Cybersecurity

RESEARCH

Open Access

Machine learning based fileless malware traffic classification using image visualization

Fikirte Ayalke Demmese[1*], Ajaya Neupane[2], Sajad Khorsandroo[1], May Wang[2], Kaushik Roy[1] and Yu Fu[2]

Abstract

In today's interconnected world, network traffic is replete with adversarial attacks. As technology evolves, these attacks are also becoming increasingly sophisticated, making them even harder to detect. Fortunately, artificial intelligence (AI) and, specifically machine learning (ML), have shown great success in fast and accurate detection, classification, and even analysis of such threats. Accordingly, there is a growing body of literature addressing how subfields of AI/ML (e.g., natural language processing (NLP)) are getting leveraged to accurately detect evasive malicious patterns in network traffic. In this paper, we delve into the current advancements in ML-based network traffic classification using image visualization. Through a rigorous experimental methodology, we first explore the process of network traffic to image conversion. Subsequently, we investigate how machine learning techniques can effectively leverage image visualization to accurately classify evasive malicious traces within network traffic. Through the utilization of production-level tools and utilities in realistic experiments, our proposed solution achieves an impressive accuracy rate of 99.48% in detecting fileless malware, which is widely regarded as one of the most elusive classes of malicious software.

Keywords Network security, Traffic classification, Fileless malware, Image visualization, Machine learning, Intrusion detection

Introduction

Network traffic flow classification is an essential network function that paves the way for dynamic and agile network management. It empowers network operators to handle different service requirements and constraints to allocate resources more efficiently while maximizing their utilization. It also enables security-based network traffic engineers to identify malicious patterns affecting network services' availability and performance. As most devices maintain their connectivity through networks, most cyber threats are triggered through network traffic which ranges from malware infection to distributed denial of service (DDoS) attacks. Hence, it is essential to identify malicious network traffic that targets the underlying devices.

Fileless malware (Kumar 2020) is a type of evasive malware that is notable for its capability to reside solely in the system's main memory, without leaving traces on the disk or file system. These malware attacks employ manipulation techniques on legitimate libraries and utilities of benign platforms to achieve their objectives (Saad et al. 2019). Conventional methods of file-based detection operate by scanning for any malicious programs or software. However, benign software that is included in a whitelist is never subjected to testing because the malware detection system does not classify them as malicious (Saad et al. 2019). This allows fileless malware to exploit vulnerabilities in trusted and widely used applications (Smelcer 2017), such as web browsers, text-processing applications, and video players. Commonly abused

*Correspondence:
Fikirte Ayalke Demmese
fademmese@aggies.ncat.edu
[1] Department of Computer Science, College of Engineering, North Carolina A&T State University, 1601 E Market St, Greensboro, NC 27411, USA
[2] Palo Alto Networks, Inc., 3000 Tannery Way, Santa Clara, CA 95054, USA

tools by fileless malware include Bitsadmin, certutil, MS Office macros, mshta, msiexec, PowerShell, psexec, registry, regsvr32, task scheduler, WMI, and VBscript (Borana et al. 2021).

Popular malware detection systems like Signature-based and Static analysis are ineffective at detecting fileless malware because this type of malware does not leave any residual traces in the file system after the attack has concluded. As a result, fileless malware has an advantage in evading traditional detection solutions (Barnes 2021). The ability to leave no trace after an attack also makes it challenging for forensic investigators to use reverse engineering to analyze fileless malware attacks (Saad et al. 2019).

In general, three unique properties of fileless malware make them different from file-based malware (Babar 2020). First, fileless malware does not require data to be written on the target's file system at the time of infection and propagation. Second, it does not download any file on disk. Third, fileless malware depends on pre-installed benign software libraries and utilities to execute the malicious payload.

Recent advances in combining network traffic visualization with machine learning techniques have shown promising results in the detection of malicious patterns in network traffic (Joo et al. 2021; Shapira and Shavitt 2019; Wang et al. 2017). The conversion of network traffic to image facilitates the visualization of possible changes in network traffic patterns before and during different attacks. Research has shown that network traffic exhibits strong patterns of behavior across multiple timescales (Taheri et al. 2018; Wang et al. 2017; Xu et al. 2021). By leveraging visualization techniques, the complex patterns and characteristics within network data can be effectively analyzed, enabling the machine learning algorithms to make more informed and precise classifications. This combination of visualization and machine learning plays a vital role in enhancing the effectiveness and reliability of detecting malicious activities in network traffic. Image-based techniques can offer advanced visual aids that enable the quick detection of suspicious, unknown malware and the timely alerting of anomalous behavior patterns. Representing malware as an image enables the detection of even small changes while preserving the overall structure of samples from the same malware family (Gibert et al. 2019). Visual representations of malware patterns can also provide a concise overview of potential attacks. Additionally, converting malicious executables into images makes it easier to differentiate between different binary sections.

Distinct sections of the executable binary are depicted as distinctive image textures, which facilitates faster analysis and can increase the productivity of malware analysts. Figure 1 demonstrates that related malware classes exhibit stronger visual similarities with one another than with unrelated classes, as shown in Fig. 2.

Summary of contributions

This paper introduces a solution for detecting fileless malware traffic using image visualization based on

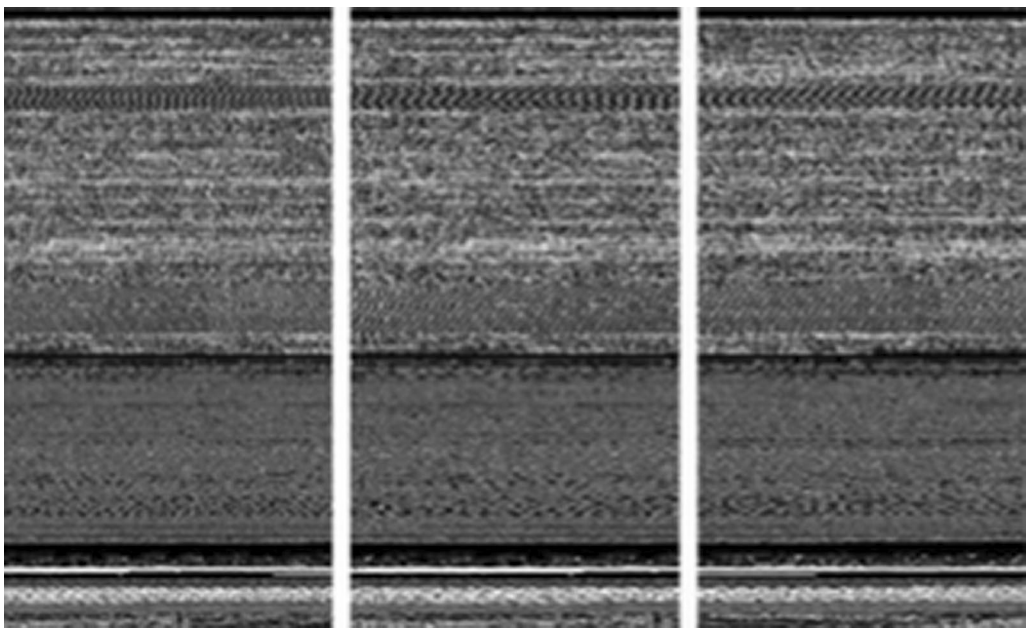

Fig. 1 Related malware families

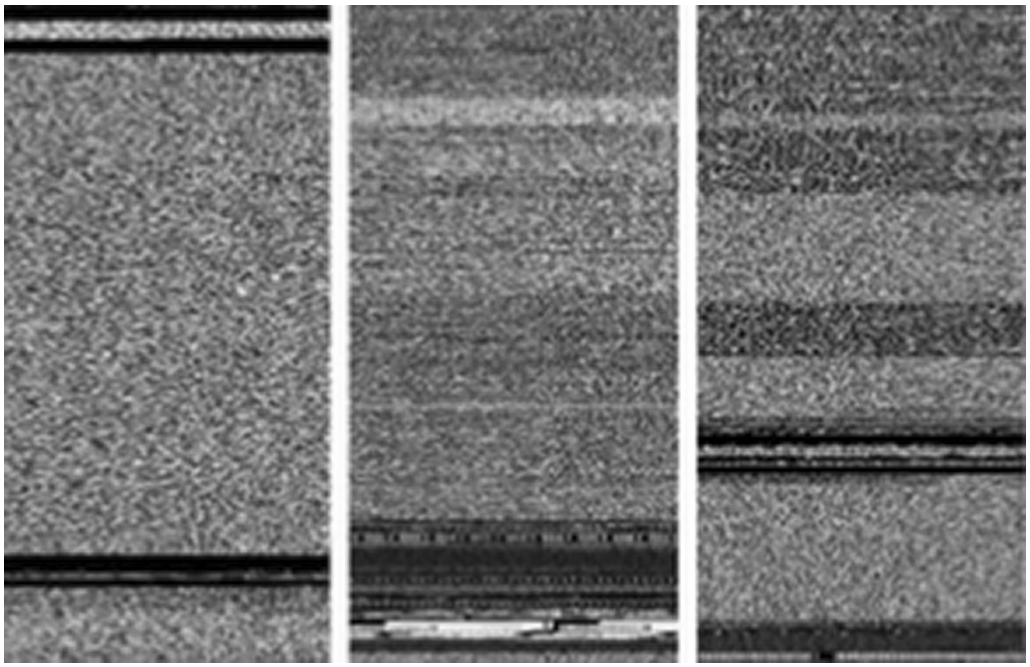

Fig. 2 Unrelated malware families

Convolutional Neural Networks (CNN). The key contributions of this paper are as follows:

