Obfuscated Memory Malware Detection

Subjects: Computer Science, Artificial Intelligence

Contributor: Sakib Shahriar Shafin , Gour Karmakar , Iven Mareels

Obfuscated Memory Malware (OMM) presents significant threats to interconnected systems, including smart city applications, for its ability to evade detection through concealment tactics. Existing OMM detection methods primarily focus on binary detection.

multiclass memory malware detection smart city deep learning

1. Introduction

The exponential increase in online activities, particularly during the Covid-19 pandemic [1], has led to significant growth toward building online infrastructures for numerous new and existing services, resulting in an unprecedented amount of data being processed and stored in cyberspace. For example, the global cloud services market is expected to grow from USD 396.1 billion in 2020 to USD 798.84 billion in 2025 at an annual growth rate of 14% [2]. As governments and organizations plan to deliver increasingly advanced services in smart city areas worldwide and factories embrace Industry 4.0 using Internet of Things (IoT) networks, it is projected that the number of IoT devices will surpass 30 billion by 2030. The multifold possibilities within a smart city concept include, among others, efficient and eco-friendly usage of technology to enhance the quality of services in healthcare [3], coordinated development boosted by smart economy [4], transportation, water, air quality management, waste management and surveillance. However, the main requirement of a smart city is connectivity across all devices and all aspects, which can only be a possibility if IoT is used on a mass scale [5], which is further proved by a recent study that shows about 70% of the USA businesses have invested heavily in Industrial IoT [6] and devices in smart homes and wearables, which directly contributes to the growth of smart cities.

To understand the vulnerabilities and underlying security challenges of smart city applications, it is crucial to understand the threats and vulnerabilities of the sensors on which those applications are built and associated threat mitigation strategies. Sensors detect and respond to physical stimuli, such as changes in temperature, pressure or motion, and convert them into analog or digital signals for subsequent processing and decision-making. From existing IoT-based smart city applications, it is evident that a wide range of sensors are now being used to operate and monitor the functions of smart cities that reach almost all facets of society. For example, one of the largest smart city projects was undertaken at the city of Santander, Spain, which utilizes Libelium's Waspmote sensor platform [7], a versatile and modular sensor network analysis space, to monitor the environmental conditions of the city systems. In this project, Meshlium scanners are used as edge devices to gather data, while 750 sensors were deployed across 22 specified zones. Temperature, luminosity, carbon monoxide and sound noise sensors were employed. Libelium's dedicated sensor nodes for smart cities, the Plug

and Sense! Smart Cities PRO, equipped with two radios for 2.4 GHz communication and IEEE 802.15.4 protocol as standard are developed exclusively for this purpose. They contain BME280 temperature, humidity and pressure sensor, along with SCP v30 07 luminosity sensor, OPC-N3 dust sensor and CO-A4 carbon monoxide sensors [8]. Libelium's next advanced project on smart city is currently under development at the city of Cartagena [9], where lampposts were integrated with air quality (OPC N3) and noise monitoring sensors equipped with fast cellular communication standards (5G). Utilizing OPC-N2 technology, AirSensa spearheads a large-scale air pollution monitoring endeavor in London, capturing precise measurements of PM1, PM2.5 and PM10 concentrations.

The Smart Nation Sensor Platform in Singapore [10] uses sensors to perform important tasks for smart city monitoring such as detecting water leaks, minimizing energy wastage and incident reporting on lightning strikes, where they integrated Vaisala GLD360 sensor network [11]. To monitor and keep track of environmental pollution, a major task for smart cities, Lecce, Italy is using air quality sensors in its "Integrated Energy Plan" to reduce CO2 emissions. A project on sustainable environment is taking place at Tartu, Estonia and Sonderberg, Norway who are utilizing smart meters and sensor-based monitoring [12] to improve energy efficiency in community housing and optimize solar energy usage. In Vitoria-Gasteiz, low-level sensors measuring temperature, humidity and CO2 are installed in dwellings, along with energy consumption measuring devices to improve living conditions [13]. The integration of sensors in smart city projects is essential for effective data-driven decision-making, and addressing the diverse challenges facing modern cities.

