Open Source Intelligence for Malicious Behavior Discovery and Interpretation

Yi-Ting Huang, Chi Yu Lin, Ying-Ren Guo, Kai-Chieh Lo, Yeali S. Sun, and Meng Chang Chen

Abstract—Cyber threats are one of the most pressing issues in the digital age. There has been a consensus on deploying a proactive defense to effectively detect and respond to adversary threats. The key to success is understanding the characteristics of malware, including their activities and manipulated resources on the target machines. The MITRE ATT&CK framework (ATT&CK), a popular source of open source intelligence (OSINT), provides rich information and knowledge about adversary lifecycles and attack behaviors. The main challenges of this study involve knowledge collection from ATT&CK, malicious behavior identification using deep learning, and the identification of associated API calls. A MITRE ATT&CK based Malicious Behavior Analysis system (MAMBA) for Windows malware is proposed, which incorporates ATT&CK knowledge and considers attentions on manipulated resources and malicious activities in the neural network model. To synchronize ATT&CK updates in a timely manner, knowledge collection can be an automatic and incremental process. Given these features, MAMBA achieves the best performance of malicious behavior discovery among all the compared learning-based methods and rule-based approaches on all datasets; it also yields a highly interpretable mapping from the discovered malicious behaviors to relevant ATT&CK techniques, as well as to the related API calls.

Index Terms—Cyber threat intelligence, dynamic analysis, malware behavior analysis, MITRE ATT&CK framework

1 Introduction

Cyber attacks have proliferated recently, incurring damages that cost individuals and companies dearly. A powerful proactive defense collects information about known attacks and comprehensively understands malicious behaviors, and further exploits this knowledge to interdict and disrupt attacks or preparations for attack [1], [2]. Thus it is crucial to grasp the characteristics of malicious behavior and the resources used therein. Open source intelligence (OSINT) assimilates experience and knowledge from the cybersecurity community to form a common knowledge base for cyber threat studies that best supports a proactive defense [3].

The attack development life cycle, such as Lockheed Martin's cyber kill chain [4], the MITRE ATT&CK (Adversarial Tactics, Techniques and Common Knowledge) framework (hereafter referred to as ATT&CK) [5], and Mandiant's adversary life cycle [6], describes the adversary process at each stage of the attack. Take for example ATT&CK: the framework is designed to describe the attacker intent and malicious behavior at each tactic stage. Once all malicious behaviors are compiled, the cybersecurity analyst can correlate them to derive a clear picture of the attack, and take the necessary action to stop or mitigate the attack. The strength of ATT&CK, one of most popular OSINTs, is its structure and openness in collecting and sharing cyber threat intelligence. In this study, we crawl the contents of ATT&CK to build the needed knowledge about malware behavior to facilitate dynamic malware analysis via deep learning.

Information about adversaries is commonly published in cyber threat intelligence (CTI) reports presented with semantic descriptions and lists of manipulated resources. Comprehension of CTI is a large-scale data-driven process that involves systematic analysis of observations, including malware, suspicious events, and other rapidly evolving cybersecurity data. To facilitate CTI usage, many studies [7], [8], [9], [10], [11] focus on collecting, analyzing, and extracting evidence such as indicators of compromise (IoCs) in CTI reports. Dealing with increasingly sophisticated cyber threats and obtaining a overall picture of the fast-evolving attack scenario from OSINT CTI helps cybersecurity analysts handle potential attacks as they are unveiled.

Holmes [12] and RapSheet [13] are state-of-the-art systems that apply manually crafted expert rules to discover advanced persistent threats or tactics, techniques, and procedures (TTPs) to detect potential attacks on their host systems. In this paper, instead of investigating a computer's system log, we focus on analyzing the dynamic behavior of malware using the knowledge from ATT&CK and neural networks.

To analyze malware activity, dynamic analysis tools such as Cuckoo Sandbox [14], CWSandbox [15], and APIf [16] record execution steps in detail to generate execution traces. Cuckoo Sandbox further applies ATT&CK with rules contributed by volunteers to detect malicious behavior. However, due to the crowd-sourced nature of Cuckoo Sandbox, the completeness and timeliness of the contributed rules (called

- *Yi-Ting Huang, Kai-Chieh Lo, and Meng Chang Chen are with the Academia Sinica, Taipei 115024, Taiwan. E-mail: {ythuang, sage66730, mcc}@iis.sinica.edu.tw.*
- *Chi Yu Lin, Ying-Ren Guo, and Yeali S. Sun are with the National Taiwan University, Taipei 10617, Taiwan. E-mail: {r07946012, r09921a01, sunny}@ntu.edu.tw.*

Manuscript received 28 Dec. 2020; revised 23 July 2021; accepted 26 Sept. 2021. Date of publication 11 Oct. 2021; date of current version 14 Mar. 2022.
This work was supported in part by CITI, Academia Sinica, and by MOST under Grants 110-2218-E-001-001-MBK and 109-2221-E-001-010-MY3.
(Corresponding author: Yi-Ting Huang.)
Recommended for acceptance by Special Issue on XAI-CTI.
Digital Object Identifier no. 10.1109/TDSC.2021.3119008

Cuckoo Signatures) may not be consistent with ATT&CK. Therefore, in this study, we construct regular expression rules to implement knowledge within ATT&CK for use as a labeling method, in addition to the Cuckoo Signatures, for later use in deep learning. We crawl the MITRE website to extract and organize the relations of TTPs and malware that can be used as another labeling method. To account for MITRE website updates, all labeling processes must be automatic and incremental.

With the rapid development of artificial intelligence (AI), data-driven methods (i.e., machine learning and deep learning approaches) can be used for cyber threat analyses such as malware analysis [17], [18] or attack analysis [19], [20]. However, AI methods are seen as black boxes, which can create confusion and doubt [21]. For example, when security analysts analyze malware using an AI model, questions may arise such as "How can I trust the decision-making of this model?" or "How has the model come to this decision for a given malware sample?" To understand how a model learns from data, studies in disciplines as varied as image caption generation [22], sentiment analysis [23], and electronic health record applications [24] incorporate attention mechanisms in neural network models to interpret model outcomes. Generally, the attention mechanism, which calculates the probability distribution over inputs, is intuitively seen as an indicator of the model's focus, as a convincing explanation is to humans. In this study, we similarly employ the attention mechanism to interpret the model outcome. These two questions can be answered if the outcome of the model is not merely a decision but also concerns semantics, as is the case with TTPs, API calls, and the associated resources in this study.

In this study, we examine whether OSINT has a role to play in using intelligence to better interpret malware. Our goal is to discover malicious behavior based on the analysis of an execution trace of Windows malware, to interpret the discovered behavior as a collection of techniques (TTPs), and to find the API calls and system resources associated with these TTPs. Three main aspects of this study are as follows:

- OSINT for cyber threat intelligence: Assimilating threat intelligence from OSINT to intercept malicious behavior requires an information extraction mechanism and a competent neural network model.
- Malicious behavior discovery: Linking a low-level malware execution trace to a high-level description of malicious behavior (i.e., TTPs) requires that we close the semantic gap between them.
- Malicious behavior explanation: Helping the security analyst to better understand the captured malicious behavior, the associated API calls and manipulated system resources constitute observable evidence.

As dynamic malware analysis has been widely used for malware analysis [17], [25], [26], [27], we present MAMBA (MITRE ATT&CK based Malicious Behavior Analysis), a system that addresses the above aspects. An execution trace includes the sequences of the invoked API calls during execution of a malware sample. MAMBA starts by extracting the TTPs and their corresponding resources from the MITRE website and cited references, and discovers TTPs from malware and their corresponding API execution call sequences

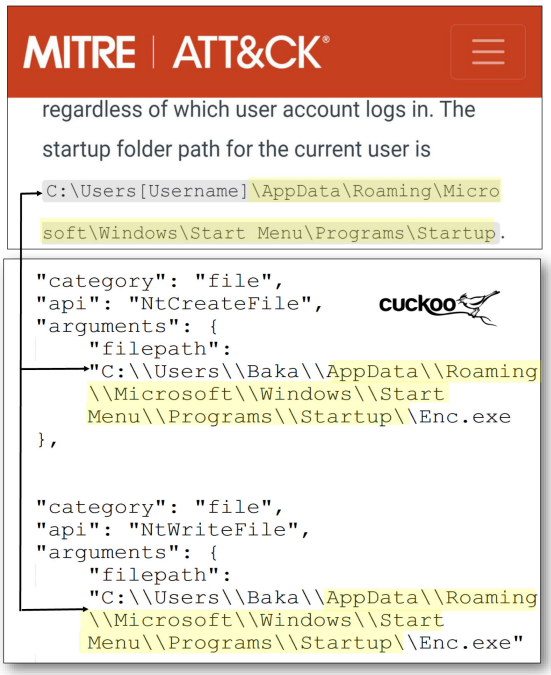

Fig. 1. Mapping knowledge from ATT&CK to a malware trace. The top MITRE webpage is about sub-technique T1547.001 and the bottom shows the API calls of JCry to partially carry out the technique.

via a well-crafted neural network model. Within an execution trace, malicious behaviors (TTPs) are undertaken by one or many API calls and may be described by CTI reports. MAMBA makes novel use of the information presented in ATT&CK as the pivotal reference in addressing the challenges in malware behavior interpretation. For instance, in Fig. 1, the sub-technique *T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder* refers to adding an executable program to a startup folder to maintain a foothold. This sub-technique can be identified when a malware sample attempts to add a malicious payload to the startup folder. High-level descriptions of TTPs in ATT&CK serve as explanations of malicious behavior, and can be used to link to low-level execution traces of malware by MAMBA. MAMBA is thus:

- Explainable. In contrast to traditional malware detection and malware classification tasks, we discover high-level semantic TTPs associated with low-level API calls for a malware sample.
- Comprehensive. Malicious behavior is composed of a series of operations and resources. By taking into account resource dependencies, MAMBA finds related TTPs and their API calls.
- Extendable. MITRE ATT&CK keeps up with the constant evolution of cybersecurity threats. MAMBA is designed to automatically retrieve the ATT&CK contents to reflect these changes.