- We propose a novel approach for malware detection using image visualization, specifically targeting fileless malware and low-rate attacks. Existing studies in image visualization-based malware detection have primarily focused on file-based malware detection and classification. However, the application of such visualization techniques for detecting evasive traces generated by stealthy fileless malware or low-rate attacks (Khorsandroo and Tosun 2018, 2019) has not been extensively explored. Notably, at the time of writing this paper, no published work has addressed the classification of Cobalt Strike beacons using machine learning and image visualization.
- We explored the viability of employing a Convolutional Neural Network (CNN) for an image-based network traffic classification to effectively identify elusive traces of fileless malware. To obtain a realistic network traffic payload, we utilized Cobalt Strike (Rahman 2021), a commercial adversary simulation software. It is worth noting that while penetration testing tools like Mimikatz, Cobalt Strike, and Metasploit are designed for assessing the security of IT networks from a red teaming perspective, Cobalt Strike, in particular, has gained notoriety for its utilization by threat actors.
- The objective of our approach is twofold: firstly, to address the relatively under-explored challenge of classifying Cobalt Strike beacons, and secondly, to efficiently identify this form of fileless malware within a vast dataset. Unlike previous works conducted in laboratory settings, our experiment was conducted in a production-level environment, ensuring the representativeness of the dataset in relation to real-world malware (Yadav and Tokekar 2021). In this experiment, we assess the performance of our model and compare it to prior techniques, taking into account the varying behaviors exhibited by Cobalt Strike beacons. We convert the collected Cobalt Strike beacon payloads into grayscale images and employ image transformation techniques to enhance their effectiveness. These generated images are then utilized to train a CNN model, enabling the classification of network traffic as either benign or malicious. Our results demonstrate that our solution surpasses existing methods in accurately detecting stealthy malicious traffic associated with fileless malware.

The remainder of the paper is organized as follows. "Related work" Section presents a detailed literature review on image-based malware classification and fileless malware detection mechanisms. This provides a foundation for understanding the existing research in the field. Building upon the literature review, "Types of use cases" Section explores the various use cases of

image-based network traffic classification. By examining different scenarios and applications, we demonstrate the practical relevance of this approach. In "Image representation methods" Section, we delve into the investigation of image representation methods. This analysis allows us to identify the most effective techniques for representing images in the context of malware detection. "Methodology: from network traffic to image classifier" Section elaborates on the proposed methodology we used to visualize Cobalt Strike beacon payloads and automatically classify them using images as input. We explain the steps and techniques involved in this process, highlighting the unique aspects of our approach. To support the visualization and classification process, "Proposed CNN architecture" Section provides details on the designed Convolutional Neural Network (CNN). We describe the architecture and components of the CNN model, highlighting its suitability for our purposes. Moving forward, in "Experimental setup and evaluation" Section, we discuss our experimental setup and the evaluation of the proposed approach. We present the data used, the metrics employed, and the results obtained, providing a comprehensive assessment of our methodology. Finally, in "Conclusion" Section, we conclude the paper and discuss future directions. We summarize the key findings and contributions of our work and highlight potential avenues for further research in the field of image-based network traffic classification and fileless malware detection.

Related work

Numerous visualization techniques relying on images have been suggested in academic studies for the purpose of facilitating the analysis of network traffic classification. This section presents a summary of image-based approaches to network traffic classification, employing various machine learning algorithms. Additionally, it explores mechanisms designed for the detection of fileless malware.

Overview of image-based network traffic classification

Nataraj et al. (2011) were among the early adopters of the malware visualization technique for the purpose of classifying diverse malware families. The method proposed by the authors involves converting malware binary data into 2D grayscale images. They noted that malware variants belonging to the same family exhibit similarities in image texture. Initially, the authors extracted GIST features to classify malware, followed by the application of K-Nearest Neighbors (KNN) algorithm using Euclidean distance. KNN is a supervised learning technique commonly used for classification and regression problems. In KNN, unclassified samples are assigned to a specific class based on a majority vote from their neighboring samples (Shabtai et al. 2009). If a significant proportion of the k nearest neighbors belong to a particular class, the sample will be assigned to that class (Xie et al. 2018). In their extensive analysis, the authors utilized a dataset of 9458 malware samples, encompassing 25 malware families, and achieved a classification accuracy of 97.18%.

Dai et al. (2018) proposed a technique involving extracting grayscale images from malware memory dump files. They employed K-Nearest Neighbors (KNN), Random Forest (RF), and Multi-Layer Perceptron (MLP) for malware classification. RF, being an ensemble model, generates a response variable based on results from different decision tree models and has proven effective in solving classification and regression problems (Xie et al. 2018). When running a random forest, it is important to specify parameters such as training data, response variables, the number of trees, predictor variables, error calculation parameters, and other relevant factors. Furthermore, MLP (Taud and Mas 2018), belonging to the class of powerful *Artificial Neural Networks (ANNs)*, is widely used. It employs supervised training to generate a nonlinear model capable of predicting output based on input data. To extract image features, the authors employed histograms of gradients and followed three key steps for classifying memory dumps. Initially, they extracted runtime malware memory dump files using a sandbox. Next, they converted the memory dump into grayscale images and extracted feature vectors using bi-cubic interpolation. To validate their approach, the authors conducted experiments using actual malware samples, demonstrating the effectiveness of their methodology.

Gibert et al. (2019) introduced a Convolutional Neural Network(CNN) to the Nataraj et al. (2011) approach to classify assembly language within portable executable (PE) files. A Convolutional Neural Network is part of deep learning whose connectivity pattern is inspired by mammals' visual cortex structure (Hubel and Wiesel 1968). Without human supervision, CNN can process an image's high dimensionality. CNN accepts the raw pixels of an image as input and performs feature extraction. By transforming the input image, CNN can learn important objects constituted within. In their work, the authors represented malware binary content as a grayscale image to find a pattern and classify malware into families. By deriving features such as local and invariant out of the image, they inferred that the presence of malware patterns could be detected.

Ni et al. (2018) proposed a malware family classification technique that combines image-based malware visualization and Convolutional Neural Network. In this work, the authors extracted similar hash values of similar malware codes using the locality-sensitive hashing technique. In this regard, similar Hash values produce

a similar grayscale image. The performance of the proposed algorithm was improved by considering additional methods such as multi-hash, major block selection, and bilinear interpolation. Finally, Convolutional Neural Networks were used to identify which family the malware belongs to.

Zhang et al. (2019) Converted network flow data to a grayscale image to classify encrypted network traffic. The authors combined Long Short-Term Memory (LSTM) and Convolutional Neural Network(CNN) to extract statistical features. As a class of recurrent neural networks (RNN), long-short-term memory networks (LSTM) are capable of dealing with long-term dependency (Liu et al. 2018). Along with the hidden state, LSTM has an additional information flow path compared to other forms of RNN. Input, output, and forget gates are the three components of Cell which is the additional processor introduced in LSTM. The Cell controls the flow of information and decides which information needs to be dropped by going through the network (Li et al. 2021). This property of Cell helps LSTM to store information that is acquired from previous steps. In their work, the authors apply the LSTM model to extract the representation vectors of the images. On the other hand, to extract representation features and classify the network CNN algorithm is used.

Fileless malware detection mechanisms

van der Eijk and Schuijt (2020) developed an algorithm to detect the presence of a Cobalt Strike Command and Control (C2) beacon in a network communication based on NetFlow data. The authors established a network topology that consists of a Cobalt Strike C2 server, domain redirection, and target infrastructure. And configure a NetFlow that helps them distinguish between malicious and benign data. They analyzed the characteristics of the regular HTTP and HTTPS network connection of Cobalt Strike. For their experiment, they used a static algorithm.

Hendler et al. (2018) proposed a Deep Neural Networks-based fileless malware detection method. In this study, the authors implemented several detectors for identifying malicious PowerShell commands. The evaluation of their detection model demonstrated high recall values and an exceptionally low false positive rate. The authors proposed that combining Natural Language Processing (NLP) techniques with Convolutional Neural Networks (CNNs) resulted in the optimal performance for this task.

Authors in Bucevschi et al. (2019) proposed a machine learning based fileless attack prevention. They tested the presence of an anomaly in command lines such as PowerShell scripts, Windows Management Instrumentation (WMI) scripts, Windows tasks, shortcut files (LNK), and Batch scripts. In this paper, the authors use One Side Class perception(OSC) which is a modified version of the Perceptron algorithm. By providing an extra stage that will maintain a small false positive rate, OSC ensures the correctness of classification in the training phase. To validate their proposed model, the authors utilized a dataset consisting of a total of 37,546 samples that were labeled as both anomaly and anomaly-free. In their experimental evaluation, the authors achieved a detection rate of 83.32%.

Rigaki and Garcia (2018) used Generative Adversarial Networks(GANs) to mimic network traffic. GAN generates network traffic, and the malware source codes were modified to accept parameters from GAN. GAN allows them to mimic legitimate application traffic and bypass malware detection. Real-life experiments were conducted on Facebook chat traffic to train it with GAN. The authors successfully mimic Facebook's chat by modifying the behavioral patterns of real-world malware samples. In this work, the authors applied deep learning to create malware samples. Additionally, they evaluated the effectiveness of these enhanced malware samples in evading detection by a machine learning-based malicious traffic detection system known as Stratosphere Linux IPS (slips). Their experiment revealed that approximately 63.42% of the malicious samples successfully bypassed the detection capabilities of the Stratosphere Linux IPS system.

The presence of malicious PowerShell was tested by FireEye (Fang 2018). The author used the Natural Language Processing (NLP) pipeline. The key to the proposed NLP module are Decoder, Named Entity Recognition (NER), Tokenizer, stemmer, Vocabulary Vectorizer, Supervised classifier(Kernel SVM, Gradient Boosted Trees, Deep Neural Networks), and Reasoning. Any encoded text within the PowerShell script will be detected and decoded. Then the PowerShell command will be tokenized to create a list of tokens. All semantically identical tokens will go through the stemming algorithm to reduce them to the original word form. Prior to inputting the token list into the machine learning algorithm, a vectorization process is applied to ensure a machine learning-friendly format. This process transforms the tokens into a suitable representation. Subsequently, the supervised model executes the prediction.

