




# Detecting Cryptomining Malware: a Deep Learning Approach for Static and Dynamic Analysis

Hamid Darabian · Sajad Homayounoot · Ali Dehghantanha · Sattar Hashemi ·  
Hadis Karimipour · Reza M. Parizi · Kim-Kwang Raymond Choo 

Received: 16 August 2019 / Accepted: 7 January 2020  
© Springer Nature B.V. 2020

**Abstract** Cryptomining malware (also referred to as cryptojacking) has changed the cyber threat landscape. Such malware exploits the victim's CPU or GPU resources with the aim of generating cryptocur-

rency. In this paper, we study the potential of using deep learning techniques to detect cryptomining malware by utilizing both static and dynamic analysis approaches. To facilitate dynamic analysis, we establish an environment to capture the system call events of 1500 Portable Executable (PE) samples of the cryptomining malware. We also demonstrate how one can perform static analysis of PE files' opcode sequences. In our study, we evaluate the performance of using Long Short-Term Memory (LSTM), Attention-based LSTM (ATT-LSTM), and Convolutional Neural Networks (CNN) on our sequential data (opcodes and system call invocations) for classification by a Softmax function. We achieve an accuracy rate of 95% in the static analysis and an accuracy rate of 99% in the dynamic analysis.

**Keywords** CryptoMining malware · Deep learning · Static analysis · Dynamic analysis

H. Darabian · S. Hashemi (✉)  
Department of Computer Science and Engineering, Shiraz  
University, Shiraz, Iran  
e-mail: s\_hashemi@shirazu.ac.ir

H. Darabian  
e-mail: h.darabian@cse.shirazu.ac.ir

S. Homayounoot  
IT and Computer Engineering Faculty, Shiraz University of  
Technology, Shiraz, Iran  
e-mail: s.homayoun@sutech.com

A. Dehghantanha  
Cyber Science Lab, School of Computer Science, Univer-  
sity of Guelph, Guelph, ON, Canada  
e-mail: adeghan@uoguelph.ca

H. Karimipour  
School of Computer Science, University of Guelph,  
Guelph, ON, Canada  
e-mail: hkarimi@uoguelph.ca

R. M. Parizi  
Department of Software Engineering and Game Develop-  
ment, Kennesaw State University, Marietta, GA 30060,  
USA  
e-mail: rparizi1@kennesaw.edu

K.-K. R. Choo  
Department of Information Systems and Cyber Security,  
University of Texas at San Antonio, San Antonio, TX  
78249, USA  
e-mail: raymond.choo@fulbrightmail.org

## 1 Introduction

Most cyber attackers and threat actors are financially motivated and likely attempt to maximize their monetary gain from available targets, for example by stealing credit card information [3, 11, 12, 42, 46]. In more recent years, there has been a shift in the malware trend, partly fueled by the introduction and popularity of cryptocurrencies [31, 44]. For example instead of directly stealing from the victims, cyber criminals

are now compromising victim's systems to use such compromised systems to form a network of systems (i.e. cryptomining pool) and mine crypto-currencies (e.g. Bitcoin), without the victims' knowledge – also referred to as cryptomining attacks [5, 8–10, 34, 38]. By compromising the victims' systems, such malware may also open up other vulnerabilities that can be exploited to facilitate other malicious activities. Hence, it is necessary for organizations and individual users to understand the risks and implement relevant mitigation strategies [17].

Malware detection approaches can be either static or dynamic [16, 27, 51]. Static malware detection relies on features extracted from executable artifacts such as opcodes, bytecodes, or strings, while dynamic techniques are based on behavioral features from system calls, captured network traffic, etc [18]. Indicator Of Compromise (IOC) is considered as a primitive approach for early detection of new malware attacks, where IOCs can be obtained from the malware code or the behavior of malware at runtime [22]. Zimba [56], for example, proposed network traffic-based and file-system based indicator using both static and dynamic analysis. The IOCs in the network traffic can include IP addresses of command and control (C&C) servers, the associated domain names, cryptomining instructions in cleartext, and file-system IOCs (e.g. cryptographic hashes of downloaded files and cryptographic hashes of malware scripts or binaries). Draghicescu [19] introduced using CPU usage processes that have access to the network, as a fingerprint of cryptominer malware behavior. As IOCs and fingerprinting methods are known as signature-based techniques (a signature is extracted from the propagated malware file), modifying malware by obfuscation methods or changing the C&C servers can easily prevent the detection of known cryptominer malware [45]. In addition, modern malware authors may also develop single-use or limited-use malware that are not captured by existing signature-based methods [39]. Therefore, we need more intelligent defense systems to detect such attacks. Detection systems based on Artificial Intelligence (AI; broadly defined to include machine learning and deep learning) do not rely on signatures; hence, they are more reliable and effective in discovering modern attacks [15].