To summarize, our work offers the following contributions:

- MAMBA incorporates knowledge from ATT&CK in deep learning analysis to discover malicious behavior.
- The MAMBA design and methodology are examined extensively using the contents of MITRE as well as

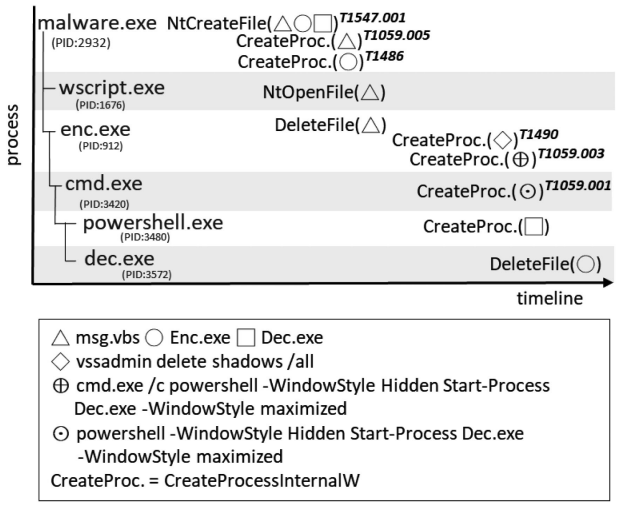

Fig. 2. Life cycle of a malware sample from malware family *JCry*.

real-world data. The evaluation outcomes meet the challenges.

- The study shows that the open-source intelligence of the MITRE ATT&CK framework facilitates cybersecurity applications.

2 Background and Motivation

In this section, we introduce a motivating example and present insight into using ATT&CK to interpret the malicious behavior lifecycle from an execution trace.

2.1 Motivating Example

We analyze a malware sample (MD5 c86c75804435efc380-d7fc436e344898) classified as a member of the JCry family [28], [29]. Fig. 2 depicts the JCry life cycle with an emphasis on its created processes, discovered TTPs, and the manipulated resources. JCry is ransomware disguised as an Adobe flash player update installer. Once it is clicked, it creates malicious files *msg.vbs* (△), *Enc.exe* (○), and *Dec.exe* (□), and stores these in the startup folder to maintain its persistence (in ATT&CK this is identified as *T1547.001 Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder*). These programs are executed when the user logs in. Executing *msg.vbs* displays an "Access Denied" message to warn that the Adobe Flash Player failed to update (*T1059.005 Command and Scripting Interpreter: Visual Basic*). The executable file *Enc.exe* encrypts the user's files for ransom (*T1486 Data Encrypted for Impact*), and also deletes shadow copies using a command to prevent recovery (*T1490 Inhibit System Recovery*), after which it launches *Dec.exe* using PowerShell to display the ransom note (*T1059.003 Command and Scripting Interpreter: Windows Command Shell*, *T1059.001 Command and Scripting Interpreter: PowerShell*).

We offer two observations. First, the manipulated resources are useful to group processes and API calls which work together to carry out malicious activities. For example, the manipulated resource *Enc.exe* (○) is used by the "malware.exe (PID=2932)", "enc.exe (PID=912)", and "dec.exe (PID=3572)" processes for its creation, execution, and deletion. Second, the malicious activities associated with these manipulated resources, e.g., files and commands, may correspond to techniques in ATT&CK. For example, the command *"cmd.exe /c powershell -WindowStyle Hidden Start-Process Dec.exe -WindowStyle maximized"* can be found on the ATT&CK TTP webpages (T1059.001 and T1059.003). While such malicious behavior is traditionally represented by indicators of compromise (IoCs) or signatures in intrusion detection systems (IDSs), in ATT&CK they are presented using natural language descriptions. In this study, the abundance and openness of the ATT&CK information facilitates the use of information retrieval techniques to collect and convert this data into knowledge for later use.

2.2 MITRE ATT&CK Framework

ATT&CK is a document source of post-compromise adversarial tactics and techniques based on real-world observations. From the contents of [5], ATT&CK is a behavioral model that consists of adversary tactics, techniques, and procedures (TTPs).

- A *tactic* represents the goals of an adversary. It categorizes the attack life cycle into different stages.
- A *technique/sub-technique* represents the technical means through which goals are accomplished. A sub-technique, inheriting a technique, corresponds to more specific action.
- A *procedure* in ATT&CK is exemplified by real-world examples, either software or an adversary group, to show their use of techniques or sub-techniques.

Each tactic serves as a class of techniques implemented by software to accomplish the tactic. For example, to establish *persistence (tactic)*, *JCry (malware)* may add a downloaded payload to the startup folder *(sub-technique T1547.001)*.

In recent years, this framework has become popular for describing the attack life cycle of either malware or an adversary group. This paper will focus on the techniques of all stages of Windows malware samples from ATT&CK. In this study, techniques refer to techniques as well as sub-techniques (hereafter techniques) and resources refer to files, libraries (modules), registries, processes, and networks. Malicious behavior of a malware sample can be represented by one or more techniques; the attack life cycle (kill chain) of malware is composed of a series of techniques.

2.3 Techniques and Execution Trace

The MITRE website provides descriptions of techniques for which MAMBA extracts resources and matches them with arguments of the API calls. This strategy is also supported by [30], in which a comprehensive analysis demonstrates a strong correlation between ATT&CK techniques and Windows API calls. As shown in Fig. 1, the resource mentioned in the webpage for technique *T1547.001 Registry Run Keys / Startup Folder* indicates that T1547.001 may be discovered if resource "C:\\Users\\...\\Startup\\Enc.exe" is accessed in an execution trace. As the figure shows, this specific resource can be found in the API calls "NtCreateFile" and "NtWriteFile", in which the connection constitutes an important clue to understanding the malicious activity. Following this procedure, MAMBA's neural network model is designed to learn the associations between TTPs and execution traces.

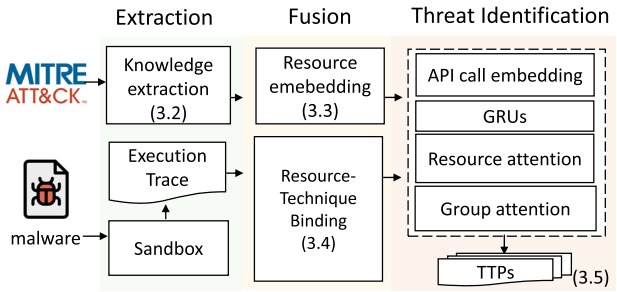

Fig. 3. MAMBA workflow.

3 System Design

The main design goal of MAMBA is to align a resource annotated with a TTP in ATT&CK to a manipulated resource used by malware. In this paper, matrices are represented using uppercase characters and vectors are represented in boldface using lowercase characters.

3.1 Overview

A high-level overview of the MAMBA workflow is shown in Fig. 3: it is composed of an extraction phase, a fusion phase, and a threat identification phase. The extraction phase includes technique extraction by extracting knowledge tuples from ATT&CK, and malware execution trace generation from a sandbox. The technique pages in ATT&CK present use cases performing the corresponding techniques. These use cases are treated as observable clues by which to detect techniques, and are extracted by MAMBA as the technique knowledge. We also consider series of API calls and sets of manipulated resources from an execution trace as a sequence of operations executed by malware. ATT&CK technique-related knowledge as well as execution-trace API calls and resources are collected in the extraction phase.

The fusion phase involves resource embedding and resource-technique binding. Even if the collected knowledge from ATT&CK and execution traces indicate the same malicious behavior, their constitutions may be different. The designed embedding mechanism maps resources to fixed-size vectors while preserving their semantic properties. In addition, in resource-technique binding we use a neural network to learn the connection between resources and techniques from ATT&CK to enable the proposed neural network to associate the embedding of resources from traces to techniques from ATT&CK.

Once the extraction phase and the fusion phase are complete, threats are identified by detecting techniques from a malware sample. First, API call embeddings are generated from the output of the fusion phase and are processed by gated recurrent units (GRUs) to obtain a sequential hidden vector. Attention mechanisms are applied to highlight the relevance between resources and API calls as well as dependencies among the bindings and API calls. Finally, threat identification yields the compromised techniques.

3.2 Knowledge Extraction From MITRE ATT&CK Framework

The first step of knowledge extraction is to extract a disclosed resource r related to a technique y as a tuple $\{r, y\}$ from the webpage for every technique in the MITRE ATT&CK framework. The regular expressions for r extraction from a token, a shadowed token (a token with gray background), or a sentence in the MITRE website are expressed in Table 1. A shadowed token is a complete path of a resource or command line; for example, the filename "C:\\Users[Username]\\...\\Startup" in Fig. 1 is a shadowed token, which can be recognized as the regular expression for directory (fd). Some resources in a sentence are not marked with gray background. For example, the sentence "... usage of the Windows Script Host (typically cscript.exe or wscript.exe) ..." from the MITRE webpage of *T1059.005 Command and Scripting Interpreter: Visual Basic* consists of two non-shadowed resources "cscript.exe" and "wscript.exe", which can be recognized by the composition of the regular expressions for filename (fn) and extension (fe). In summary, we collect from the MITRE website 988 resources associated with 229 techniques, forming 2100 $\{r, y\}$ pairs.