Borana et al. (2021) proposed an assistive tool to detect fileless malware. This tool will perform a forensic examination to identify abnormal processes and abnormal activities on the system and network. The authors also discussed fileless malware life cycles along with their infection strategies. In their study, Handaya et al. (2020) suggested three machine learning algorithms(KNN, SVM, Random Forest) to detect fileless cryptocurrency

mining malware accurately. The authors recommended that researchers use the EMBER dataset that contains more than 1 million SHA-256 hashes collected from portable executable (PE) files. This dataset contains 900 K samples for training and 200 K samples for testing.

Types of use cases

Image-based traffic classification holds potential for various applications in network traffic analytics. The existing body of literature primarily emphasizes two key use cases: malware detection and traffic engineering.

Internet traffic classification

Internet traffic classification encompasses the classification of network traffic into different traffic classes, utilizing a range of features (Dhote et al. 2015). Two main approaches have been employed for network traffic classification: rule-based and machine learning-based methods. Rule-based methods involve techniques such as port-based and payload-based approaches. Conversely, machine learning-based methods encompass statistical-based and behavioral-based approaches, as depicted in Fig. 3.

Rule-based approaches

In this approach, packets that enter the network will be classified according to their predefined hard-coded rules (Lim et al. 2019; Wang et al. 2017). This method is one of the traditional classification methods that suffer from dynamic ports, and encrypted applications (Zhang et al. 2012). Thus, the following two approaches are frequently employed.

(i) *Port-Based Approaches:* As one of the traditional classification methods, port-based methods rely mainly on port numbers, and it uses only information from the packet's header (Dhote et al. 2015; Tahaei et al. 2020). This method was successful because many application hosts use a well-known port to communicate with other hosts. In early times, it was easier to find a packet's target port number as most applications will have their port numbers registered at the Internet Assigned Numbers Authority (IANA). Then using the classifier, the port numbers will be associated with the application. For example, port number 53 is associated with DNS, and port 80 is for HTTP traffic. Even though this method is fast and has a simple implementation, it fails to detect correctly if a fake port number is in the traffic (Tahaei et al. 2020). In addition, the existence of dynamic port allocation in recent applications which are private ports and are not available in the IANA's list, and encryption of packet header of the IP layer which creates obfuscation in TCP or UDP port number, makes port-based approaches obsolete (Barut et al. 2020).

(ii) *Payload-Based Approaches:* To address the limitations of port-based approaches, payload or deep packet inspection (DPI) has been utilized. With this method, network traffic is classified by examining both the packet header and the payload information from the application layer (Lim et al. 2019;

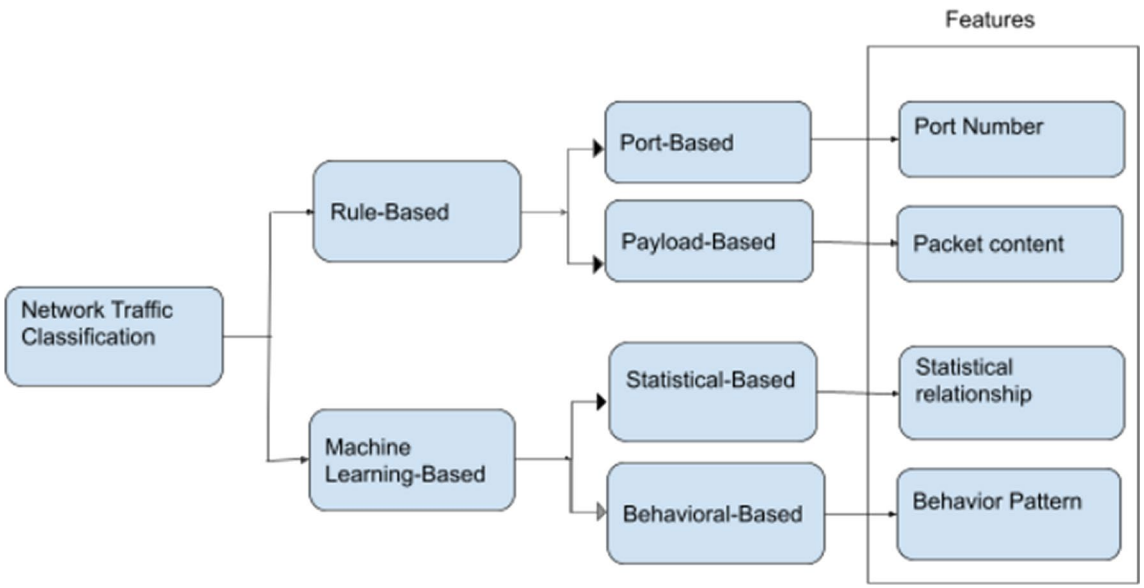

Fig. 3 Internet traffic classification approaches

Tahaei et al. 2020), and matching them to stored signatures (Lim et al. 2019). In this approach, the payloads are scrutinized bit by bit to identify a match for a predetermined byte sequence. Then stored signatures are compared with the matched bit stream and classification will be performed accordingly (Dhote et al. 2015). This method solves the problem of port number dependency (Lim et al. 2019) and performs the network application classification accurately. Despite its advantage, this method fails to classify correctly if signatures are not up to date or if the payload is encrypted (Barut et al. 2020). Additionally, DPI can have high computational requirements, which can cause delays in network traffic (Dhote et al. 2015; Tahaei et al. 2020). Stochastic packet inspection (SPI) was proposed as a solution to the limitations of payload-based classification methods. SPI uses statistical methods to analyze the traffic flow of packets rather than inspecting the payload itself. SPI captures statistical features of the traffic flow such as packet arrival times, packet size, inter-packet time, and direction of the flow (Zhao et al. 2021). These statistical features are then used to create models for different traffic classes. SPI is more resilient to encrypted traffic and can provide a more accurate classification than payload-based methods (Tahaei et al. 2020; Zhao et al. 2021). Moreover, this method exposes user data privacy as the packet's content is inspected thoroughly (Tahaei et al. 2020).

Machine learning-based approaches

The limitations of previous techniques directed researchers to apply machine learning approaches that do not depend only on the port number or payload (Barut et al. 2020). Accordingly, the following two approaches are commonly used.

(i) *Statistical-Based Approaches:* This approach uses flow-level properties such as flow duration, flow idle time, packet inter-arrival time, and packet length with the assumption that traffic at the network or transport layers will be unique for certain classes of applications (Dhote et al. 2015; Nguyen and Armitage 2008; Tahaei et al. 2020). These methods solve the problem of payload-based approaches as they avoid content inspection (Tahaei et al. 2020).

(ii) *Behavioral-Based Approaches:* In this approach, flow level, packet level, and connection level data were used in order to check a host's behavior. Packet header fields such as IP address, port number, and protocol type play the main role in behavior-based classification to identify an application behavior in a host (Zhao et al. 2021).

Image representation methods

Data preprocessing plays a vital role in preparing raw traffic data for classification and detection algorithms. Its objective is to convert the raw data into a suitable format that machine learning algorithms can effectively utilize. One common approach is to convert the data into an image format, which can be fed into machine learning models for classification. Representing traffic data as an image enables the extraction of significant features and patterns that assist in the identification of different traffic classes and the detection of anomalies (Nataraj et al. 2011; Naeem et al. 2019).

Network flow to image conversion

In line with existing literature, raw data will be captured as packet capture files (PCAP) or binary files (BIN). The collected network traces will go through three stages: *traffic splitting*, *traffic sanitizing*, and *outlier removal*. Traffic splitting categorizes the captured traffic into different representations, such as flow based on header fields, layer seven information, connection sessions with header fields, or connection sessions with only layer seven information. Flows based on session information are typically stored as PCAP files, while flows based on layer seven information are saved in BIN format.

During sanitization, Media Access Control(MAC) and Internet Protocol(IP) addresses in the data link and network layers are randomized, removing identical, duplicated, and empty files without altering the data format. This process helps address biases during machine learning model training.

Outliers, files abnormally larger or smaller than the rest, are removed before generating image files from the captured traffic. To ensure uniformity, the input data size (images from network traffic) is adjusted by trimming or padding. Trimming reduces the file size to the desired length, while padding adds 0×00 to smaller files. The preprocessed data, now uniform in size, is transformed into grayscale images by representing each byte as a pixel in the image. A cutoff size is chosen to ensure equal length and width, e.g., a 28×28 pixel grayscale image corresponds to a cutoff size of 784 bytes. Figure 4 illustrates the general concept of mapping a byte array to a grayscale image.

