increase of the complexity of the learnt decision boundaries. For example, SVM-rbf boundary is expressed in a function of support vectors (closest data points to the learnt hyperplane), its time/memory complexity is influenced by the number of support

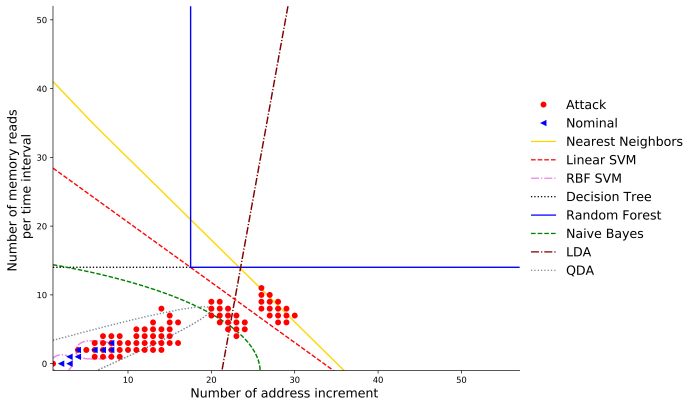

Fig. 4. Decision boundaries tested on DB(lin). Projection of the 5-D dataset on the 2-D space *number of memory reads per time interval* vs. *number of address increment*. The time interval is set to 15 seconds.

vectors chosen during the training (time and space complexity is $O(n_{sv} \times n_f)$ with n_{sv} the number of support vectors and n_f number of features).

When SVM-rbf was trained on classic dataset, only 4 support vectors were used in defining the decision boundary learned from classic dataset, while 110 support vectors were needed for the the decision boundary learned from mixed dataset. Same thing, was noticed on the constructed decision tree from classic dataset compared to mixed dataset. The depth of decision tree increased from 1 (classic dataset) to 4, influencing by that memory and computation complexity. The computation/memory complexity of QDA decision boundary, however, has maintained the same level (as it only depends on features' number $O(n_f^2)$ for time complexity and $O(n_f^3)$ for the space complexity). As only 5 features are calculated during each time window, the time and space complexity of QDA is potentially suitable for an IoT node that is resources constrained.

5 Conclusion

In this work, we have compared the detection performance and leakage before detection of nine binary classifiers used to detect memory access attack. Two training strategies were used, learning from one type of attack and testing on other types of attacks and learning from multiple types of attacks. Quadratic Discriminant Analysis (QDA) presented the best compromise between classification accuracy and computation/memory complexity for the thermostat use case. Further studies are needed to determine the best binary classifier for memory access attacks along with estimating the gain in time/memory when such detection is implemented compared to existing learned detection strategies. We will also investigate the inflicted cost inflicted in order for the attacker to bypass such detector.

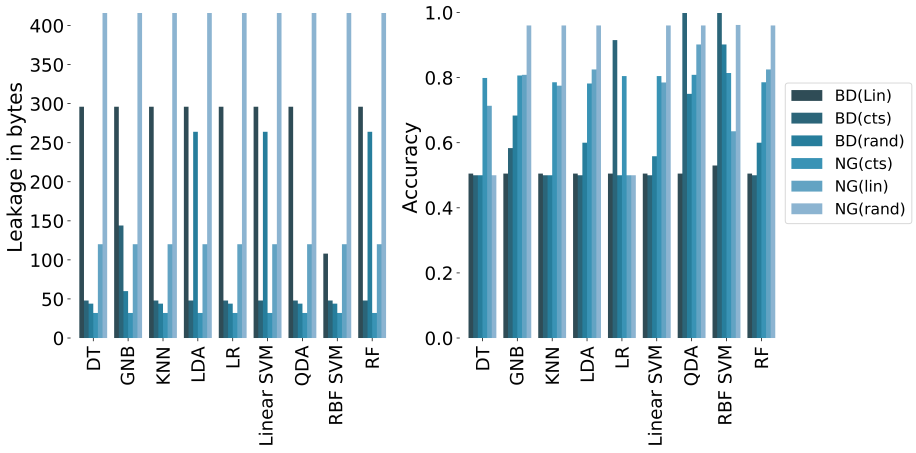

Fig. 5. Performance comparison of the classifiers, trained on the Classic dataset.