Although conventional machine learning (ML) algorithms have demonstrated their potential in many different applications, they may not effectively solve

sequential problems with significant amount of information [29, 30, 40, 47]. Deep learning approaches, on the other hand, have the potential to obtain better representation for much more complex data (particularly sequential data), which are free from hand-crafted features and feature engineering [52]. Detecting cryptomining malware using AI-based static or dynamic analysis of Program Executable (PE) appears to be an understudied topic, at the time of this research. Therefore, this paper investigates the possibility of detecting cryptomining malware using AI-based static and dynamic analysis of PE samples. Specifically, we use deep learning techniques, i.e. Long Short-Term Memory (LSTM), Attention-based LSTM (ATT-LSTM), and Convolutional Neural Networks (CNN), on sequences of opcodes and system call events.

The remainder of this paper is organized as follows. In the next section, we present our proposed approach. Sections 3 and 4 describe our evaluation setup and findings, respectively. In the evaluation, we achieve an accuracy rate of 95% in the static analysis and an accuracy rate of 99% in the dynamic analysis. Section 5 concludes this paper.

## 2 Hunting Cryptomining Malware Using Deep Learning

In recent years, there has been a focus to develop more powerful models for training from sequential data. As our prepared datasets are sequential (see Section 3), we train effective models to apply on our sequences of opcodes and system call events to detect cryptomining malware.

As previously discussed, we utilized LSTM [20], ATT-LSTM [48], and CNN [35] to build our deep learning structure. Both LSTM and ATT-LSTM are Recurrent Neural Network (RNN) models, which can be used to process sequential data [53]. While CNN benefits from local coherence in the input (often image), CNN only considers the current input. On the other hand, RNN models (especially LSTM) consider both previously received inputs and current input. Although CNN is usually known for its performance on image classification problems, it is applicable to sequential data by applying 1D convolution layer [4]. We also used Google Tensor Flow [1] as the back-end structure of the deep learning approach to perform model evaluation tasks.

As the length of sequences may be unequal, we performed a zero-padding operation on sequences with length less than the median of all sequences. We also truncated sequences with larger length to create our final dataset of sequences. In the case of opcodes dataset, we converted sequences ( $S_i$ ) of opcodes ( $opc_{i,j}$ ) into equivalent sequences of scalar values with mapped opcodes ( $M_{opc_{i,j}}$ ), as neural networks require numerical representations. To map each  $opc_{i,j}$  into a scalar value ( $M_{opc_{i,j}}$ ), there are a number of methods that can be deployed (one-hot encoding, bag of word, word2vec, Glove, etc). In this paper, we used Glove method [37], which is an unsupervised learning algorithm for obtaining vector representations for words. Glove learns a new vector representation for each word by random initialization. Then, during the backpropagation process, word's vectors that are in a context will have close proximities in the vector space. If  $S_i = \{M_{opc_{i,1}}, M_{opc_{i,2}}, \dots, M_{opc_{i,p}}\}$  is the input sequence, the output of Glove embedding with  $d$  dimensions would be in