Binary code to image conversion

There are research works (e.g., Nataraj et al. 2011; Ni et al. 2018; Naeem et al. 2019; Su et al. 2018) that

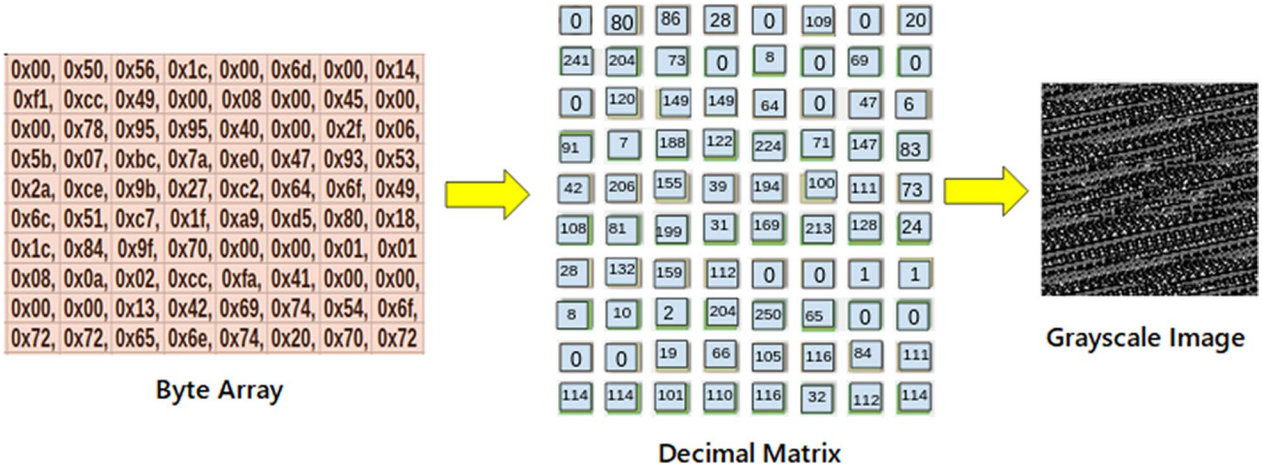

Fig. 4 Byte array to grayscale image conversion

discuss techniques for converting raw binary code into an 8-bit vector and generating a grayscale image as shown in Fig. 5. The grayscale image represents the binary code as a 2-dimensional array, where each pixel has an 8-bit unsigned integer value ranging from 0 to 255. Black is represented by 0, and white is represented by 255. Each pixel in the grayscale image represents intensity information, ranging from 0 to 255 (Kumar et al. 2016). In the conversion process, the binary bit string is divided into 8-bit substrings, which are then converted to decimal numbers. The conversion is done by summing the binary digits multiplied by powers of 2. For example, a binary number $B = (b_{n-1}...b_4b_3b_2b_1b_0)$ can be converted to a decimal number $D = (b_0 * 2^0 + b_1 * 2^1 + b_2 * 2^2 + b_3 * 2^3 + b_4 * 2^4...)$). As Fig. 6 illustrates, a bit string such as $B = 0110000010101100$ can be split into two substrings of $B1 = 01100000$ and $B2 = 10101100$. Then, B1 and B2 can be converted further to decimal numbers (that is, $B1 = 01100000 \rightarrow 96$ and $B2 = 10101100 \rightarrow 172$). The resulting decimal numbers form a 1D vector representing the intensity of each pixel (Kancherla and Mukkamala 2013). The 1-dimensional array can be transformed into a 2-dimensional matrix based on the preferred width. Since there are varying input samples, the grayscale output image will have distinct widths, and heights (Gibert et al. 2019).

Binary to color code mapping

A recent work (Chukka and Devi 2021) has demonstrated the ability to better capture patterns within malware instruction using opcode as a color-coded pixel. The binary-to-color code mapping process involves five steps: collecting opcodes from all code sections, identifying unique opcodes, mapping each unique opcode to a distinct color, arranging opcodes in a two-dimensional image grid, and replacing the opcodes on the grid with the corresponding color codes from the mapping. According to Chukka and Devi (2021), using grayscale images has limited effectiveness in identifying behavioral patterns in binaries. Grayscale representation of raw binary files becomes noisy due to the varying sections in Portable Executable (PE) binaries (e.g.,.text,.data).

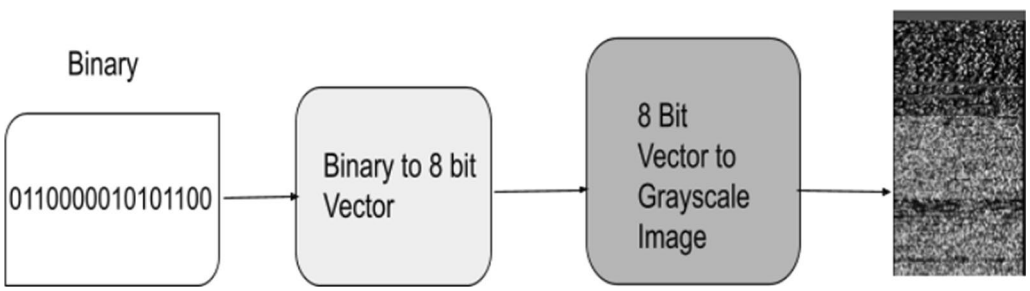

Fig. 5 Binary to grayscale image

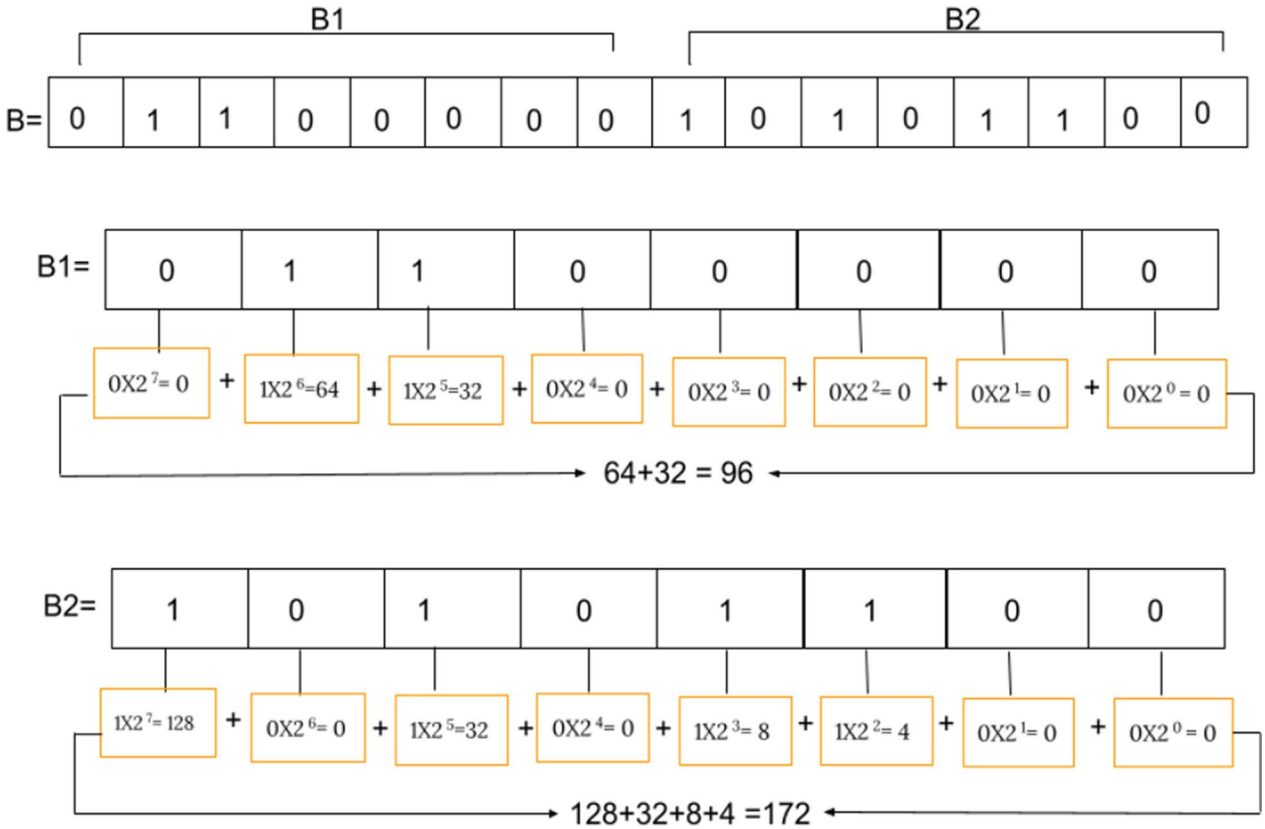

Fig. 6 Binary to decimal conversion

Binary to RGB image

RGB color format uses three colors (Red, Green, Blue) represented by 8 bits each, totaling 24 bits to define a color out of millions of possibilities. A recent technique for malware detection involves converting raw binary files into RGB images. In RGB images, each pixel represents three sequential bytes, allowing for capturing more bytes in each row and ensuring consistent byte-level alignment. This approach facilitates placing more information in each row, making visual similarities between similar samples more apparent and easily identifiable. Unlike grayscale-based approaches, RGB-based encoding enables more compact images by reducing pixel space with a 1/3 ratio, resulting in less distortion during post-image resizing. However, this approach may have drawbacks when byte-level variations become prominent in raw binary data, compared to 8-bit grayscale encoding (Bozkir et al. 2019, 2021).

Non-image data into an image conversion

The authors in Sharma et al. (2019) introduced the concept of DeepInsight, which transforms non-image data into an image through three main steps. Firstly, non-linear dimensionality reduction techniques like kernel PCA or t-SNE are applied to map features into a lower-dimensional space, typically two-dimensional. Secondly, the smallest rectangle box encompassing all points is determined using the convex hull algorithm, and the image is rotated to fit horizontally or vertically. Lastly, the pixels are framed and mapped to obtain the final pixel coordinates. In DeepInsight, pixels that do not contribute to representing features are left blank, particularly in the presence of outliers, which can sometimes account for significant portions of the image. Additionally, compared to similar techniques such as IGTD (Zhu et al. 2021), DeepInsight produces larger images, necessitating more memory and longer training times.

Symbolic data conversion

In datasets containing both numeric and symbolic data types, symbolic features like protocol types (e.g., TCP, UDP, ICMP) need to be converted into numeric form to be used in machine learning models. One-hot encoding is a common technique for this purpose. It represents categorical variables as binary vectors. In the context of network traffic analysis, one-hot encoding can be applied

to represent the byte values of the traffic. Each distinct category is mapped to an integer value, and then each integer value is represented as a binary vector. Entries corresponding to categories not represented by the vector are set to zero, while the entry representing the category is set to one. Thus, the one-hot encoder takes an input value of m integers and outputs an m-sized binary vector. For example, to represent the byte value '0', a 256-sized vector is constructed with the first entry set to one and the others set to zero. This transformation replaces single values with vectors and converts the original 1D vector of integers into a 2D binary array (Krupski et al. 2021).