To evaluate the accuracy of knowledge extraction from the MITRE ATT&CK framework, we randomly selected 50 techniques to manually label technique-related resources from the corresponding webpages, and measured the accuracy of

TABLE 1
Regular Expressions for Resource Categories

C†	Token	Sentence
F	*(fd)?(fn)?fe*	
L	*(fd)?(fn)?le*	
R	*rk.**	*rksk*
	CLSID	
P	*cmd*	–
N	*ip*	
	domain	

Type‡	Regular expression	Example
fd	*([a-z]:\\|%\w+%\\?)?([^*|\\\:\"\<\>\?\/]+\\)**	C:\Users[Username]\...\Startup
fn	*[^(*|\\\:\"\<\>\?\/]+*	cmd
fe	*\.[a-z0-9]{3,5}*	.exe
le	*\.dll*	.dll
rk	*(hkcu|hklm|hkcr|hkcc|hku|hkey(_\w+){1,2}| sam|security|software|system|default|userdiff)*	HKEY_LOCAL_MACHINE
sk	*\\([^*|\\\:\"\<\>\?\/]+\\)*([a-z0-9]+)?*	\Software\Microsoft\...\Internet Settings
CLSID	*{[a-z0-9]{8}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{4}-[a-z0-9]{12}}*	{e17d4fc0-5564-11d1-83f2-00a0c90dc849}
cmd	*([a-z]:(\\[^*|\\\:\"\<\>\?\/]+)+)? [^*|\\\:\"\<\>\?\/]+(\.[a-z0-9]{3,5})?([^]+)**	vssadmin delete shadows /all /quiet
ip	*\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}(:\d{1,5})?*	127.0.0.1
domain	*([a-z0-9-_]+\.)+(com|net)*	red-gate.com

†*Abbreviations C, F, L, R, P, and N represent category, file, library, registry, process, and network, respectively.*
‡*fd = directory, fn = filename, fe = extension, le = library extension, rk = root key, sk = subkey path, CLSID = class id, cmd = command, domain = domain name.*

resource extraction. Two authors read every technique webpage on ATT&CK to find related resources and label a resource with a word or a phrase, such as a filepath, a registry, or a command, used to implement the technique. Their agreement (Cohen's kappa = 0.739) [31] is considered substantial. When they disagreed on a labeled resource, the third author joined the discussion to make a decision. This resulted in a total of 1159 resources annotated among 50 techniques. The average number of resources per technique was 23.18 (SD = 20.44). To verify the MAMBA extraction capability, we compared MAMBA with the shadowed tokens presented in the ATT&CK web pages: it outperforms ($p < 0.001$ by McNemar's test [32]) the ATT&CK shadowed tokens. MAMBA achieves a precision, recall, and F1 of 0.906, 0.770, and 0.833, respectively, with 19.72 (SD = 18.5) discovered resources, whereas the ATT&CK shadowed tokens achieve only 0.924, 0.475, and 0.627, respectively, and discover 11.90 (SD = 11.98) resources.

3.3 Resource Representation

In this step, each resource found from ATT&CK and the execution trace is embedded into a resource embedding e_r. An embedding maps a variable-length resource to a fixed-length feature vector in the embedding domain. As resources are not necessarily represented in the same way between ATT&CK and the execution traces, we seek to preserve their closeness in the embedding domain for later neural network processing. For instance, as shown in the example in Fig. 1, the startup folder path in ATT&CK consists of the token "Users[Username]", which is slightly different from the "Users\\Baka" in the execution trace. To ensure that the neural network learns properly, their embeddings should be close.

We employ the paragraph vector distributed memory method (PV-DM) [33] to transform a resource into an n-dimensional real-valued vector. PV-DM is an unsupervised learning algorithm to transform a sentence, a paragraph, or a document into a fixed-length vector. As it is based on skip-gram embedding techniques, it preserves semantics and word ordering to facilitate the use of embeddings for similarity computation while maintaining the closeness property. In this study, we tokenize each resource and treat each token as a word in the PV-DM model. To reduce the influence of unseen words, we build a resource vocabulary set by excluding out-of-vocabulary and rare words whose frequency is lower than a given threshold. Once the learning of the PV-DM for resource is completed, the resource embedding function is ready.

3.4 Resource-Technique Binding

Once resource embedding e_r is generated, the next step is to build a neural network to learn the relation between a resource and a technique. A resource can be seen as a plausible clue to the implementation of a technique y to achieve its tactical intent. A multiple layer perceptron (MLP) is trained using the pairs $\{e_r, y\}$ from ATT&CK, used to predict the likelihood of techniques given a resource from an execution trace.

Formally, given a set of N pairs of $\{e_r, y\}$ from ATT&CK, the objective of the learning function is to maximize the average log probability with respect to the MLP weights W_z:

$$\max \frac{1}{N} \sum \log p(y|e_r, W_z). \tag{1}$$

We apply W_z to derive the hidden vector z_r for each resource r, computed as

$$z_r = \sigma(W_z e_r), \tag{2}$$

where σ is the activation function. For a manipulated resource extracted from an API call, we use the same embedding function in Section 3.3 to transform r into e_r, and further compute the hidden vector z in (2), which can be considered its contribution to TTPs for later neural network processing.

3.5 Threat Identification Phase

The goal of the threat identification phase is to identify malicious behavior (TTPs) y from a malware execution trace with API calls $\boldsymbol{x} = \{x_1, x_2, \ldots x_{p \times |T|}\}$. Formally, given a training set of M pairs of $\{x, y\}$, the objective of the learning function is to maximize the average log probability with respect to the MAMBA neural network with all trainable weights θ, including W_c, W_n, W_v, and W_d (which will be defined later):

$$\max \frac{1}{M} \sum_m \log p(y|x, \theta). \tag{3}$$

The attack life cycle can be recognized by a series of techniques identified from API calls with their arguments.

A resource-based API call group is defined as a collection of the related API calls that share the same resource. Given a malware execution trace, the threat identification phase produces resource-based API call groups for each process, after which it compares resource-based API call groups with other call groups in all processes and predicts possible techniques. The structure of the threat identification phase is shown in Fig. 4.

An execution trace is composed of the traces of all processes; each process trace is a sequence of API calls. A single API call x consists of a category c, an API function name n, and one or more argument values (i.e., resources). In Fig. 1, for instance, API call "NtCreateFile" belongs to the "file" category and has argument values such as "C:\\Users\\... \\Startup\\Enc.exe". The embedding of API call x is a concatenation of the embeddings of category e_c, API name e_n, and resources e_{r_1}, e_{r_2}, e_{r_3} (only three resources are considered):

$$e_x = [e_c; e_n; [e_{r_1}; e_{r_2}; e_{r_3}]], \tag{4}$$

where $[;]$ is concatenation, and $e_{r_1}, e_{r_2}, e_{r_3}$ are from the PV-DM model in Section 3.3.

$$e_c = W_c x_c \tag{5}$$

$$e_n = W_n x_n, \tag{6}$$

where W_c and W_n are the weight matrices of category c and API name n, and x_c and x_n are one-hot encodings of category and API name. W_c and W_n are trained during the training phase of the MAMBA neural network model.

To preserve ordinal information, the sequence of the API call embeddings in a process is handled using gated recurrent

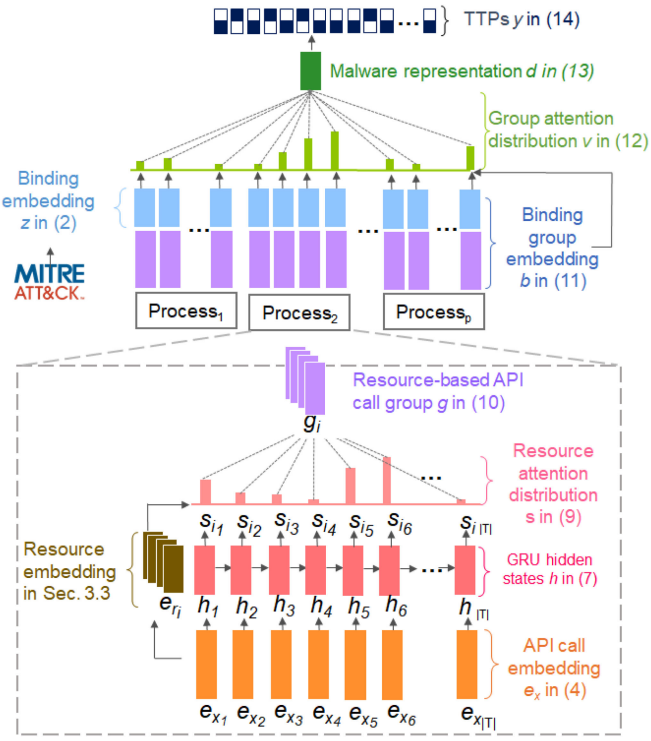

Fig. 4. MAMBA neural network model.

units (GRUs). A member of the recurrent neural network family, GRUs compute efficiently with performance comparable to LSTMs [34]. As GRUs are commonly used in malware analysis [35], [36], MAMBA considers GRUs to process the variable-length input sequence $\boldsymbol{e_x} = \{e_{x_1}, e_{x_2}, \ldots, e_{x_{|T|}}\}$ and produce a hidden state h. At time step t, the hidden state h_t of the GRU is updated by

$$h_t = \text{GRU}(h_{t-1}, e_{x_t}). \tag{7}$$

GRUs learn a probability distribution over an input sequence such that the output $\boldsymbol{h}$ encodes sequential information from the first API call to the current API call.

To find the connection between each pair of API call x_t and system resource r_i in a process, we use a resource attention mechanism as the score function that is the maximum value of the inner product of resource embedding e_{r_i} against the three resource embeddings $e_{r,t}$ of API call x_t in (8):

$$\text{score}(e_{r_i}, x_t) = \max\left(\frac{e_{r_i} e_{r_1 t}}{|e_{r_i}||e_{r_1 t}|}, \frac{e_{r_i} e_{r_2 t}}{|e_{r_i}||e_{r_2 t}|}, \frac{e_{r_i} e_{r_3 t}}{|e_{r_i}||e_{r_3 t}|}\right). \tag{8}$$

The result is normalized to derive the resource attention weights s_{it} as a distribution over all API calls:

$$s_{it} = \frac{\exp(\text{score}(e_{r_i}, x_t))}{\sum_{t'=1}^{|T|} \exp(\text{score}(e_{r_i}, x_{t'}))}. \tag{9}$$

Given these attention weights, we compute a group vector g_i as the weighted sum of the API call hidden states $\boldsymbol{h}$ for a certain resource r_i:

$$g_i = \sum_{t=1}^{|T|} s_{it} h_t. \tag{10}$$

Also, a binding embedding z_i for a resource r_i can be acquired in (2) as a feature corresponding to technique y. The group vector g_i is combined with the binding embedding z_i to yield the binding group embedding b_i for each resource:

$$b_i = [g_i; z_i]. \tag{11}$$

For each resource of a process, the binding group embedding b includes information not only from API calls but also from ATT&CK. At this step, each process is represented by a collection of binding group embeddings.