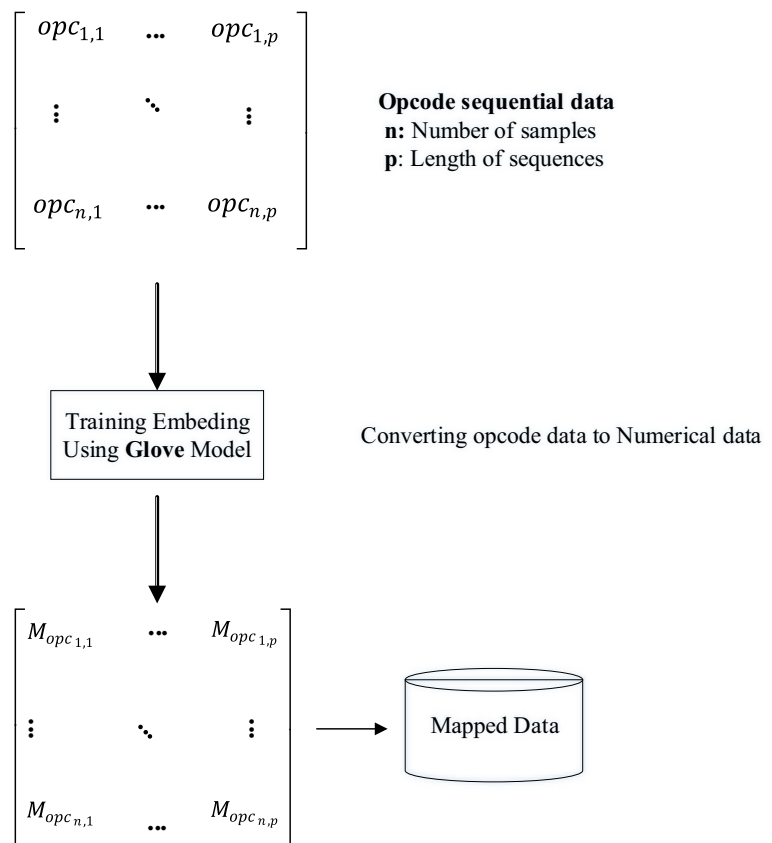
the form of  $\{Z_1, Z_2, \dots, Z_p\}$ , where  $Z_i = \{z_1, z_2, \dots, z_d\}$  is the embedded vector. Figure 1 shows the embedding process for opcodes sequences. Similar to opcodes analysis, Glove method is applied for system calls to convert events to numerical values (see Fig. 2).

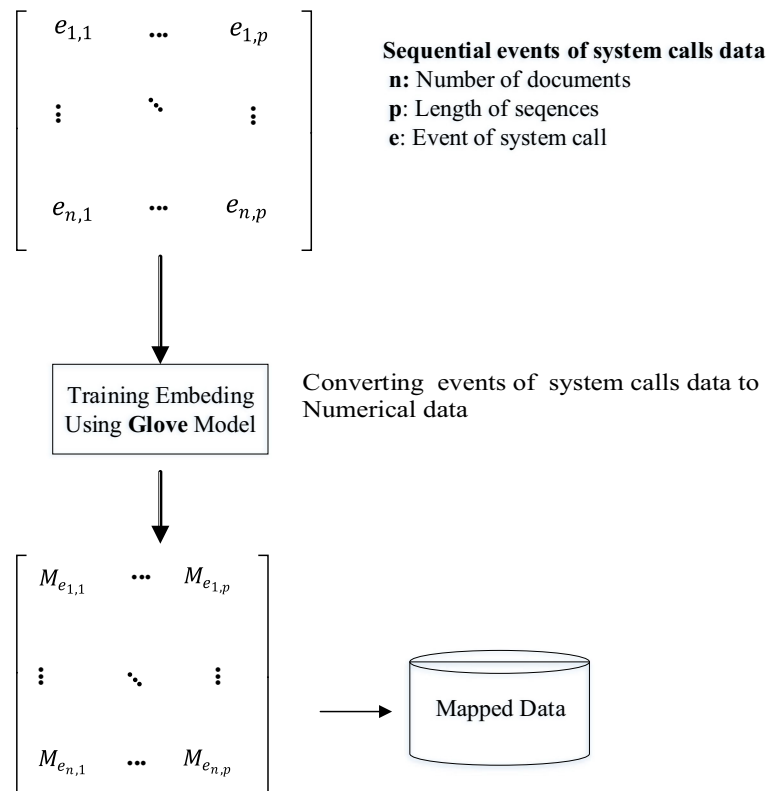
## 2.1 LSTM

LSTM has been widely used for speech recognition [21], language modeling [43], sentiment analysis [32], text prediction [54], and any other context with sequential data, especially opcode sequences. Since LSTM structure is capable of learning dependencies between our datasets, it can be used in opcodes or system calls event as our prepared dataset follow a sequential form. Figure 3 shows the steps to train a classifier with the capability of distinguishing cryptomining from benign samples.