Methodology: from network traffic to image classifier

This section elaborates on how the captured network traffic will be converted into an image. It then discusses the architecture of the Convolutional Neural Network (CNN), which takes the produced images as input.

Image generation: implementation

We perform image conversion based on the payload collected from one of the prominent attack toolkits used by malicious actors, namely Cobalt Strike (Seazzu 2016). Cobalt Strike is a threat-emulating proprietary software suite that emulates embedded actor beacons in a network. A Cobalt Strike beacon is a fileless malware attack. Fileless malware (Kumar 2020) is an evasive malware that does not rely on files. Instead, it exploits vulnerabilities on legitimate, trusted, and widely used applications to infect a system (Smelcer 2017). When fileless malware is in action, it performs malicious activity using native tools built into a system to steal data, interfere with operations, or use compute resources. Moreover, traditional methods of detecting malware will no longer work to detect these threats since they cannot leave any traces behind. It is why fileless malware attacks have 10 times higher success rate than traditional file-based attacks (Sanjay et al. 2018). The use of beacons within network traffic is a common method for communicating with external servers and emulating malicious commands. Cobalt Strike beacons, in particular, are known for their ability to blend in with legitimate traffic due to their communication flexibility, making them difficult to detect (Seazzu 2016). Apart from spear-phishing, Cobalt Strike is capable of mimicking malware and other sophisticated threat techniques to obtain unauthorized access to systems (Rahman 2021).

The Cobalt Strike beacon payload employs asynchronous communication patterns characterized as "low and slow." For beacon communication, Cobalt Strike uses three alternative transport mechanisms, such as HTTP, HTTP(s), and DNS. In this experiment, we used an HTTP request header. Through an HTTP GET or POST request, Cobalt Strike's beacon payload retrieves tasks from its team server. The size of the traffic flow for HTTP requests generated by the Cobalt Strike beacon depends and influenced by a variety of factors. These factors include the type and size of the requested resource, the HTTP headers that are sent, and the particular configuration of the beacon payload being used. By default, to complete most task packages in a single request, the payload limits its data usage to 1 MB of encrypted data per request. Beacon does not make HTTP requests in parallel; instead, it sends one request and waits for its response. If any data that is intended for the team server is beyond the limit, the beacon chunker will divide it into 100-byte chunks (Mudge 2019). Every component sends a separate HTTP request back to the team server. We used a Cobalt Strike payload with a base64 encoded HTTP request header to implement image conversion. The process starts with decoding the payloads into byte format. Since the payload length is very small, the image created will also have a small size. To have an image that can be readily visible, we choose an image dimension of 64×64 pixels, an image in a perfect square shape. The input payload will be resized into the next perfect square by applying the zero-padding technique. Then, bytes are converted to their corresponding ASCII values and create a one-dimensional (1D) array. The generated 1D array will then be fed into the Python NumPy (Bressert 2012) Reshape module to be converted to a two-dimensional (2D) array. Finally, OpenCv (Culjak et al. 2012) interpolation transformation is used to resize the images to the model's desired size. If the image has a smaller size than expected, it will be enlarged. It may also be shrunk if it exceeds the preferred image dimension of 64×64 pixels. Figure 7 depicts the process mentioned above. In contrast, Algorithm 1 presents the pseudo-code used to convert incoming network traffic payload to a grayscale image which is fed into the CNN. The network traffic payload is enhanced to fit the length and width requirement of the CNN algorithm using standard interpolation techniques.

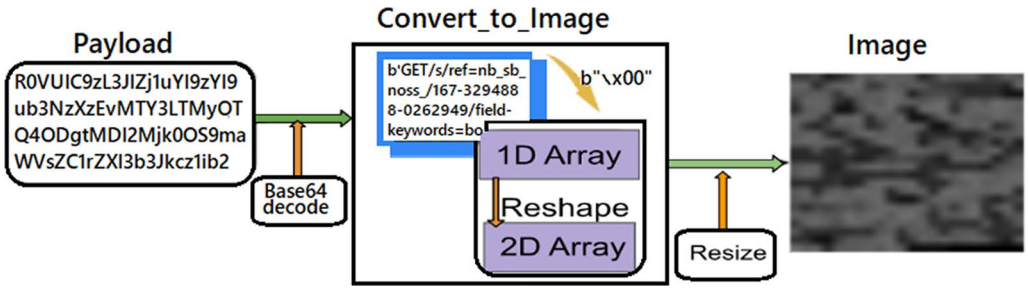

Fig. 7 Turning payload into grayscale image

Algorithm 1 Payload to Image Conversion

procedure ENHANCE(Img, L, W)
  if $Img.length < L$ **then**
    $Enhance \leftarrow resize(Img, L, W, i = INTER\_AREA)$
  else
    $Enhance \leftarrow resize(Img, L, W, i = INTER\_LINEAR)$
  end if
  return $Enhanced$
end procedure
procedure MAKESQUARE(Payload)
  l,$w \leftarrow \sqrt{Payload.size}$
  $Payload.append(x00 * l * w - payload.size)$
  return $Payload, l, w$
end procedure
procedure CONVERTTOIMAGE(RawPayload, L, W)
  $Payload \leftarrow based64decode(RawPayload)$
  $Payload, l, w \leftarrow MAKESQUARE(Payload)$
  $1D\_Array \leftarrow readBuffer(Payload)$
  $Image \leftarrow reshape(1D\_Array, l, w)$
  $Enhanced\_Image \leftarrow ENHANCE(Image, L, W)$
  return $Enhanced\_Image$
end procedure

Proposed CNN architecture

In this section, we will elaborate on the CNN architecture used for our model.

A convolutional neural network(CNN) is part of deep learning whose connectivity pattern is inspired by mammals' visual cortex structure (Hubel and Wiesel 1968). Without any human supervision, CNN is capable of processing an image's high dimensionality. CNN was originally created for image classification and can accept the raw pixels of an image as input and perform feature extraction. By transforming the input image, CNN can learn about important objects contained within the image. CNN is selected for this malware image classification experiment because of its ability to learn spatial hierarchies of features from images and handle the high dimensional input space.

Traditional neural networks like Support Vector Machines (SVM) and Random Forests are inefficient at processing images as they require feature engineering and assume that each input feature is independent. However, in an image, each pixel is correlated with its neighboring pixels, which results in a high dimensional input space. CNNs solve this problem by using convolutional layers that can extract relevant features from the input image by sliding a filter over it and learning a set of weights. The outputs of these convolutional layers are then passed through pooling layers, which reduce the dimensionality of the output while retaining the important features. Moreover, CNNs can also learn local invariances in translations and rotations. Robustness to variation is essential for image classification tasks where the object of interest can appear in different parts of the image. CNN achieves this by using shared weights for the filters in convolutional layers, which allows the network to recognize the same feature in different parts of the image.

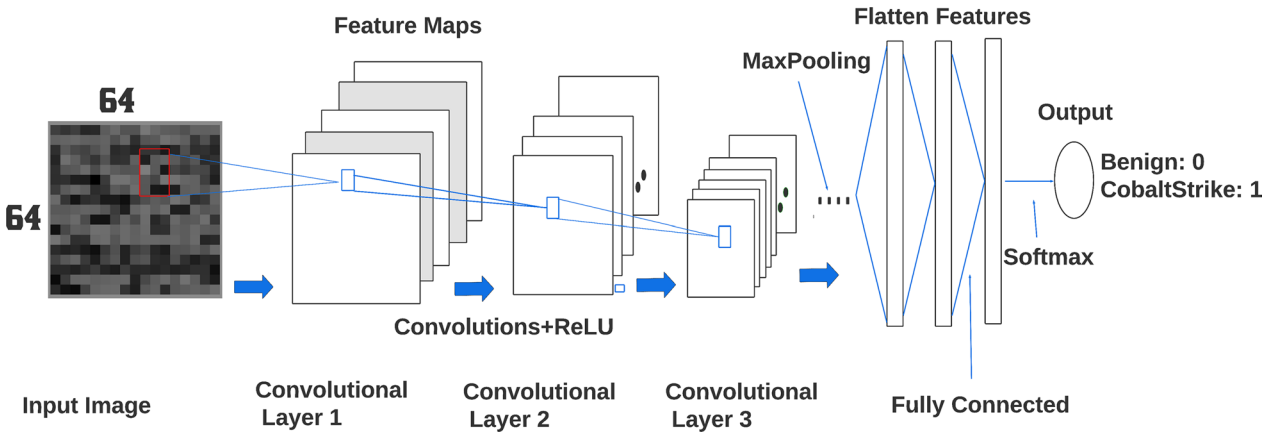

Fig. 8 Proposed CNN architecture

In our experiments, once the payload is transformed into an image, it is processed further using the CNN architecture. The convolutional layer produces feature maps proportional to the input image. As Fig. 8 shows, the proposed CNN model has three convolutional layers. A two-dimensional (2D) convolution layer is followed by a Rectified Linear Unit (ReLU) activation layer, followed by batch normalization and a max pooling layer. CNN-based models are proven to be effective at analyzing images. We opt for Convolutional Neural Networks to classify sessions because they possess the ability to learn directly from raw bytes in network traffic, eliminating the requirement for feature extraction. This is important for detecting fileless malware, as it does not leave traditional artifacts on disk that can be used for detection. By representing the data in an image format, it becomes easier to identify patterns and anomalies that may indicate the presence of malware. We chose a dense neural network to classify malicious and benign sessions based on the features learned by the convolutional layer from images.