The exponential use of these sensors and IoT devices in smart cities and regular IoT environments as a whole present a unique set of challenges. Because of their resource-constrained nature and criticality of some applications, these devices are particularly vulnerable to complex and hard-to-detect attacks [14]. As most devices used in the lower tiers of IoT systems are left with up to 1 MB only in memory after installation of the operating system [15], resource-intensive security mechanisms are practically unfeasible. The importance of sectors where IoT devices are deployed, such as medical, energy and military operations, compounded by the scarcity of resources, makes them an easy target for cyber-attacks [16].

In the cyber attack landscape, malware is the most vicious threat to security. Their ability to stay undetected in systems and deploy automated coordinated attacks makes them particularly destructive for distributed systems such as IoT and Smart cities. Attacks such as RapperBot (August 2022), a recent variation of Mirai malware [17], infected a large number of IoT devices with a combined attack from over 3500 unique IPs, underpinning the importance of protecting resource-constraint devices from cyberattacks. Obfuscated memory malware (OMM) is a particularly notorious form of malware that employs obfuscation techniques to obscure its presence in the device memory and hide its activities such as code scrambling, command-and-control, string compression and encryption, and code injection to evade detection. Their polymorphic ability to change their behavior with each iteration also makes their goals impossible to understand, thus making it difficult to extract information about the intent of the malware. OMMs include some of the most dangerous viruses, such as Ransomware, Spyware and Trojans [18]. Thus, OMMs have certainly become powerful tools for infiltrating secure networks and stealing or destroying valuable information. Consequently, there has been a growing interest in the development of robust and efficient detection mechanisms that can analyze and identify OMMs in memory.

Recent advances in deep learning (DL) have led to an increased use of advanced neural networks algorithms, such as Convolutional Neural Networks (CNNs) and Long-Short Term Memory (LSTMs), to detect and identify malware [19]. However, it is important to consider the complexity and sporadic nature of OMMs activity patterns, especially when performing multiclass detection (i.e., identifying individual attack types), as the nature of these malware makes the task more challenging. While researchers have implemented a wide range of singular and hybrid models for malware detection, most of the research has focused on binary detection only, which detects the presence/absence of attack within a system. Multiclass detection is critical for embedded and constrained devices to optimize their security strategy by observing and investigating the specific nature of each attack type and devising more custom malware prevention systems. Another major challenge is that most existing methods for detecting OMMs fail to deliver sufficient detection accuracy while maintaining small model sizes. Very few works have focused on models specifically tailored to resource-constrained environments, as most DL methods consist of many layers and a large number of parameters, resulting in model sizes that exceed 1MB and are impractical for deployment on end devices [15].

Compared to LSTM which has a tendency to forget patterns after a while, Bi-directional LSTMs, have the unique ability to extract rich feature representations and model complex patterns while considering both past and future contexts [20]. They are adept at retaining or discarding critical and redundant information through the use of gates, thus reducing the resources required. These capabilities are useful to overcome the challenges posed by OMMs in targeted systems.

2. Obfuscated Memory Malware Detection

2.1. Detecting the Presence of an OMM Attack (Binary Classification)

Lee et al. [21], present an anti-obfuscation classification method for Android malicious applications that integrates Recurrent Neural Network (RNN) and CNN for their ability to learn patterns in sequence data. VirusTotal was used to collect data for the study, extracting application package name, authentication data, permission, and intention features from multiple short strings. The authors attempted to achieve both anti-obfuscation capability and a lightweight design by focusing on package and certificate data as inputs because they are simple but easy to interpret for validation. The evaluation steps included five separate models, from Ngram feature model to using CNN and RNN. A model that combines CNN and RNN with additional features about permission and taken actions attains the highest FPR (False Positive Rate) detection rate of 97.7% for binary classification. The evaluation dataset is also particularly not made of obfuscated malware, but rather random Android attack samples. Therefore, the reported results are unlikely to hold true when tested on obfuscated dataset. Furthermore, the size of the model weights is about 4 MB, and the training time per epoch reaches 8 min, which is infeasible for IoT devices.