The next step is to aggregate the binding group embeddings from each process and produce a malware representation d for prediction. As shown in the example in Fig. 2, resources may be manipulated among processes; thus we apply a self-attention mechanism to highlight dependencies among the binding group embeddings. The self-attention mechanism allows each binding group embedding to interact with the other embeddings to determine which should get more attention:

$$v_i = \text{softmax}(W_v b_i), \tag{12}$$

where W_v is weight matrix of the two-layer dense network. The malware representation $\boldsymbol{d}$ is the aggregation of the group attention scores $\boldsymbol{v}$ and the binding group embeddings $\boldsymbol{b}$:

$$\boldsymbol{d} = \boldsymbol{v}\boldsymbol{b}. \tag{13}$$

The technique prediction task is a multi-label classification problem with a sigmoid layer at the end of the classifier. The predicted probability of each technique produced by the sigmoid function is independent of the others:

$$\boldsymbol{y} = \text{sigmoid}(W_d \boldsymbol{d}). \tag{14}$$

Algorithm 1 concludes the operations of MAMBA neural network model described above.

Algorithm 1. MAMBA Neural Network

Input: an execution trace $\boldsymbol{x}$
Output: a set of TTPs $\boldsymbol{y}$

while θ has not converged **do**
  Forward Propagation:
  for each process p **do**
    $\boldsymbol{r} \leftarrow$ Extract a set of resources from x_p
    $\boldsymbol{e_r} \leftarrow$ Get resource embedding in Section 3.3
    $\boldsymbol{z_r} \leftarrow$ Get binding embedding in (2)
    $\boldsymbol{e_x} \leftarrow$ API_call_embedding($\boldsymbol{x}$) in (4)
    $\boldsymbol{h} \leftarrow$ GRU($\boldsymbol{e_x}$) in (7)
    for each e_r **in** $\boldsymbol{e_r}$ **do**
      $s_{e_r,\boldsymbol{h}} \leftarrow$ resource_attention(e_r, $\boldsymbol{h}$) in (9)
      $g_r \leftarrow$ group_embedding($s_{e_r,\boldsymbol{h}}$, $\boldsymbol{h}$) in (10)
      $b_r \leftarrow$ binding_group(g_r, z_r) in (11)
    end for
  end for
  $\boldsymbol{v} \leftarrow$ group_attention($\boldsymbol{b}$) in (12)
  $d \leftarrow$ malware_representation($\boldsymbol{v}$, $\boldsymbol{b}$) in (13)
  $\boldsymbol{y} \leftarrow$ sigmoid(d) in (14)
  Backward Propagation:
  Conduct backward propagation with Adam;
end while
# Use the trained network to discover TTPs $\boldsymbol{y}$ of an execution trace $\boldsymbol{x}$

4 Empirical Studies

We designed experiments to answer the following critical questions.

- Q1: How effectively does MITRE knowledge improve TTPs extraction?
- Q2: How effectively are the true TTPs extracted from a given malware sample using MAMBA?
- Q3: What makes MAMBA capable of identifying TTPs?
- Q4: How well does MAMBA perform against realistic attack campaigns?
- Q5: How well does MAMBA locate API calls associated with the predicted TTPs?

For Q1 and Q2, we collected two datasets from MITRE and MalShare [37] and used three labeling methods: MITRE, Cuckoo, and RegExp. We compared the performance of MAMBA, two rule-based methods, five traditional machine learning methods, and three deep learning approaches in Evaluation 1. To answer Q3 and understand the contributions of each component, we further conducted an ablation study. To answer Q4, we analyzed malware samples provided in the ATT&CK APT29 description to examine MAMBA's capabilities. Finally, to answer Q5, we present a case study showing that MAMBA locates the API calls and manipulated resources associated with the predicted TTPs.

4.1 Data Collection

Here we describe the collection of samples and labels used in the evaluations. The MITRE ATT&CK framework (version 7) for Windows includes 12 tactics, 148 techniques, 214 subtechniques, and 378 pieces of software. We gathered malware samples and their corresponding TTPs presented in ATT&CK as the ground truth (note that this association is termed ATT&CK labeling). That is, each sample in the ATT&CK dataset is collected based on the documents referenced on each technique page on the MITRE ATT&CK website; its labels (TTPs) are determined accordingly. For each technique page on the MITRE ATT&CK website, the malicious activity is described by one or more referenced documents. We accessed these documents and used regular expressions to crawl and extract the MD5, SHA1, and SHA256 hashes of the associated malware samples. To validate the extracted hashes, we uploaded the hashes to VirusTotal [38] for verification. If a reference document involved more than one technique, we discarded it to eliminate ambiguity. We also discarded inaccessible references such as those with anti-crawler prevention, machine-unreadable content, and broken links. A total of 2,335 malware samples (referred to as the ATT&CK dataset) were collected corresponding to 67 techniques. We also collected 23,655 malware samples from MalShare [37] verified as malware by VirusTotal [38] from January 2018 to April 2019. The combination of the ATT&CK and MalShare datasets is called the Big dataset. The statistics of the two datasets are shown in Table 2. For instance, the average number of process per malware sample is 3.82, and the average API calls and resources per process are 2,023.47 and 329.55 respectively, for the ATT&CK dataset.

Since samples from MalShare lack TTP labels, we considered two rule-based label methods: Cuckoo Signatures (Version 2.0.7), which recognizes 43 TTPs using signatures provided by crowd intelligence, and RegExp, a regular expression set generated based on the TTP descriptions in ATT&CK which recognizes 169 TTPs. To label each malware sample, we applied these label methods to the Big dataset.

TABLE 2
Dataset Statistics

Source	ATT&CK	MalShare
Sample	2335	23655
Process	3.82±23.91	3.17±16.12
API call	2023.47±23415.45	4346.29±26399.03
Selected API calls	514.54±1199.05	642.05±1203.08
Resource	329.55±1296.19	531.21±6563.29
TTPs per sample	5.94±3.78	2.80±2.98[§], 12.70±5.89[¶]

[§]*Cuckoo Signatures* [¶]*RegExp*

We randomly divided the datasets into a training set (80%), a development set (10%), and a testing set (10%). We continued the above process until the F-test on the TTP distributions of the three sets showed no significant differences.

4.2 Implementation Settings

We used Cuckoo Sandbox [14] to obtain execution traces of malware samples. In the MAMBA implementation, the PV-DM model in Section 3.3 for resource embedding used the Gensim library [39] to produce a 100-dimension embedding vector as e_r. For the PV-DM model parameters, the minimum frequency threshold for each resource token was set to 5, and the size of the context window was 2. For training both resource-technique binding in Section 3.4 and the MAMBA neural network in Section 3.5, we used the loss function with cross entropy and the Adam optimizer to update the parameters, with an initial learning rate of 0.01.

The size of binding embedding z_r was set to 50. The identity function was used for the σ function in (2). The weight matrix W_z was for the two-layer dense networks, set to $\mathbb{R}^{100\times100}$ and $\mathbb{R}^{100\times50}$. We set each API call and GRU hidden state size to 400 and 100 respectively, and set the maximum timestamp t to 500. For category and API name embedding, the weight matrices W_c and W_n were $\mathbb{R}^{100\times7}$ $\mathbb{R}^{100\times36}$. Both of the weight matrices W_v and W_d were two-layer dense networks: W_{v1} and W_{v2} were set to $\mathbb{R}^{150\times64}$ and $\mathbb{R}^{64\times1}$, and W_{d1} and W_{d2} were $\mathbb{R}^{150\times64}$ and $\mathbb{R}^{64\times|y|}$.

4.3 Evaluation 1: MAMBA Evaluations

In this section, we compare the performance of MAMBA and other methods using the ATT&CK and Big datasets to answer Q1 and Q2. Table 3 compares the performance of MAMBA with two rule-based systems (Cuckoo Signatures and RegExp), five traditional machine learning methods, i.e., LinearSVC (Linear Support Vector Classifier), Random Forest, Decision Tree, GaussianNB (Gaussian Naive Bayes), and KNeighbors (K-nearest Neighbors) in Scikit-learn [40], and three conventional neural networks, i.e., MLP (multilayer perceptron), RNN (recurrent neural network), and LSTM (long-short term memory). Machine learning-based and deep learning-based methods are commonly used in malware analysis [35], [41], [42], [43]. As traditional machine learning methods could not accept a complete execution trace as input, we took the first five hundred API calls (with API categories and API function names only) of