During the training phase, to prevent overfitting, we incorporated batch normalization. We further utilized a two-dimensional (2D) Maxpooling layer to downsample the images. Additionally, we applied the Adam algorithm (Kingma and Ba 2014) and tuned various parameters to achieve a true positive rate of over 95% and a false positive rate of less than 0.01%. These included the distribution of benign and malicious sessions in the training dataset, the number of CNN and dense layers, types of optimizers, learning rate, dropout rate, number of filters, size of filters, and the number of training epochs. We closely monitored the loss and accuracy of our model on the validation dataset during the training process to determine the optimal parameters. By utilizing the Adam algorithm, we were able to identify the best hyperparameters that produced the most optimal validation results. We selected a batch size of 32 that proved to work best in our case. We flatten the outputs of the last convolutional module and pass them through a couple of fully connected layers. At the same time, we applied regularization through dropout modules between these layers. We set the dropout probability equal to 0.4.

Table 1 Summary of the developed model

Layer (type)	Output shape	Param #
Conv2d (Conv2D)	(None, 61, 61, 256)	4352
Conv2d_1 (Conv2D)	(None, 58, 58, 512)	2097664
Batch_normalization(BatchNo)	(None, 58, 58, 512)	2048
Conv2d_2 (Conv2D)	(None, 55, 55, 256)	2097408
Max_pooling2d (MaxPooling2D)	(None, 27, 27, 256)	0
Flatten (Flatten)	(None, 186624)	0
Dense (Dense)	(None, 256)	4777600
Dropout (Dropout)	(None, 256)	0
Dense_1 (Dense)	(None, 64)	16448
Dense_2 (Dense)	(None, 2)	130
Tota params: 51,994,050		
Trainable params: 51,993,026		
Non-trainable params: 1024		

The final layer of our model employed softmax to transform the logits generated by the dense layer into probability distributions. During the training process, the cross-entropy loss function was utilized to minimize errors. The loss value obtained was assessed on a scale ranging from 0 to 1, where a value of 0 indicated a perfect model with no errors. Table 1 shows a summary of the developed model. This CNN model managed a total of 51,993,026 parameters.

Experimental setup and evaluation

Experimental setup

To implement the proposed CNN architecture, we used TensorFlow (Abadi et al. 2016), and Ketkar (2017), which will help the model extract high-level features from the images. Our experiments are executed on Palo Alto private Cloud. The virtual machines (VMs) used for this experiment run Ubuntu 18.04 LTS with 32 GB of RAM with dedicated GPU support.

Dataset

Our experimental analysis is based on the Cobalt Strike beacon payload collected from Palo Alto Networks' research infrastructure. We acquired a benign dataset of network traffic from an enterprise network, comprising over 3.6 million sessions collected over the span of a week. We labeled the sessions as benign by running them through VirusTotal, and Palo Alto Networks products. Additionally, we obtained a dataset of malicious traffic by simulating command and control communications of Cobalt Strike exploit kits. The Cobalt Strike payloads, referred to as beacons, were created to communicate with its team server using malleable profiles from a public GitHub repository (Mudge 2018), and additional malleable profiles collected by the Palo Alto Networks. In total, we created 6,271 malicious sessions. All the benign and malicious sessions were stored as PCAP files.

Due to skewed class proportions in the dataset, it was challenging for the model to work and generalize well. To mitigate this, we employed resampling techniques by downsampling the benign dataset and upsampling the malicious dataset. Specifically, we used undersampling to reduce the size of the benign dataset by applying deduplication on the hostname and URI (Uniform Resource Identifier) path. We clustered the sessions with the same hostname and URI path in the request header of the first packet in the session and randomly selected one session from each group. After deduplication, we were left with 567,620 sessions in the benign dataset.

To upsample the malicious dataset, we shuffled the arrangement of the header fields in an HTTP packet. This was possible because shuffling the structure of the header fields in HTTP payloads does not change the functionality of the HTTP request. However, from our preliminary experiments on model selection, we observed that the model learned to classify benign and malicious sessions based on the structure of the header fields, resulting in false positives based on the similarity in the header structure. Since each header had more than five header fields, we could generate 120 sessions with different permutations of the header fields. After running a few experiments, we found that upsampling the malicious sessions by a factor of 30 achieved the best results and a benign-to-malicious dataset ratio of 75:25. Following upsampling, the size of our malicious dataset was expanded to 188,130 samples. After model optimization, we performed our testing using new unknown samples to our model to find out testing accuracy. Running our model on a dataset it has never seen before will help our model to avoid overfitting and obtain an unbiased assessment of the performance of our model. We used a total of 865,206 (859,885 benign and 5321 malicious) samples for testing. Table 2 summarizes the total dataset used for training and testing.

Table 2 Summary of dataset

	Training set	Testing set
No. of benign samples	567,620	859,885
No. of malicious samples	188,130	5321

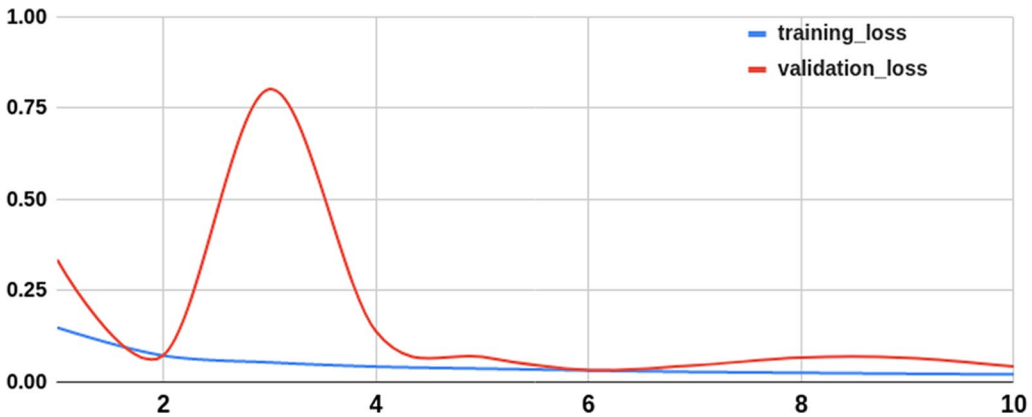

Fig. 9 Training loss versus validation loss

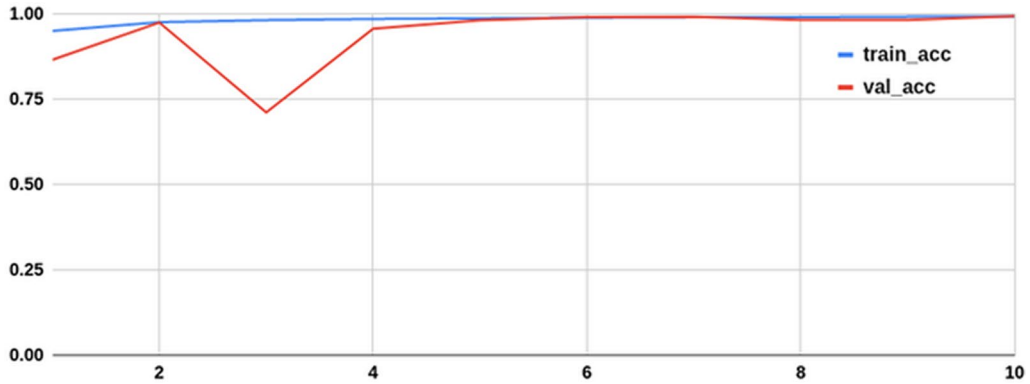

Fig. 10 Training accuracy versus validation accuracy

Table 3 Training and validation accuracy for 10 epochs

Epoch	1	2	3	4	5	6	7	8	9	10
Training accuracy	95.14	97.70	98.30	98.65	98.83	99.00	99.14	99.17	99.31	99.34
Validation accuracy	86.66	97.61	71.27	95.80	98.30	99.20	99.25	98.83	98.30	99.48

Evaluation

We evaluated the model generated from the training pipeline on the non-overlapping sets of the malicious dataset obtained from the wild to measure the true positive rate of the model. The false positive rate was measured by testing the model in the sessions we collected by crawling the top Alexa websites. It was computationally expensive for us to run multiple iterations of cross-validation, as our model had 51 million parameters and more than 800,000 samples. However, during training and validation, we split our dataset at different random seeds, and we observed consistent results. We eventually fixed the random seed for final model to make our results reproducible. Figure 9 presents how loss variation was optimized. From the graph, training loss and validation loss follow each other, which shows that the model is not overfitted. Furthermore, Fig. 10 illustrates the disparity between the achieved training accuracy and validation accuracy. The model's overall classification accuracy is 99.48%. Table 3 presents the training and validation accuracies attained by the model across 10 epochs.

Performance metrics in our experimentation include: *i) True Positive (TP)* to present the malicious data which is correctly classified as malicious), *ii) True Negative (TN)* to manifest the benign data which is correctly classified as benign), *iii) False Negative (FN)* to show the malicious data which is classified as benign, and *iv) False Positive (FP)* to express benign data which has been classified as malicious.