In [22], authors examine the challenge of making traditional detection systems resistant to obfuscation in malware. The authors analyze various popular virus scanners and conclude that current detection algorithms are significantly less effective when faced with obfuscated malware. The objective of the study is to develop a model called Obfusifier, which while trained on unobfuscated data, can accurately predict when it encounters obfuscated

malware. For that, the authors assume certain parts of malware code, such as API invocations and control messages, cannot be obfuscated by attackers without breaking the functionality of the malware code. This hypothesis is utilized to extract features centered on API usage from the VirusShare malware repository. The proposed model employs JITANA [23] to generate method graphs for understanding call relationships in VirusShare apps. The model utilizes Google-provided API lists to only retain unchangeable, simple android API graphs. Using depth-first search, the model analyzes critical API call frequencies to gain insight into the generation of sensitive communication paths. Eight features, each from the original graph, simplified graphs and sensitive paths, are used to build the dataset. Decision Tree, Random Forest and Support Vector Machine are applied to the data with 10-fold cross-validation. After extensive training, the highest accuracy of 95% is achieved using Random Forest for binary classification. While the model demonstrates detection ability to some degree of obfuscation, it is unable to detect malicious inter-app communications between obfuscated attackers and is unable to detect malware if obfuscation is performed on the native code instead of applying at runtime.

Stacked Ensemble was employed in [18], to classify memory malware. They design and create a dataset comprising very recent and advanced obfuscated attacks on device memory, where features were extracted from memory dumps using VolMemLyzer, a feature extractor for machine learning systems. The dataset consists of 58,596 samples, containing 15 attack types belonging to Ransomware, spyware and trojan families. At first, they evaluate the detection accuracy of the dataset through traditional algorithms such as Random Forest, Decision Tree, kNN, SVM and Linear Perceptron. Of those methods, Random Forest attains the highest accuracy of 98%. To further increase the accuracy, an ensemble model using stacked generalization was created with Naïve-Bayes, Random Forest and Decision Tree as base and Logistic Regression as Meta-Learner. The method improves the accuracy to 99.02%. However, the study did not implement attack classification, which is crucial for dealing with the increasing number of attack types and variations. Additionally, the model takes 1 s to predict only 125 samples, which can create latency in networks that require constant data streaming and monitoring, such as IoT systems, making it not efficient for real-time monitoring scenarios.

The CIC-Malmem-2022 dataset has garnered considerable attention in recent literature. In [24], the authors employ oversampling and XGBoost as preprocessing techniques to combat class imbalance in the dataset, and after evaluating multiple algorithms, their experiments saw that Random Forest and Multilayer Perceptron(MLP) surpass the performance of other models, achieving an accuracy rate of almost 100%, although simply for detection. Although the study is commendable, it lacks the evaluation of individual attack classification. Furthermore, the ratio of attack and benign samples is equal in the dataset (Section 3), thus raising questions about the necessity for SMOTE, which is typically used for imbalanced datasets. Another contribution is the study by Louk et al. [25], who specifically analyze the performance of tree-based ensembles for binary classification of malware, with a specific focus on evaluating the performance of Xgboost, Catboost, LightGBM, and Random Forest. However, the authors do not specifically concentrate on building a model for obfuscated malware and instead aim to assess the generalizability of the algorithms across various contexts. The results of the study highlight Xgboost as the top performer.

2.2. Detecting Attack Families or Attack Types (Multiclass Classification)

Kim et al. [26] delve into the use of both global and local features for detecting obfuscated malware, with a specific focus on the efficacy of a variational autoencoder in defining a latent space to learn complex distributions of virus 2D vectors, which were transformed into grayscale images to gain insight into global features. The research also meticulously examines the capabilities of local features in describing distinct variations in small patterns within images. The team leveraged a Generative Adversarial Network model to extract global features, as it enables the production of data through random sampling, thereby expanding the range of detectable malware. The proposed method was rigorously evaluated using the Microsoft BIG 2015 dataset, where samples were used for image generation. The autoencoder was trained with a different mixture of Gaussian distributions, then transfer learning was used to build the detector, where the encoder is reused to inherit the original capacity of learning. The model achieves a detection accuracy of 97.47% for the detection of eight classes of malware. However, the study did not evaluate the optimal selection of features, which could have reduced redundancy in the feature space, resulting in improved performance, and neither did they consider any constraints in terms of resources. The assumption that the file sizes of the graphs based on the feature are sufficient to deduce whether an app contains malicious content or not is not reliable in cases where the attacker employs compression attributes.