TABLE 3
Comparison of Various Methods for TTP Discovery

Source	ATT&CK dataset					Big dataset (ATT&CK + MalShare)									
Label	ATT&CK					Cuckoo					RegExp				
Metric	P	R	F1	FPR	FNR	P	R	F1	FPR	FNR	P	R	F1	FPR	FNR
Cuckoo	0.078*	0.048*	0.049*	0.044*	0.858*	-					0.087*	0.326*	0.131*	0.036*	0.903*
RegExp	0.081*	0.206*	0.099*	0.351*	0.659*	0.061*	0.231*	0.092*	0.283*	0.468*	-				
Random Forest	0.375†	0.144*	0.208*	0.005*	0.429*	0.417*	0.203*	0.273*	0.003†	0.400	0.813*	0.850*	0.831*	0.010*	0.094*
LinearSVC	0.200*	0.370‡	0.260*	0.088*	0.343*	0.380*	0.408*	0.394*	0.039*	0.298*	0.734*	0.584*	0.650*	0.046*	0.133*
GaussianNB	0.321*	0.490	0.388*	0.092*	0.268†	0.349*	0.560*	0.430*	0.089*	0.217*	0.738*	0.658*	0.606*	0.047*	0.132*
Decision Tree	0.342*	0.332*	0.337*	0.040*	0.344*	0.437*	0.464*	0.450*	0.044*	0.316*	0.745*	0.745*	0.745*	0.026*	0.113*
KNeighbors	0.416*	0.365*	0.389*	0.028*	0.375*	0.468*	0.456*	0.462*	0.029*	0.281*	0.777*	0.772*	0.774*	0.020*	0.110*
MLP	0.556*	0.365*	0.427*	0.010	0.319*	0.833*	0.779*	0.801*	0.006	0.111*	0.938*	0.926*	0.928*	0.004*	0.037*
RNN	0.461*	0.214*	0.288*	0.009	0.394*	0.688*	0.604*	0.631*	0.010*	0.251*	0.850*	0.774*	0.796*	0.009*	0.115*
LSTM	0.552‡	0.289†	0.362‡	0.006†	0.358*	0.770*	0.711*	0.735*	0.007	0.144*	0.920*	0.900*	0.908*	0.006*	0.051*
MAMBA	0.667	0.569	0.591	0.011	0.220	0.898	0.887	0.891	0.006	0.061	0.945	0.954	0.949	0.005	0.024

$^{\dagger}p < 0.05$, $^{\ddagger}p < 0.01$, $^{\star}p < 0.001$.
Mann–Whitney U test is applied to compare the performance of MAMBA against that of each compared method.

an execution trace and used PCA (principle component analysis) [44] to reduce the dimensions of the execution trace. For the traditional machine learning methods, the reduced API call sequences and associated TTPs were used as input, whereas for the conventional deep learning models, MAMBA's API call embeddings and associated TTPs were used as input.

Evaluation 1 has 3 sets of comparisons using ATT&CK, Cuckoo Signatures, and RegExp labels sequentially as ground truth for comparisons; the first set is used with the ATT&CK dataset, and the other two are used with the Big dataset. Table 3 compares various methods for TTP discovery in terms of precision (P), recall (R), F1 score, false positive rate (FPR), and false negative rate (FNR), including the statistical significance level as per the Mann–Whitney U test [45] in comparison to MAMBA. MAMBA has significant performance differences with most of the compared methods. Both Cuckoo Signatures and RegExp perform poorly as the TTPs cover only part of ATT&CK labels on the ATT&CK dataset; as a result, they achieve F1 scores of 0.049 and 0.099 and false negative rates of 0.858 and 0.659. The five traditional machine learning methods perform slightly better than the rule-based approaches, with F1 scores ranging from 0.389 to 0.260, as they learn the relationship between API calls and TTPs. The conventional neural networks learn from the embeddings of API calls, yielding high precision, i.e., 0.556, 0.461 and 0.552 for MLP, RNN, and LSTM, and relatively low false positive rates, i.e., 0.010, 0.009, and 0.006. Compared to the deep learning models, MAMBA demonstrates significant improvements due to its preservation of resource and group dependencies in the attention mechanism and its utilization of resource and binding embeddings.

Take JCry as an example: as shown in Section 2.1, JCry is identified with six malicious behaviors: T1547.001, T1486, T1490, T1059.001, T1059.003, and T1059.005. Whereas the MLP model correctly labels T1547.001 only, MAMBA additionally identifies T1059.001 and T1059.003 because it makes use of ATT&CK resource knowledge such as "*powershell*", "*-WindowStyle Hidden*" and "*cmd.exe*" (see Section 4.6.2 for details). For the LSTM model, although it handles sequential data, it fails to identify any of JCry's malicious behavior as it ignores the semantics and structure of malware. To sum up, MAMBA yields superior performance due to the resource attention and group attention mechanisms as well as ATT&CK knowledge and resource embeddings.

For the second and third set of comparisons, first, the agreement (Krippendorff's $\alpha = -0.120$) [46] between Cuckoo Signatures and RegExp is low, which demonstrates the subjective nature of rule-based systems. Conventional rule-based approaches exhibit limited performance on TTP discovery, as shown in Table 3. Moreover, given sufficient training data, learning-based approaches learn from latent information in the data, and yield better predictions using the Big dataset than when using the ATT&CK dataset. When using these two labeling methods MAMBA achieves around 90% in terms of precision, recall, and F1 score, and produces a 0.6% false positive rate and a 6% false negative rate, which is the best performance of all the methods.

From Table 3, MAMBA outperforms in terms of precision, recall, and F1 at 0.667, 0.569, and 0.591 respectively. To answer *Q1*, the result shows the ATT&CK labeling and dataset does provide useful knowledge to extract TTPs from execution traces, but due to the limited number of malware samples and TTP labels, the resultant performance is moderate. For question *Q2*, we conclude that: 1) MAMBA accurately identifies TTPs compared to rule-based and other learning-based approaches on the Big dataset using both labeling methods. 2) Comparing the results on the ATT&CK dataset against those on the Big dataset, given sufficient samples and labels, MAMBA achieves high precision, recall, and F1, attesting the efficacy of the MAMBA neural network model.

4.4 Evaluation 2: Ablation Test

MAMBA includes knowledge from ATT&CK (binding embeddings), group dependencies (group attention), and API calls (resource attention). We conducted an ablation study to understand the contributions of each component to TTP identification using the RegExp labels on the Big dataset.

Table 4 shows that after the removal of one or two components, MAMBA still performs well, and indeed demonstrates statistically significant improvements by the Mann–Whitney

TABLE 4
Ablation Test Results on Big Dataset

	P	R	F1
MAMBA	0.945	0.954	0.949
– group attention	0.942*	0.935*	0.938*
– resource attention	0.911	0.920†	0.915†
– binding embedding	0.940*	0.933	0.936
– (resource + group attention)	0.949‡	0.925*	0.936*
– (binding embedding + group attention)	0.886*	0.932*	0.908*
– (binding embedding + resource attention)	0.934*	0.912*	0.919*

†p <0.05, ‡p <0.01, *p <0.001. *Mann–Whitney U test is applied to compare the performance of MAMBA against that of each compared method.*

U test [45] against most of the models. All components have positive effects on the F1 score; in particular, resource attention, that is, measuring the association between manipulated resources and API calls, has an obvious impact. For instance, MAMBA discovers that a sample (MD5 b9f7ff253e508d01cfa6defccdbad400) extensively manipulates the resource "*aevce.exe*" among invoked API calls such as RegSetValue, NtCreateFile, and CreateProcessInternalW to implement T1547.001; when resource attention is removed, MAMBA does not discover this TTP. This verifies that resource attention measures the correlation between resources and API calls, enhancing the ability of MAMBA to discover TTPs. Another interesting finding is that precision increases when only considering binding embedding, ("– (resource attention + group attention)"); one reasons is because it generates the fewest TTP predictions to increase precision. Taking *T1057 Process Discovery* as an example, 33 resources such as "*tasklist*" are extracted from ATT&CK. MAMBA recognizes that sample MD5 203d4f3541012300368ee97420f46f5f attempts to list processes with a command, *tasklist /fi "imagename eq rfusclient.exe"*, whereas MAMBA with fewer components fails to identify the TTP. This shows that the relation between the command and the extracted resource transfers successfully to the binding embedding, boosting performance.

To answer *Q3*, each component of MAMBA, binding embedding, group attention and resource attention helps to discover TTPs.

4.5 Evaluation 3: APT29 Case Study

The ATT&CK evaluations use known attack methods of APT groups such as APT29 [47] to evaluate cybersecurity products. In 2019, 21 security vendors participated in the evaluation using this emulated adversary environment. With this experiment we examined the capability of MAMBA trained with ATT&CK dataset and ATT&CK labeling in dealing with malware samples used in a well-known APT29 adversary, and compared the predicted TTPs with the ATT&CK APT29 Evaluation results [48]. The malware samples deployed in APT29 are well-documented in [49], [50], [51]; we collected these 310 malware samples for the evaluation and compared the outcome with those of the attending vendors.

Taking 310 execution traces as inputs, MAMBA discovered 67 TTPs among 9 tactics. (As a side note, when trained with the Big dataset, MAMBA discovered 90 TTPs among 10 tactics.) Whereas 56 TTPs are listed in the APT29 Evaluations, Fig. 5 shows that 20 TTPs are recognized by MAMBA against those results from the security vendors [48]. In Fig. 5,

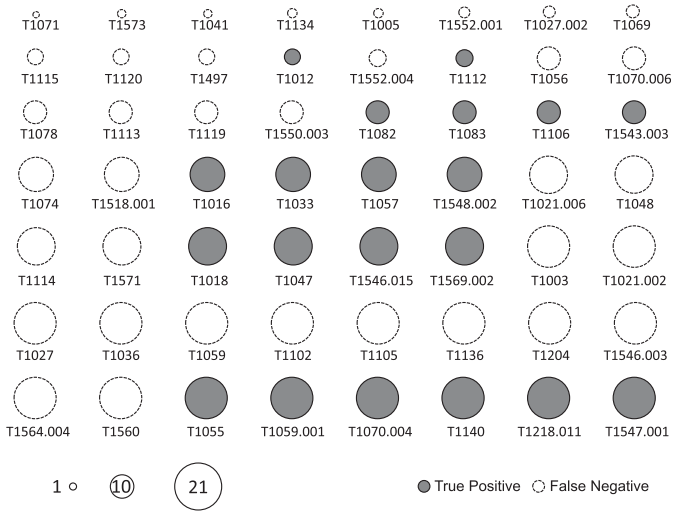

Fig. 5. Comparisons of MAMBA and security vendors on 56 TTPs listed in APT29 evaluations.

the larger the circle is, the more vendors recognize the TTP; true positives and false negatives of MAMBA prediction are represented with different colors. In addition, MAMBA recognizes TTPs—e.g., *T1056.001 Input Capture: Keylogging* and *T1059.003 Command and Scripting Interpreter: Windows Command Shell*—beyond the 56 TTPs in the ATP29 Evaluation, although the discovery of these two TTPs is consistent with [49]. However, MAMBA does produce false positive TTPs, such as *T1546.010 Event Triggered Execution: AppInit DLLs*, which is misidentified because MAMBA treats the registry subkey "... Windows\LoadAppInit_DLLs" in an execution trace as "...\AppInit_DLLs" in the MITRE webpage.