A confusion matrix is known to work well with binary classification. Therefore, the true positive rate (TPR), which describes how accurately the model classifies, can also be described in terms of a confusion matrix. The true positive rate (TPR) can be calculated using the confusion matrix as shown in Eq. (1):

$$TPR = \frac{TP}{TP + FN} \tag{1}$$

Furthermore, the False Negative Rate (FNR) shows an incorrect negative classification and is defined in Eq. (2):

$$FNR = \frac{FN}{TP + FN} \tag{2}$$

On the other hand, the True Negative Rate (TNR) deduces the correct negative classification and is defined in Eq. (3):

$$TNR = \frac{TN}{TN + FP} \tag{3}$$

Finally, False Positive Rate (FPR) represents the ratio of incorrect classifications and is defined in Eq. (4):

Table 4 Evaluation metrics results

Type	Source	Total test data	Detected	TPR (%)	FNR (%)	TNR (%)	FPR (%)
Malicious	Wildfire	5148	4749	92.24	7.75	N/A	N/A
	APTs in the wild	29	28	96.55	3.44	N/A	N/A
	Customers	30	30	100	0	N/A	N/A
	Public profiles of cobalt strike	114	109	95.61	4.39	N/A	N/A
Benign	Top Alexa websites request headers	859,885	1214	N/A	N/A	99.85	0.14

$$FPR = \frac{FP}{TN + FP} \tag{4}$$

It is also noteworthy that we define our accuracy, precision, recall, and F1 score metric in terms of a confusion matrix. The four metrics are defined in Eqs. (5), (6), (7), and (8) respectively.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

$$Precision = \frac{TP}{TP + FP} \tag{6}$$

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{8}$$

Accuracy serves as a measure of the overall correctness of a model's predictions, whereas precision rate gauges the model's capability to accurately identify positive samples. Recall, on the other hand, signifies the model's proficiency in correctly identifying all positive samples. F1-score represents a harmonious balance between precision and recall, indicating an effective trade-off between the two metrics.

Discussion

The integration of machine learning and image visualization for fileless malware classification is a relatively recent field. Table 4 presents the detection rates achieved by our model on a test dataset. We collected a total of 5321 malicious samples from four different sources (Wildfire, APTs in the wild, Palo Alto Customers, and Public profiles of Cobalt Strike). As for the benign samples, we utilized the Top Alexa website request headers dataset, consisting of 859,885 benign samples. The model successfully identified and classified 4916 malicious samples as malicious, resulting in an overall average detection rate of 92.38%. Table 3 demonstrates that our approach exhibits improving training and validation accuracy with each epoch, ultimately achieving an average accuracy of 99.4% and 99.5% for the training and validation phases, respectively.

In addition, we conducted a comparison of our classification accuracy with that of existing approaches (i.e., van der Eijk and Schuijt 2020; Khalid et al. 2023), as shown in Table 5. Our experiments involved 193,451 fileless malware samples and resulted in an accuracy of 99.48%, precision of 80.2%, recall of 92.38%, and F1-score of 85.86%. As presented in Table 5, the research conducted by van der Eijk and Schuijt (2020) employed a dataset comprising only 17 Cobalt Strike beacons and achieved an accuracy of 99.99%. The precision, recall, and F1-score were reported as 75%, 88.2%, and 81% respectively. It is worth noting that van der Eijk and Schuijt (2020) utilized a deep packet inspection based static algorithm to detect Cobalt Strike beacons in network traffic. Furthermore, a small number of non-real-world malware samples were used in their experiment to identify features. In contrast, Khalid et al. (2023) used 26 fileless malware samples and achieved an accuracy of 93.3%, with a true positive rate (TPR) of 87.5% and a false positive rate (FPR) of 0%. Precision, recall, and F1-score were not reported in Khalid et al. (2023). What sets our study apart is the utilization

Table 5 Comparison of evaluation metrics with related works

Authors	Total fileless malware samples used	Accuracy (%)	TPR (%)	FPR (%)	Precision (%)	Recall (%)	F1-score (%)
This work	193,451	99.48	92.40	0.14	80.2	92.38	85.86
van der Eijk and Schuijt (2020)	17	99.99	88.24	0.004	75	88.2	81.0
Khalid et al. (2023)	26	93.3	87.5	0	NA	NA	NA

Table 6 Comparison of this work with prior image visualization based malware classification techniques

Authors	Algorithm	Visualization method/ mapping	Use case	Accuracy%	Real world data	Computation cost	Remark
Nataraj et al. (2011)	KNN	Grayscale/2D	Malware detection	97.18	No	High	Manual feature extraction with high computational cost
Dai et al. (2018)	MLP	Grayscale/2D	Malware detection	95.2	Yes	High	Extracted malware memory dump files at runtime, and hardware features may escape from the detected feature as malware
Ni et al. (2018)	CNN	Grayscale/2D	Malware detection	99.26	No	High	Features are extracted by disassembly of malware
Zhang et al. (2016)	CNN	Grayscale/2D	Malware detection	96.7	Yes	Medium	Only 2-tuple of opcode sequences are used to represent malware binaries
Abdullayeva (2019)	Deep learning	Color/2D	Malware detection	79.21	No	Low	Divided high resolution images into grids
Gibert et al. (2019)	CNN	Grayscale/2D	Malware classification	97.4	Yes	High	Big image size (256X256)
Vasan et al. (2020)	Deep learning	Color/2D	Malware classification	98.82	No	High	Due to complex and utilized deep pre-trained models
Kumar et al. (2016)	Random forest	Grayscale/2D	Android malware classification	91	No	High	Any feature selection method not used
Ran et al. (2018)	CNN	Grayscale/3D	Traffic classification	86.02	No	Low	Used only spatial features
Saleh and Ji (2020)	CNN	Grayscale/2D	Internet network classification	98.9	Yes	High	Very high dimensional image processing
This work	CNN	Grayscale/2D	Cobalt Strike beacon detection	99.48	Yes	Low	Raw traffic flow to image

of a substantial real-world dataset comprising fileless malware samples. This approach enhances the reliability and generalizability of our findings.

Table 6 provides a comparison between our approach and previous literature that incorporates both machine learning and image-based visualization. It is important to note that all the works presented in Table 6 focus on file-based malware. The table highlights that while prior techniques may achieve high accuracy, they often come with significant computational costs due to computationally intensive image pre-processing steps.

Limitations and challenges

One of the main challenges in this study stems from the limited availability of samples. There is a notable imbalance between the number of benign and malicious instances collected, which presents obstacles to achieving optimal performance and generalization of the model. Furthermore, obtaining labeled fileless malware samples for training purposes is challenging due to their scarce availability. To tackle this issue, we utilized resampling techniques, specifically downsampling the benign dataset and upsampling the malicious dataset, to mitigate the imbalanced class proportions.

While convolutional neural networks (CNNs) have shown effectiveness in various computer vision tasks, their application in classifying network traffic and detecting malware may face certain limitations. Unlike images, network traffic data is sequential and requires capturing temporal dependencies and contextual information. CNNs, on their own, may not be able to adequately model and utilize such dependencies, potentially impacting their effectiveness in these tasks. Another limitation is CNNs need fixed-size inputs, which is difficult to achieve with network traffic data. Network packets vary in length, and different protocols have distinct structures. Preprocessing data to fit fixed-size inputs may cause information loss or distortion.

Conclusion

Previous studies in the field of malware detection and classification primarily focused on file-based malware and utilized image visualization techniques. However, this study represents one of the early works specifically concentrating on fileless malware detection using image-based visualization and Convolutional Neural Networks (CNNs). Fileless malware exploits vulnerabilities in legitimate applications, making it difficult for conventional file-based methods to detect them. In this research, we conducted experimental research using evasive and realistic offensive traffic generated by proprietary Cobalt Strike beacons at the production level. By converting Cobalt Strike beacon payloads into grayscale images and training our proposed CNN model on them, we achieved a high level of accuracy. The outcome demonstrates that our approach effectively detects traces of fileless malware in network traffic.

Moving forward, our future studies will focus on innovative image enhancements aimed at enhancing the local contrast within regions of images that represent fileless malware traffic. This approach will enable us to identify similar pixel values, improve the interoperability of fileless malware images, and provide better input for the machine-learning classifier. Additionally, we plan to explore different image conversion algorithms, such as converting payloads into bitmaps and generating RGB-colored 3D images, in order to investigate their impact on the accuracy of machine learning-based traffic classification.

Acknowledgements

We would like to sincerely thank the anonymous reviewers for their valuable comments, which have greatly enhanced the quality of our paper. Additionally, we would like to express our deep appreciation to Ms. Tamera Ziglar, Senior Director of Development at the College of Engineering, North Carolina A &T State University (NCAT), for her instrumental role in facilitating and supporting the collaboration between NCAT and Palo Alto Networks, Inc. We are truly grateful for her efforts in establishing this fruitful partnership.

Author contributions

FAD performed the writing, while FAD and AN conducted the necessary experiments and analyzed the results obtained. SK secured funding, reviewed the obtained results, proposed revisions, and edited the manuscript. MW provided administrative support, both technically and financially; MW, KR, and YF reviewed the paper, provided comments, feedback, and valuable guidance. All authors read and approved the final manuscript.

Funding

This work was supported in part by NSF Grants #2113945 and #2200538 and a generous financial and technical support from Palo Alto Networks, Inc. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding agencies.