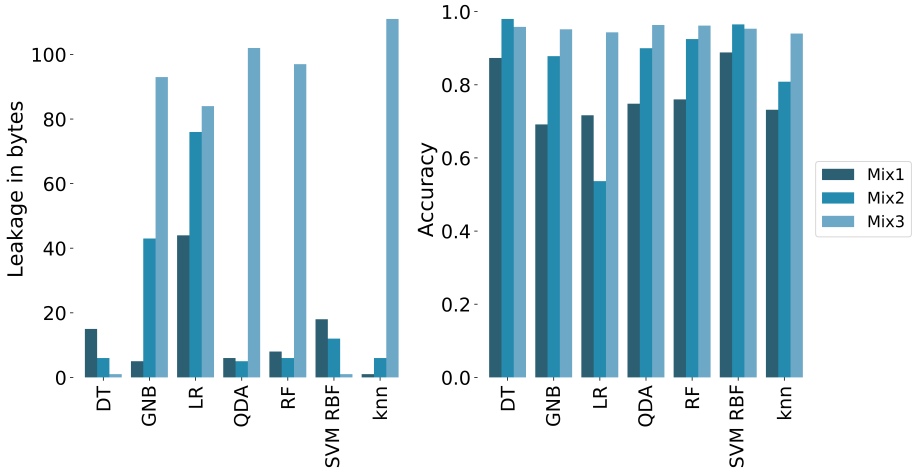

Fig. 6. Performance comparison of the classifiers, trained on the Mixed dataset.

Acknowledgements This work was partially supported by the French ANR via Carnot funding, and by the French national program "Programme Investissements d'Avenir IRT Nanoelec" ANR-10-AIRT-05.

References

1. Adegbija, T., Rogacs, A., Patel, C., Gordon-Ross, A.: Microprocessor Optimizations for the Internet of Things: A Survey. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **37**(1) (Jan 2018)

2. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Theory of Cryptography Conference. pp. 474–495. Springer (2009)
3. Colp, P., Zhang, J., Gleeson, J., Suneja, S., De Lara, E., Raj, H., Saroiu, S., Wolman, A.: Protecting data on smartphones and tablets from memory attacks. ACM SIGARCH Computer Architecture News (1), 177–189 (2015)
4. Dorrendorf, L.: Protecting drive encryption systems against memory attacks. IACR Cryptology ePrint Archive p. 221 (2011)
5. Guyon, I., Elisseeff, A.: An introduction to feature extraction. In: Feature extraction, pp. 1–25. Springer (2006)
6. Jimenez, J.C.: Hack the world practical reverse engineering a router. jcjc-dev.com/2016/06/08/reversing-huawei-4-dumping-flash/, accessed: 2018-18-04
7. Keranen, A., Ersue, M., Bormann, C.: Terminology for Constrained-Node Networks. https://tools.ietf.org/html/rfc7228
8. Kim, S., Cheon, J.H., Joye, M., Lim, S., Mambo, M., Won, D., Zheng, Y.: Strong adaptive chosen-ciphertext attacks with memory dump (or: The importance of the order of decryption and validation). In: IMA International Conference on Cryptography and Coding. pp. 114–127. Springer (2001)
9. Kohavi, R., et al.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. vol. 14, pp. 1137–1145. Montreal, Canada (1995)
10. Lee, Y.W., Moh, S., Muller, A.: Memory monitoring circuit for detecting unauthorized memory access (1998), uS Patent 5,729,716
11. Mazhelis, O.: One-class classifiers: a review and analysis of suitability in the context of mobile-masquerader detection. South African Computer Journal (36), 29–48 (2006)
12. Müller, T., Freiling, F.C.: A systematic assessment of the security of full disk encryption. IEEE Transactions on Dependable and Secure Computing **12**(5), 491–503 (2015)
13. Obermaier, J., Tatschner, S.: Shedding too much light on a microcontrollers firmware protection. In: 11th USENIX Workshop on Offensive Technologies (WOOT 17). USENIX Association (2017)
14. Petroni Jr, N.L., Hicks, M.: Automated detection of persistent kernel control-flow attacks. In: Proceedings of the 14th ACM conference on Computer and communications security. pp. 103–115. ACM (2007)
15. Ronen, E., Shamir, A., Weingarten, A.O., OFlynn, C.: IoT Goes Nuclear: Creating a ZigBee Chain Reaction. pp. 195–212. IEEE (May 2017). https://doi.org/10.1109/SP.2017.14
16. Stewin, P., Bystrov, I.: Understanding dma malware. In: International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 21–41. Springer (2012)
17. Von Gemuenden, D.: Detecting a read access to unallocated or uninitialized memory (2016), uS Patent App. 15/038,243
18. Xia, Y., Liu, Y., Chen, H., Zang, B.: Cfimon: Detecting violation of control flow integrity using performance counters. In: Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on. pp. 1–12
19. Yoon, M.K., Mohan, S., Choi, J., Christodorescu, M., Sha, L.: Learning execution contexts from system call distribution for anomaly detection in smart embedded system. In: Internet-of-Things Design and Implementation (IoTDI), 2017 IEEE/ACM Second International Conference on. pp. 191–196. IEEE

20. Yoon, M.K., Mohan, S., Choi, J., Sha, L.: Memory heat map: Anomaly detection in real-time embedded systems using memory behavior. In: Proceedings of the 52Nd Annual Design Automation Conference. pp. 35:1–35:6. ACM, New York, NY, USA (2015)