To answer *Q4*, MAMBA demonstrates the feasibility of capturing TTPs on malware samples used in a threat group. However, there are still some shortcomings due to the statistical characteristics of deep learning and the size limitation of ATT&CK dataset.

4.6 Evaluation 4: Resource and API Locating

This section presents a post-processing heuristic to locate the APIs and manipulated resources of malicious behaviors, and discusses a case study that demonstrates the effectiveness of API call locating.

4.6.1 Inference and API Locating

At inference time, MAMBA predicts TTPs $\hat{\boldsymbol{y}}$ and locates related API calls $\bar{\boldsymbol{x}} \subseteq \boldsymbol{x}$ for a given execution trace $\boldsymbol{x}$. Given the group attentions in (12) and the resource attentions in (9), we find the dominant resources for discovering TTPs and locating the related API calls in a process. More specifically, a set of manipulated resources $\bar{r}_{\hat{y}}$ is selected based on two criteria: i) the similarity between the resources extracted from ATT&CK and the manipulated resources, and ii) the group attention. The similarity scores reveal the likelihood that a certain resource is being manipulated to implement a TTP. The group attention measures how much information a resource provides, that is, whether it is a common or rare resource across API calls. A large group attention value for a resource indicates that the resource is frequently used among API calls or processes; in contrast, a resource with a

small group value means that it is uniquely representative, or is used only by chance. Security analysts use this to select observable resources by setting a threshold thd for the corresponding similarity scores and k as the number of highest and lowest attention values. Once the resources are selected, malicious behavior can be located via the API calls whose resource attention values are larger than the largest attention value minus α times the standard deviation. Algorithm 2 describes the locating process for the alignment of API calls and resources.

Algorithm 2. MAMBA API Call Locating

Input: an execution trace $\boldsymbol{x}$, a set of group attentions $\boldsymbol{v}$, a set of resource attentions $\boldsymbol{s}$, a set of predicted TTPs $\hat{\boldsymbol{y}}$ from the MAMBA neural network, knowledge pairs of {resource r, TTP y} from Section 3.2

Output: a set of selected manipulated resources $\bar{\boldsymbol{r}}_{\hat{y}}$ and the corresponding API call subsequences $\bar{\boldsymbol{x}}_{\hat{y}j}$

for each TTP $\hat{y}$ **do**
  # Select possible resources $\bar{\boldsymbol{r}}_{\hat{y}}$ for a certain TTP $\hat{y}$
  $\boldsymbol{i} \leftarrow$ Extract resources r from knowledge pairs $\{r, y\}$ given TTP $\hat{y}$
  for each resource i **do**
    for each manipulated resource j **in** x **do**
      $\text{score}(i, j) = \text{sim}(e_i, e_j)$
    end for
  end for
  $\bar{\boldsymbol{r}}_{\hat{y}} \leftarrow$ Extract j when $\text{score}(i, j) > thd_{\hat{y}}$
  $\bar{\boldsymbol{r}}_{\hat{y}} \leftarrow$ Extract top and bottom k of sort($\boldsymbol{v}$)
  # Locate API call $\bar{\boldsymbol{x}}_{\boldsymbol{j}}$ for a certain resource j.
  for each resource j **in** $\bar{\boldsymbol{r}}_{\hat{y}}$ **do**
    for each resource attention s **in** $\boldsymbol{s}_j$ **do**
      $\bar{\boldsymbol{x}}_{\hat{y}j} \leftarrow$ Extract x when $s \geq \max(\boldsymbol{s}_j) - \alpha \text{std}(\boldsymbol{s}_j)$
    end for
  end for
end for

4.6.2 Case Study

As there is no benchmark for a quantitative evaluation of the efficacy of associated API call locating, we here present a case study on a JCry malware sample to demonstrate MAMBA's ability to align API calls. The malicious activities of JCry were presented as a motivating example in Section 2.1. The malware sample manipulates 8,440 resource groups in seven processes. Based on the MITRE website [28], JCry is labeled with seven TTPs: T1547.001, T1059.001, T1059.003, T1059.005, T1486, T1490, and T1204.002. Nine techniques are predicted by MAMBA, among which T1547.001, T1059.001, and T1059.003 are consistent with the content on the MITRE website; T1033, T1070.004, T1082, T1016, T1218.010, and T1220 are not listed.

Fig. 6 shows the sorted group attentions and their associated resource attentions of selected resources found by Algorithm 2. The highest group attention refers to subkey 2392_regkey1 "HKEY_LOCAL_MACHINE\SOFTWARE\Classes", which is heavily manipulated (443 times). Its high group attention scores and high associated resource attentions lead to the discovery of TTP *T1082 System Information Discovery*. The 2392_regkey1 subkey and its many high resource attentions, depicted as the first row of resource

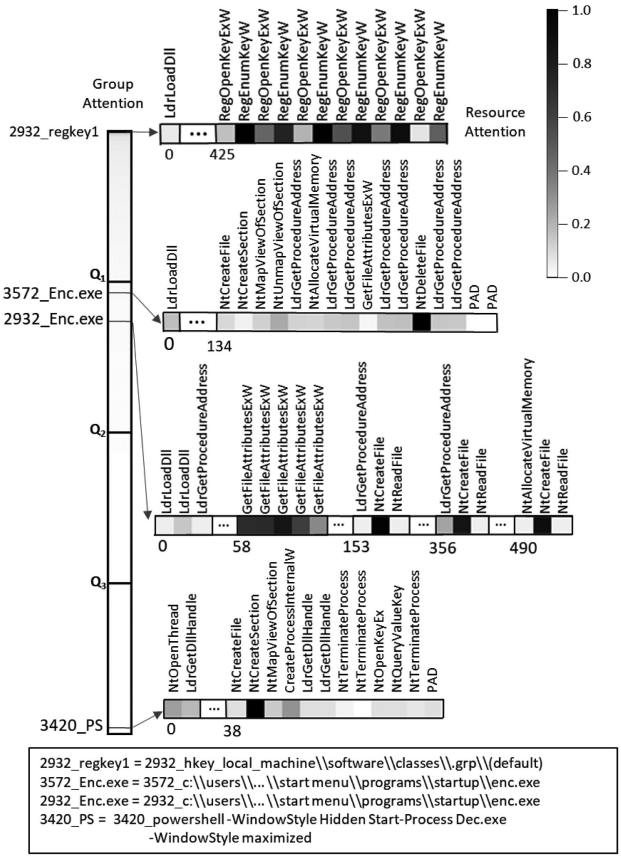

Fig. 6. Group attention and resource attention diagram in JCry analysis.

attention in Fig. 6, such as APIs "RegEnumKeyW" and "RegOpenKeyExW" that enumerate and attempt to open subkeys, support the discovery of the TTP. The behavior meets the description of T1082; "RegEnumKeyW" and "RegOpenKeyExW" are the associated API calls.

The algorithm then finds 3572_Enc.exe, with which we find the TTP *T1070.004 Indicator Removal on Host: File Deletion* discovered by MAMBA but not documented on the MITRE website [28]. The group attention (3572_Enc.exe) and its highest resource attentions "NtDeleteFile" together support the discovery of TTP T1070.004. This malicious behavior can be observed in the execution trace as well: it deletes the self-created files to evade detection.

Fig. 6 depicts the discovery of TTPs T1547.001 and T1059.001, which are listed on the MITRE website. The group attention of 2932_Enc.exe is high and the resource attentions of associated API calls such as "NtCreateFile" and "GetFileAttributesExW" are also high, suggesting *T1547.001 Boot or Logon Autostart Execution: Registry Run Keys/Startup Folder*. The next group attention for the command line 3420_PS and its resource attentions "NtCreateSection" and "CreateProcessInternalW" contribute to the identification of TTP *T1059.001 Command and Scripting Interpreter: PowerShell*.

However, MAMBA incorrectly recognizes TTPs *T1033 System Owner/User Discovery* and *T1218.010 Signed Binary Proxy Execution:Regsvr32*, as their behaviors are not found in the Cuckoo Sandbox execution trace. TTP *T1220 XSL Script Processing* is mistakenly recognized when JCry renames and encrypts XML files. Moreover, *T1204.002 User Execution: Malicious File* is not recognized because it involves human

TABLE 5
Discovered Life Cycle of JCry

Execution	Persistence	Privilege escalation	Defense evasion	Discovery
T1059.001	T1547.001	T1547.001	(T1070.004)	(T1016)
T1059.003			(T1218.010)	(T1033)
			(T1220)	(T1082)

(*) indicates the TTP is not listed in [28].

action. Finally, MAMBA does not recognize TTPs T1059.005, T1486, and T1490.

Following the MITRE ATT&CK framework, Table 5 presents the life cycle of the associated TTPs of the JCry analysis, indicating correspondences between the discovered TTPs and TTPs listed in [28].

To answer *Q5*, the group and resource attention mechanisms indeed capture the relations among the predicted TTPs, the manipulated resources, and the corresponding API calls; some mistakes are made because they are not found in the execution trace, some require human interaction, and some are not explainable.

5 Related Work

Open Source Intelligence and Cyber Threat Intelligence. Open Source Intelligence (OSINT) involves information gathering, collection, processing, and correlation from open data sources such as technical reports, blogs, forums, or social networks [52]. In recent years, OSINT and CTI reports have been utilized to provide threat information to proactively mitigate potential attacks [53]. Whereas a number of studies have discussed the mechanism and standards of threat intelligence extraction [7], [8], [10], [11] and sharing [54], [55], another line of work investigates the integration of threat intelligence for detecting network threats [56] and advanced persistent threats [57]. In this study, MAMBA integrates MITRE ATT&CK into a neural network model to leverage the ever-increasing OSINT and CTI reports for malware behavior analysis.