Availability of data and materials

Not applicable

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 13 March 2023 Accepted: 14 June 2023
Published online: 02 December 2023

References

Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. Osdi 16:265–283

Abdullayeva F (2019) Malware detection in cloud computing using an image visualization technique. In: 2019 IEEE 13th International Conference on Application of Information and Communication Technologies (AICT), pp 1–5. IEEE, https://doi.org/10.1109/AICT47866.2019.8981727

Babar FM (2020) Emerging & unconventional malware detection using a hybrid approach. PhD thesis, University of Windsor (Canada)

Barnes E (2021) Fileless attacks: addressing evolving malware threats. https://www.infosecurity-magazine.com/opinions/fileless-attacks-malware/ Accessed Accessed 19 Oct 2022

Barut O, Luo Y, Zhang T, Li W, Li P (2020) Netml: a challenge for network traffic analytics. 1, 13006, arXiv preprint arXiv:2004.13006

Borana P, Sihag V, Choudhary G, Vardhan M, Singh P (2021) An assistive tool for fileless malware detection. In: 2021 World Automation Congress (WAC), pp 21–25

Bozkir AS, Cankaya AO, Aydos M (2019) Utilization and comparision of convolutional neural networks in malware recognition. In: 2019 27th Signal Processing and Communications Applications Conference (SIU), pp 1–4

Bozkir AS, Tahillioglu E, Aydos M, Kara I (2021) Catch them alive: a malware detection approach through memory forensics, manifold learning and computer vision. Comput Secur 103:102166

Bressert E (2012) SciPy and NumPy: an Overview for Developers. " O'Reilly Media, Inc.", ISBN: 9781449361624

Bucevschi AG, Balan G, Prelipcean DB (2019) Preventing file-less attacks with machine learning techniques. In: 2019 21st International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), pp 248–252

Chukka A, Devi V (2021) Detection of malicious binaries by deep learning methods. International Conference on Internet of Things, vol 2021. Big Data and Security, IoTBDS-Proceedings. Science and Technology Publications, Lda, N/A, pp 132–139

Culjak I, Abram D, Pribanic T, Dzapo H, Cifrek M (2012) A brief introduction to opencv. In: 2012 Proceedings of the 35th International Convention MIPRO, pp 1725–1730

Dai Y, Li H, Qian Y, Lu X (2018) A malware classification method based on memory dump grayscale image. Digit Investig 27:30–37

Dhote Y, Agrawal S, Deen AJ (2015) A survey on feature selection techniques for internet traffic classification. In: 2015 International Conference on Computational Intelligence and Communication Networks (CICN), pp 1375–1380

Fang V (2018) Malicious PowerShell Detection via Machine Learning. https://www.mandiant.com/resources/blog/malicious-powershell-detection-via-machine-learning Accessed Accessed 22 Oct 2022

Gibert D, Mateu C, Planes J, Vicens R (2019) Using convolutional neural networks for classification of malware represented as images. J Comput Virol Hack Tech 15(1):15–28

Handaya W, Yusoff M, Jantan A (2020) Machine learning approach for detection of fileless cryptocurrency mining malware. J Phys Conf Ser 1450:012075

Hendler D, Kels S, Rubin A (2018) Detecting malicious powershell commands using deep neural networks. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp 187–197

Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195(1):215–243

Joo H, Choi H, Yun C, Cheon M (2021) Efficient network traffic classification and visualizing abnormal part via hybrid deep learning approach: Xception+ bidirectional gru. Glob J Comput Sci Technol 21(3):1–10
Kancherla K, Mukkamala S (2013) Image visualization based malware detection. In: 2013 IEEE Symposium on Computational Intelligence in Cyber Security (CICS), pp 40–44
Ketkar N (2017) Introduction to keras. In: Deep Learning with Python, Springer, pp 97–111
Khalid O, Ullah S, Ahmad T, Saeed S, Alabbad DA, Aslam M, Buriro A, Ahmad R (2023) An insight into the machine-learning-based fileless malware detection. Sensors 23(2):612
Khorsandroo S, Tosun AS (2018) Time inference attacks on software defined networks: Challenges and countermeasures. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), pp 342–349
Khorsandroo S, Tosun AS (2019) White box analysis at the service of low rate saturation attacks on virtual sdn data plane. In: 2019 IEEE 44th LCN Symposium on Emerging Topics in Networking (LCN Symposium), pp 100–107
Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980
Krupski J, Graniszewski W, Iwanowski M (2021) Data transformation schemes for cnn-based network traffic analysis: a survey. Electronics 10(16):2042
Kumar S et al (2020) An emerging threat fileless malware: a survey and research challenges. Cybersecurity 3(1):1–12
Kumar A, Sagar KP, Kuppusamy K, Aghila G (2016) Machine learning based malware classification for android applications using multimodal image representations. In: 2016 10th International Conference on Intelligent Systems and Control (ISCO), pp 1–6
Li P, Tang H, Yu J, Song W (2021) Lstm and multiple cnns based event image classification. Multimed Tools Appl 80(20):30743–30760
Lim H-K, Kim J-B, Kim K, Hong Y-G, Han Y-H (2019) Payload-based traffic classification using multi-layer lstm in software defined networks. Appl Sci 9(12):2550
Liu J, Zhang X, Zhang J, An J, Li C, Gao L (2018) Hyperspectral image classification based on long short term memory network. In: 2018 Fifth International Workshop on Earth Observation and Remote Sensing Applications (EORSA), pp 1–5
Mudge R (2018) Malleable-C2-Profiles. https://github.com/rsmudge/Malleable-C2-Profiles
Mudge R (2019) Cobalt Strike: Beware of Slow Downloads. https://www.cobaltstrike.com/blog/beware-of-slow-downloads/ Accessed 18 Apr 2023
Naeem H, Guo B, Naeem MR, Ullah F, Aldabbas H, Javed MS (2019) Identification of malicious code variants based on image visualization. Comput Electr Eng 76:225–237
Nataraj L, Karthikeyan S, Jacob G, Manjunath BS (2011) Malware images: visualization and automatic classification. In: Proceedings of the 8th International Symposium on Visualization for Cyber Security, pp 1–7
Nguyen TT, Armitage G (2008) A survey of techniques for internet traffic classification using machine learning. IEEE Commun Surv Tutor 10(4):56–76
Ni S, Qian Q, Zhang R (2018) Malware identification using visualization images and deep learning. Comput Secur 77:871–885
Rahman A (2021) Cobalt Strike: Defining Cobalt Strike Components & BEACON. https://www.mandiant.com/resources/blog/defining-cobalt-strike-components Accessed 05 Oct 2022
Ran J, Chen Y, Li S (2018) Three-dimensional convolutional neural network based traffic classification for wireless communications. In: 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pp 624–627
Rigaki M, Garcia S (2018) Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In: 2018 IEEE Security and Privacy Workshops (SPW), pp 70–75
Saad S, Briguglio W, Elmiligi H (2019) The curious case of machine learning in malware detection. Mach Learn Interpret Malware Detect 5:11
Saad S, Mahmood F, Briguglio W, Elmiligi H (2019) Jsless: A tale of a fileless javascript memory-resident malware. In: International Conference on Information Security Practice and Experience. Springer, pp 113–131
Saleh I, Ji H (2020) Network traffic images: A deep learning approach to the challenge of internet traffic classification. In: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pp 0329–0334
Sanjay B, Rakshith D, Akash R, Hegde VV (2018) An approach to detect fileless malware and defend its evasive mechanisms. In: 2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS), pp 234–239
Seazzu L (2016) Cobalt strike 3.0. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States)
Shabtai A, Moskovitch R, Elovici Y, Glezer C (2009) Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Inf Secur Tech Rep 14(1):16–29
Shapira T, Shavitt Y (2019) Flowpic: encrypted internet traffic classification is as easy as image recognition. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp 680–687
Sharma A, Vans E, Shigemizu D, Boroevich KA, Tsunoda T (2019) Deepinsight: A methodology to transform a non-image data to an image for convolution neural network architecture. Sci Rep 9(1):1–7
Smelcer J (2017) Rise of fileless malware. PhD thesis, Utica College
Su J, Vasconcellos DV, Prasad S, Sgandurra D, Feng Y, Sakurai K (2018) Lightweight classification of iot malware based on image recognition. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), vol 2, pp 664–669
Tahaei H, Afifi F, Asemi A, Zaki F, Anuar NB (2020) The rise of traffic classification in iot networks: A survey. J Netw Comput Appl 154:102538
Taheri S, Salem M, Yuan J-S (2018) Leveraging image representation of network traffic data and transfer learning in botnet detection. Big Data Cogn Comput 2(4):37
Taud H, Mas J (2018) Multilayer perceptron (mlp). In: Geomatic Approaches for Modeling Land Change Scenarios. Springer, pp 451–455
van der Eijk V, Schuijt C (2020) Detecting cobalt strike beacons in netflow data. Technical report, University of Amsterdam
Vasan D, Alazab M, Wassan S, Naeem H, Safaei B, Zheng Q (2020) Imcfn: Image-based malware classification using fine-tuned convolutional neural network architecture. Comput Netw 171:107138
Wang W, Zhu M, Zeng X, Ye X, Sheng Y (2017) Malware traffic classification using convolutional neural network for representation learning. In: 2017 International Conference on Information Networking (ICOIN), pp 712–717
Xie J, Yu FR, Huang T, Xie R, Liu J, Wang C, Liu Y (2018) A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges. IEEE Commun Surv Tutor 21(1):393–430
Xu P, Eckert C, Zarras A (2021) Falcon: malware detection and categorization with network traffic images. In: International Conference on Artificial Neural Networks, pp 117–128
Yadav B, Tokekar S (2021) Recent innovations and comparison of deep learning techniques in malware classification: a review. Int J Inf Secur Sci 9(4):230–247
Zhang J, Xiang Y, Wang Y, Zhou W, Xiang Y, Guan Y (2012) Network traffic classification using correlation information. IEEE Trans Parallel Distrib Syst 24(1):104–117
Zhang J, Qin Z, Yin H, Ou L, Hu Y (2016) Irmd: malware variant detection using opcode image recognition. In: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp 1175–1180
Zhang Y, Zhao S, Zhang J, Ma X, Huang F (2019) Stnn: A novel tls/ssl encrypted traffic classification system based on stereo transform neural network. In: 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), pp 907–910
Zhao J, Jing X, Yan Z, Pedrycz W (2021) Network traffic classification for data fusion: a survey. Inf Fusion 72:22–47
Zhu Y, Brettin T, Xia F, Partin A, Shukla M, Yoo H, Evrard YA, Doroshow JH, Stevens RL (2021) Converting tabular data into images for deep learning with convolutional neural networks. Sci Rep 11(1):1–11

Publisher's note