A hybrid deep learning model was introduced in [27] that incorporates a deep belief network with a gate recurrent unit (GRU) for the detection of obfuscated malware on Android devices. They aim to make their model less error-prone by adding dynamic features collected at runtime alongside normal static features. The DBN feed-forward network is employed for static features, but its inability to detect dynamic obfuscated malware requires the addition of GRUs, due to their ability to capture temporal dependencies in the input data. Despite the authors' effort to design a relatively simple model comprising only 5 to 25 neurons, the error rate remains substantial, ranging from 8 to 35%. The study does not prioritize the establishment of an optimal balance between model complexity and accuracy, but instead, it seeks to demonstrate that an increase in the number of neurons can aid in the more accurate detection of disguised malware. The amount of data used in the study is relatively small, only 280 samples, and although they show that an increase in neurons per layer can improve detection accuracy, the false alarm rate is still considerably high. Further evaluation with more samples might be of use to improve the overall performance of the model.

In [28], an approach to detect obfuscated malware was proposed, which uses the network signal behavioral signatures and API call patterns derived from a simulation environment established using a Cuckoo sandbox. More than 270,000 samples were generated of which 32000 were used. The authors reiterate the assumption made in [22] that API calls and behaviors cannot be obfuscated and uses those features to detect malware. To identify critical features, the authors utilized an Information Gain ratio, which was incorporated into a Random Forest algorithm to not only detect malware attacks but also classify them into five classes. By focusing on the sequences and frequency of API calls and augmenting this feature set with 55 additional features to enhance family classification, the system provides a granular understanding of malware behavior in the simulated environment. However, despite claims of reduced feature space and improved computational efficiency, the authors fail to provide empirical evidence to support these claims.

References

1. Lallie, H.S.; Shepherd, L.A.; Nurse, J.R.; Erola, A.; Epiphaniou, G.; Maple, C.; Bellekens, X. Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic. Comput. Secur. 2021, 105, 102248.
2. The Business Research Company. Cloud Services Global Market Briefing 2021: COVID 19 Impact and Recovery. 2021. Available online: https://www.marketresearch.com/Business-Research-Company-v4006/Cloud-Services-Global-Briefing-Covid-30435480/ (accessed on 13 March 2023).
3. Saba, T. Intrusion detection in smart city hospitals using ensemble classifiers. In Proceedings of the 2020 13th International Conference on Developments in eSystems Engineering (DeSE), Liverpool, UK, 14–17 December 2020; pp. 418–422.
4. Chen, Z.; Chan, I.C.C. Smart cities and quality of life: A quantitative analysis of citizens' support for smart city development. Inf. Technol. People 2023, 36, 263–285.
5. Arasteh, H.; Hosseinnezhad, V.; Loia, V.; Tommasetti, A.; Troisi, O.; Shafie-khah, M.; Siano, P. Iot-based smart cities: A survey. In Proceedings of the 2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC), Florence, Italy, 7–10 June 2016; pp. 1–6.
6. Honeywell International Inc. Why the Industrial Internet of Things Matters. 2019. Available online: https://www.honeywell.com/us/en/news/2019/06/why-the-industrial-internet-of-things-matters (accessed on 13 March 2023).
7. Libelium. Smart Santander: The Most Ambitious Smart City Project in Europe. Available online: https://bit.ly/3qmoYzr (accessed on 25 April 2023).
8. Libelium. Libelium Smart Cities PRO Sensor Guide. Available online: https://development.libelium.com/smart-cities-pro-sensor-guide/sensors (accessed on 25 April 2023).
9. Libelium. Smart Lampposts in Cartagena to Measure Air Quality and Noise. Available online: https://www.libelium.com/libeliumworld/success-stories/smart-lampposts-in-cartagena-to-measure-air-quality-and-noise (accessed on 9 May 2023).
10. Smart Nation Singapore. Smart Nation Sensor Platform. Available online: https://www.smartnation.gov.sg/initiatives/strategic-national-projects/smart-nation-sensor-platform (accessed on 25 April 2023).
11. Vaisala. Lightning Density Maps for Every Country in the World. Blog Post. 2023. Available online: https://www.vaisala.com/en/blog/2023-03/lightning-density-maps-every-country-world (accessed on 15 May 2023).