Malware Behavior Discovery. Behavior-based malware analysis detection learns behavior first and detects malware later [27], [58], while other studies focus directly on extracting malicious behaviors or common behaviors of a family. Christodorescu *et al.* [59], Fredrikson *et al.* [60], and Palahan *et al.* [61] mine malicious behavior by comparing dependence graphs extracted from the execution behavior of malware against that of benign software, and produce sub-graphs specific to malicious behavior. Bayer *et al.* [26] develop clustering algorithms to discover behaviors that are characteristic of a family in a group of malware. Rieck *et al.* [25] learn malware behavior using support vector machines, in which API calls with high corresponding weights represent malware behavior. Similar to these works, MAMBA discovers malware behavior. However, whereas their discovered behavior must be further interpreted by security analysts, MAMBA presents them as mappings to MITRE ATT&CK TTPs.

MITRE ATT&CK Techniques. Several recent studies leverage ATT&CK as a rich source of knowledge. TTPDrill [7] constructs an ontology of attack patterns that maps threat actions to TTPs based on a collection of CTI reports. Legoy *et al.* [62] develop rcATT to identify tactics and techniques for textual CTI reports based on ATT&CK content. Al-Shaer *et al.* [63] analyze and cluster the APT and software data reported by ATT&CK to predict techniques. These works concern only the natural-language contents of the MITRE or CTI reports, and can be seen as a foundation that facilitates the use of ATT&CK knowledge in MAMBA.

TTP Identification. Holmes [12] and RapSheet [13], the most closely related works, attempt to connect adversarial behavior to ATT&CK techniques. Both leverage provenance graphs and rules to map audit logs to advanced persistent threat (APT) stages and TTPs on a host. These works differ from the proposed method in three ways. First, the scope of MAMBA is different from the two studies, as they focus on either APT or endpoint detection and not malware behavior. Also, the focus of this work is to develop a neural network and integrate ATT&CK knowledge to discover TTPs, which differs greatly from their pattern matching approaches. Finally, as ATT&CK evolves, MAMBA is easily adapted to new versions of the framework, whereas their approaches require human involvement for pattern development.

Embedding Techniques. Rather than manually developing solutions for security-related events, the use of embedding techniques has brought significant benefits to the field of cybersecurity. Mimura *et al.* [19] embed proxy server logs to detect unknown malicious communication. ATTCK2vec [20] learns attack embeddings which correlate a collection of billions of security events with common vulnerabilities and exposures (CVEs). PROVDETECTOR [17] embeds malware execution paths in the provenance graphs to detect stealthy malware. SDAC [18] embeds API call paths from each Android app process to detect malware. Similar to [19] and [17], MAMBA uses a PV-DM model to embed resource values from both ATT&CK and execution traces. With advances in embedding techniques, we preserve the closeness in the embedding domain and use it to bind components and identify threats.

6 Conclusion

In MAMBA, the proposed system, the key drivers to discovering MITRE techniques include 1) incorporating knowledge from the MITRE ATT&CK framework, 2) considering the relation between resources and API calls, and 3) leveraging resource dependencies among processes. Based on these drivers, the design of the MAMBA neural network includes 1) binding embeddings, 2) resource attention, and 3) group attention. These ensure that MAMBA achieves the best performance on both ATT&CK and Big datasets. In addition, this study demonstrates a usage of the MITRE ATT&CK framework in cybersecurity applications in general that increases the interpretability of the deep learning outcomes.

The information collected from ATT&CK has limitations, as the data collection process of the MITRE ATT&CK framework relies heavily on contributions from security experts and organizations; as a result the data may be neither timely nor complete. This limits the capability of cybersecurity systems that rely solely on the MITRE ATT&CK framework as a knowledge source. In this case, performance can be improved if the system adopts more OSINTs and other reliable sources. In this work, we focus only on Windows malware and its

associated TTPs, but the concept of our approach is not limited to certain operating systems since malicious behaviors could be discovered by aligning the manipulated resources to ATT&CK knowledge.

When an adversary with knowledge of MAMBA regular expressions creates new malware variants to avoid TTP discovery, there are two ways that MAMBA could still recognize the technique. First, besides extracting knowledge directly from ATT&CK, MAMBA can learn malicious behaviors from malware execution traces. For example, the TTP *T1070.004 File Deletion* contains ten resources, including the command "*rm -rf*", on ATT&CK. Malware sample MD5 cd1c95aa6f45101735d444aeb447225c does not use any extracted resource appearing on ATT&CK such as "*rm -rf*", but implements the technique with an API call, "*DeleteFileW*". In this case MAMBA still identifies the TTP *T1070.004 File Deletion* by learning the association between "*rm -rf*" and "*DeleteFileW*" from data. This shows MAMBA can learn the relationship between TTPs and API calls from execution traces. Second, we leverage the fact that open-source intelligence reports are constantly updated. When an additional resource is used to implement a technique, OSINT knowledge (e.g., ATT&CK) may be updated accordingly. For example, resource *powershell.exe -windowstyle hidden -exec bypass -file "%appdata%\onedrive.ps1"* used to implement TTP *T1059.001 Command and Scripting Interpreter: PowerShell* is not available in version 6[1] but is in version 7[2]. Given the latest version of ATT&CK, MAMBA can keep up with the behavior of JCry. To summarize, MAMBA can still discover malicious behavior of new malware variants by not only learning from data but also from the up-to-date CTI knowledge. Detecting adversarial examples in malware analysis, such as [64] and [65], remains an active research area in which MAMBA has room to improve.

This is the first attempt to leverage the knowledge collected from OSINTs in deep learning analysis of malicious behavior. Still, MAMBA has several drawbacks. For instance, some discovered TTPs cannot be explained and some TTPs cannot be found. One reason for these limitations is insufficient or imbalanced annotated resources on the MITRE website [62]. In future work, we plan to enrich the knowledge associated with TTPs via data augmentation or by adopting more OSINTs. On the other hand, we observed that the results on the attention distribution in our case study indicate only a human-understandable explanation and do not reflect the model's actual reasoning process for the model's outcome [66]. Another direction of future work is to improve the faithfulness of attention-based explanations for malicious behavior discovery, such as in [67].

REFERENCES

[1] Executive Office of the President, "Artificial Intelligence, Automation, and the Economy," 2016. [Online]. Available: https://obamawhitehouse.archives.gov/sites/whitehouse.gov/files/documents/Artificial-Intelligence-Automation-Economy.PDF

[2] N. Kaloudi and J. Li, "The AI-based cyber threat landscape: A survey," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–34, 2020.

[3] R. Azevedo, I. Medeiros, and A. Bessani, "PURE: Generating quality threat intelligence by clustering and correlating OSINT," in *Proc. 18th IEEE Int. Conf. Trust Secur. Privacy Comput. Commun/ 13th IEEE Int. Conf. Big Data Sci. Eng.*, 2019, pp. 483–490.

[4] Lockheed Martin Corporation, "Gaining the advantage: Applying cyber kill chain methodology to network defense," 2015. [Online]. Available: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Gaining_the_Advantage_Cyber_Kill_Chain.pdf

[5] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "MITRE ATT&CK: Design and philosophy," 2018. [Online]. Available: https://www.mitre.org/sites/default/files/publications/pr-18-0944-11-mitre-attack-design-and-philosophy.pdf

[6] A. Mandiant, "Exposing one of China's cyber espionage units," 2013. Accessed: Aug. 24, 2020. [Online]. Available: http://intelreport.mandiant.com/Mandiant_APT1_Report.pdf

[7] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "TTPDRILL: Automatic and accurate extraction of threat actions from unstructured text of CTI sources," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, 2017, pp. 103–115.

[8] G. Husari, X. Niu, B. Chu, and E. Al-Shaer , "Using entropy and mutual information to extract threat actions from cyber threat intelligence," in *Proc. IEEE Int. Conf. Intell. Secur. Inform.*, 2018, pp. 1–6.

[9] S. Zhou, Z. Long, L. Tan, and H. Guo, "Automatic identification of indicators of compromise using neural-based sequence labelling," in *Proc. 32nd Pacific Asia Conf. Lang. Inf. Comput.*, 2018.

[10] Z. Zhu and T. Dumitras, "ChainSmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2018, pp. 458–472.

[11] Y. Gao, X. LI, H. PENG, B. Fang, and P. Yu, "HinCTI: A Cyber threat intelligence modeling and identification system based on heterogeneous information network," *IEEE Trans. Knowl. Data Eng.*, 2020, doi: 10.1109/TKDE.2020.2987019.

[12] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. N. Venkatakrishnan, "HOLMES: Real-time APT detection through correlation of suspicious information flows," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 1137–1152.

[13] W. U. Hassan, A. Bates, and D. Marino, "Tactical provenance analysis for endpoint detection and response systems," in *Proc. IEEE Symp. Secur. Privacy*, 2020, pp. 1172–1189.

[14] *Cuckoo Sandbox*. [Online]. Available: https://cuckoosandbox.org/

[15] *The Sandbox*. [Online]. Available: https://cwsandbox.org/

[16] S.-W. Hsiao, Y. S. Sun, and M. C. Chen, "Hardware-assisted MMU redirection for in-guest monitoring and API profiling," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2402–2416, Jan. 2020.

[17] Q. Wang *et al.*, "You are what you do: Hunting stealthy malware via data provenance analysis," in *Proc. Symp. Netw. Distrib. Syst. Secur.*, 2020, doi: 10.14722/ndss.2020.24167.

[18] J. Xu, Y. Li, R. Deng, and K. Xu, "SDAC: A slow-aging solution for android malware detection using semantic distance based API clustering," *IEEE Trans. Dependable Secure Comput.*, 2020, doi: 10.1109/TDSC.2020.3005088}.

[19] M. Mimura and H. Tanaka, "Heavy log reader: Learning the context of cyber attacks automatically with paragraph vector," in *Proc. Int. Conf. Inf. Syst. Secur.*, 2017, pp. 146–163.

[20] Y. Shen and G. Stringhini, "ATTACK2VEC: Leveraging temporal word embeddings to understand the evolution of cyberattacks," in *Proc. 28th USENIX Conf. Secur. Symp.*, 2019, pp. 905–921.

[21] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–42, 2018.

[22] K. Xu *et al.*"Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.

[23] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," 2017, *arXiv:1612.08220*.

[24] E. Choi, M. T. Bahadori, J. A. Kulas, A. Schuetz, W. F. Stewart, and J. Sun, "RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, 2016, pp. 3512–3520.

[25] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and classification of malware behavior," in *Proc. Int. Conf. Detection Intrusions Malware Vulnerability Assessment*, 2008, pp. 108–125.

1. https://attack.mitre.org/versions/v6/techniques/T1086/
2. https://attack.mitre.org/versions/v7/techniques/T1059/001/

[26] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, and E. Kirda, "Scalable, behavior-based malware clustering," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2009, pp. 8–11.
[27] T. Wüchner, A. Cisłak, M. Ochoa, and A. Pretschner, "Leveraging compression-based graph mining for behavior-based malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 99–112, Jan./Feb. 2017.
[28] *JCry*. [Online]. Available: https://attack.mitre.org/software/S0389/
[29] S. L. Lee, "CB TAU threat intelligence notification: JCry ransomware pretends to be adobe flash player update installer." [Online]. Available: https://www.carbonblack.com/blog/cb-tau-threat-intelligence-notification-jcry-ransomware-pretends- to-be-adobe-flash-player-update-installer/
[30] K. Oosthoek and C. Doerr, "SoK: ATT&CK techniques and trends in windows malware," in *Proc. Int. Conf. Secur. Privacy Commun. Syst.*, 2019, pp. 406–425.
[31] A. J. Viera *et al.*, "Understanding interobserver agreement: The kappa statistic," *Family Med.*, vol. 37, no. 5, pp. 360–363, 2005.
[32] B. S. Everitt, *The Analysis of Contingency Tables*. London, U.K./Boca Raton, FL, USA: Chapman and Hall/CRC, 2019.
[33] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.
[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *Proc. NIPS Workshop Deep Learn.*, 2014.
[35] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2017, pp. 2482–2486.
[36] H. Zhou, X. Yang, H. Pan, and W. Guo, "An android malware detection approach based on SIMGRU," *IEEE Access*, vol. 8, pp. 148404–148410, 2020.
[37] *MalShare*. [Online]. Available: https://malshare.com/
[38] *VirusTotal*. [Online]. Available: https://www.virustotal.com/
[39] *GenSim. Doc2vec paragraph embeddings.* [Online]. Available: https://radimrehurek.com/gensim/models/doc2vec.html
[40] F. Pedregosa *et al.*"Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.
[41] G. E. Dahl, J. W. Stokes, L. Deng, and D. Yu, "Large-scale malware classification using random projections and neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2013, pp. 3422–3426.
[42] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2015, pp. 1916–1920.
[43] K. Sethi, R. Kumar, L. Sethi, P. Bera, and P. K. Patra, "A novel machine learning based malware detection and classification framework," in *Proc. Int. Conf. Cyber Secur. Protection Digit. Services*, 2019, pp. 1–4.
[44] I. Jolliffe, "Principal component analysis," *Technometrics*, vol. 45, no. 3, 2003, Art. no. 276.
[45] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 1947.
[46] K. Krippendorff, "Reliability in content analysis: Some common misconceptions and recommendations," *Hum. Commun. Res.*, vol. 30, no. 3, pp. 411–433, 2004.
[47] *APT29 Group*. [Online]. Available: https://attack.mitre.org/groups/G0016/
[48] *APT29 Emulation - Enterprise Evaluation*, 2019. [Online]. Available: https://attackevals.mitre-engenuity.org/APT29/
[49] F-Secure Labs., "The Dukes: 7 years of Russian cyberespionage," *Retrieved Aug.*, 2015. [Online]. Available: https://blog-assets.f-secure.com/wp-content/uploads/2020/03/18122307/F-Secure_-Dukes_Whitepaper.pdf
[50] S. Adair, "PowerDuke: widespread post-election spear phishing campaigns targeting think tanks and NGOs." 2016. [Online]. Available: https://www.volexity.com/blog/2016/11/09/powerduke-post-election-spear-phishing-campaigns-targeting-think-tanks-and-ngos/
[51] M. Faou, M. Tartare, and T. Dupuy, "ESET research White papers: Operation ghost. The Dukes aren't back – they never left." 2019. [Online]. Available: https://www.welivesecurity.com/wp-content/uploads/2019/10/ESET_Operation_Ghost_Dukes.pdf
[52] J. Pastor-Galindo, P. Nespoli, F. G. Mármol, and G. M. Pérez, "The not yet exploited goldmine of OSINT: Opportunities, open challenges and future trends," *IEEE Access*, vol. 8, pp. 10282–10304, 2020.
[53] W. Tounsi and H. Rais, "A survey on technical threat intelligence in the age of sophisticated cyber attacks," *Comput. Secur.*, vol. 72, no. C, pp. 212–233, 2018.
[54] S. Qamar, Z. Anwar, M. A. Rahman, E. Al-Shaer, and B.-T. Chu, "Data-driven analytics for cyber-threat intelligence and information sharing," *Comput. Secur.*, vol. 100, no. 67, pp. 35–58, 2017.
[55] T. D. Wagner, K. Mahbub, E. Palomar, and A. E. Abdallah, "Cyber threat intelligence sharing: Survey and research directions," *Comput. Secur.*, vol. 87, 2019, Art. no. 101589.
[56] I. Vacas, I. Medeiros, and N. Neves, "Detecting network threats using OSINT knowledge-based IDS," in *Proc. 14th Eur. Dependable Comput. Conf.*, 2018, pp. 128–135.
[57] Y. Li, W. Dai, J. Bai, X. Gan, J. Wang, and X. Wang, "An intelligence-driven security-aware defense mechanism for advanced persistent threats," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 646–661, Mar. 2019.
[58] A. Saracino, D. Sgandurra, G. Dini, and F. Martinelli, "MADAM: Effective and efficient behavior-based android malware detection and prevention," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 1, pp. 83–97, Jan./Feb. 2018.
[59] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in *Proc. 6th Joint Meeting Eur. Softw. Eng. Conf. ACM Symp. Found. Softw. Eng.*, 2007, pp. 5–14.
[60] M. Fredrikson, S. Jha, M. Christodorescu, R. Sailer, and X. Yan, "Synthesizing near-optimal malware specifications from suspicious behaviors," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 45–60.
[61] S. Palahan, D. Babić, S. Chaudhuri, and D. Kifer, "Extraction of statistically significant malware behaviors," in *Proc. 29th Annu. Comput. Secur. Appl. Conf.*, 2013, pp. 69–78.
[62] V. Legoy, M. Caselli, C. Seifert, and A. Peter, "Automated retrieval of ATT&CK tactics and techniques for cyber threat reports," 2020, *arXiv:2004.14322*.
[63] R. Al-Shaer, J. M. Spring, and E. Christou, "Learning the associations of MITRE ATT&CK adversarial techniques," in *Proc. IEEE Conf. Commun. Netw. Sec.*, 2020, pp. 1–9, doi: 10.1109/CNS48642.2020.9162207.
[64] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. OReilly , "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. IEEE Secur. Privacy Workshops*, 2018, pp. 76–82.
[65] H. Li, S. Zhou, W. Yuan, X. Luo, C. Gao, and S. Chen, "Robust android malware detection against adversarial example attacks," in *Proc. Web Conf.*, 2021, pp. 3603–3612.
[66] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," in *Proc. Conf. Empir. Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 11–20.
[67] G. Chrysostomou and N. Aletras, "Improving the faithfulness of attention-based explanations with task-specific information for text classification," in *Proc. 59th Ann. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Language Process. (Volume 1: Long Papers*, 2021, pp. 477–488. [Online]. Available: https://aclanthology.org/2021.acl-long.40, doi: 10.18653/v1/2021.acl-long.40.

Yi-Ting Huang received the PhD degree in information management from National Taiwan University, Taipei, Taiwan, in 2015, and is now a postdoctoral fellow with the Institute of Information Science, Academia Sinica. Her research interests include malware analysis, deep learning, and natural language processing in educational applications.

Chi Yu Lin is currently working toward the master's degree in the Data Science Degree Program, National Taiwan University, Taipei, Taiwan, and Academia Sinica, Taipei, Taiwan, devoted to research methods in data mining, deep learning, and cybersecurity.

Ying-Ren Guo is currently working toward the master's degree in the Department of Electrical Engineering: Master Program of Cybersecurity, National Taiwan University, Taipei, Taiwan. His research interests include malware reverse engineering and analysis, and cybersecurity.

Kai-Chieh Lo received the graduate (MS program) degree in computer science from the National Tsing Hua University, Hsinchu, Taiwan, in 2019. He is currently a research assistant with Academia Sinica. His research interests include image processing and information security with deep learning.

Yeali S. Sun received the BS degree from the Computer Science and Information Engineering Department, National Taiwan University, Taipei, Taiwan, and the MS and PhD degrees in computer science from the University of California, Los Angeles (UCLA), Los Angeles, California, in 1984 and 1988, respectively. From 1988 to 1993, she was with Bell Communications Research Inc. In August 1993, she jointed National Taiwan University and is currently a professor with the Department of Information Management. Her research interests include Internet security and forensics, quality of service (QoS), cloud computing and services, and performance modeling and evaluation.

Meng Chang Chen received the PhD degree in computer science from the University of California, Los Angeles, California, in 1989. He was with AT&T Bell Labs and currently he is a research fellow/professor with the Institute of Information Science, Academia Sinica, Taiwan. His research interests include computer and network security, wireless network, deep learning for complicated applications, data and knowledge engineering.