12. SmartEnCity. SmartEnCity-Smart Zero Carbon City Solutions. Available online: https://smartencity.eu/ (accessed on 25 April 2023).
13. Larrinaga, F.; Pérez, A.; Aldalur, I.; Hernández, J.L.; Izkara, J.L.; Sáez de Viteri, P. A holistic and interoperable approach towards the implementation of services for the digital transformation of smart cities: The case of Vitoria-Gasteiz (Spain). Sensors 2021, 21, 8061.
14. Shalaginov, A.; Azad, M.A. Securing resource-constrained iot nodes: Towards intelligent microcontroller-based attack detection in distributed smart applications. Future Internet 2021, 13, 272.
15. Mohaimenuzzaman, M.; Bergmeir, C.; Meyer, B. Pruning vs XNOR-net: A comprehensive study of deep learning for audio classification on edge-devices. IEEE Access 2022, 10, 6696–6707.
16. Rashid, M.M.; Kamruzzaman, J.; Imam, T.; Kaisar, S.; Alam, M.J. Cyber attacks detection from smart city applications using artificial neural network. In Proceedings of the 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Gold Coast, Australia, 16–18 December 2020; pp. 1–6.
17. SocRadar. Linux Malware “Rapper” Bot Brute Forcing SSH Servers. 2022. Available online: https://socradar.io/linux-malware-rapperbot-brute-forcing-ssh-servers/ (accessed on 13 March 2023).
18. Carrier, T.; Victor, P.; Tekeoglu, A.; Lashkari, A.H. Detecting Obfuscated Malware using Memory Feature Engineering. In Proceedings of the ICISSP, Online Streaming, 9–11 February 2022; pp. 177–188.
19. Hosseini, S.; Nezhad, A.E.; Seilani, H. Android malware classification using convolutional neural network and LSTM. J. Comput. Virol. Hacking Tech. 2021, 17, 307–318.
20. Zhang, S.; Zheng, D.; Hu, X.; Yang, M. Bidirectional long short-term memory networks for relation classification. In Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation, Shanghai, China, 30 October–1 November 2015; pp. 73–78.
21. Lee, W.Y.; Saxe, J.; Harang, R. SeqDroid: Obfuscated Android malware detection using stacked convolutional and recurrent neural networks. In Deep Learning Applications for Cyber Security; Springer: Berlin/Heidelberg, Germany, 2019; pp. 197–210.
22. Li, Z.; Sun, J.; Yan, Q.; Srisa-an, W.; Tsutano, Y. Obfusifier: Obfuscation-resistant android malware detection system. In Proceedings of the Security and Privacy in Communication Networks: 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, 23–25 October 2019; Proceedings, Part I 15; Springer: Berlin/Heidelberg, Germany, 2019; pp. 214–234.
23. Tsutano, Y.; Bachala, S.; Srisa-an, W.; Rothermel, G.; Dinh, J. Jitana: A modern hybrid program analysis framework for android platforms. J. Comput. Lang. 2019, 52, 55–71.

24. Talukder, M.A.; Hasan, K.F.; Islam, M.M.; Uddin, M.A.; Akhter, A.; Yousuf, M.A.; Alharbi, F.; Moni, M.A. A dependable hybrid machine learning model for network intrusion detection. J. Inf. Secur. Appl. 2023, 72, 103405.
25. Louk, M.H.L.; Tama, B.A. Tree-Based Classifier Ensembles for PE Malware Analysis: A Performance Revisit. Algorithms 2022, 15, 332.
26. Kim, J.Y.; Cho, S.B. Obfuscated malware detection using deep generative model based on global/local features. Comput. Secur. 2022, 112, 102501.
27. Kolli, S.; Balakesavareddy, P.; Saravanan, D. Neural Network based Obfuscated Malware detection. In Proceedings of the 2021 International Conference on System, Computation, Automation and Networking (ICSCAN), Puducherry, India, 30–31 July 2021; pp. 1–5.
28. Hansen, S.S.; Larsen, T.M.T.; Stevanovic, M.; Pedersen, J.M. An approach for detection and family classification of malware based on behavioral analysis. In Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC), Kauai, HI, USA, 15–18 February 2016; pp. 1–5.