The LSTM layer in Fig. 3 includes several LSTM units followed by a SoftMax [6] function to assign a probability to each class for classifying instances.

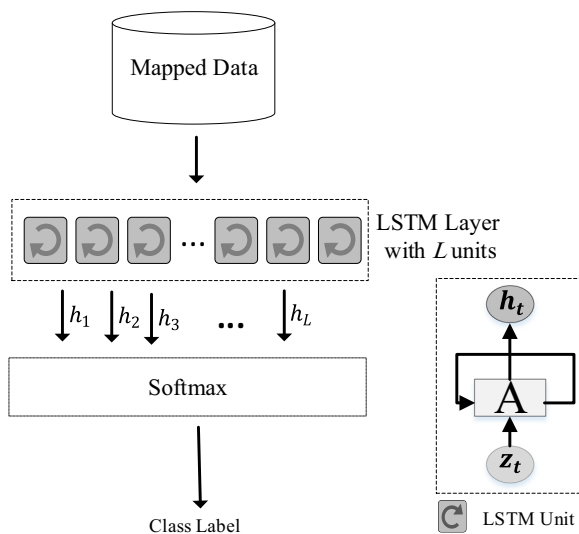
**Fig. 1** Embedding process on opcodes



**Fig. 2** Embedding process on the system call events

## 2.2 Attention Based LSTM

One challenge in the NLP systems is that important information can appear at any position in the sentences

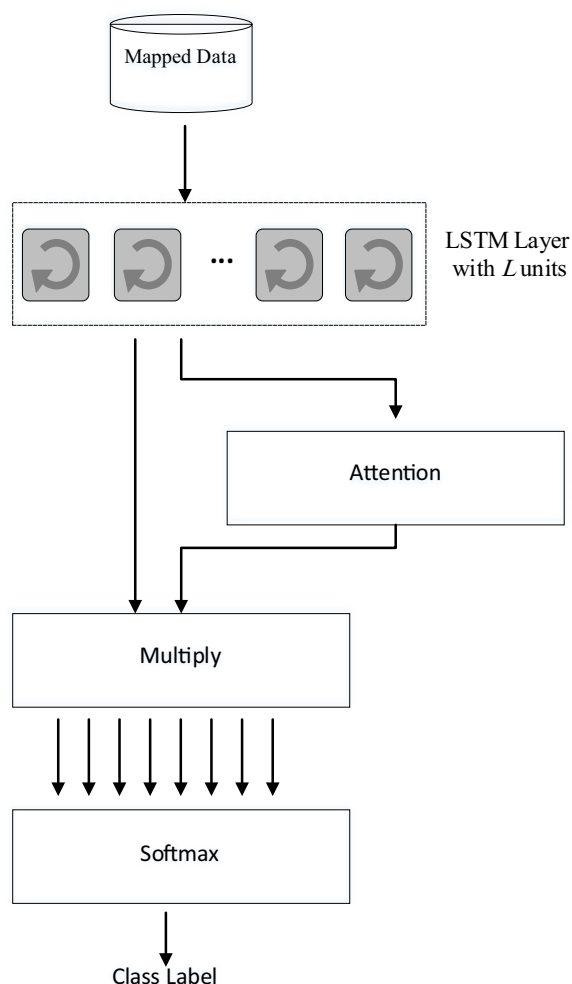
**Fig. 3** LSTM model for detecting cryptomining malware

[55]. In the field of malware detection, after preprocessing and extracting opcodes (or even systems calls, etc.), a sequence of opcodes makes the samples and essential information which can cause discrimination between the classes of data appear in each part of the sample.

Attentive neural networks have been used recently in a wide range of NLP tasks from machine translations [2] to speech recognition [13] to question answering systems [24] to image captioning [49], and so on. The main idea behind the attention-based model is that the available words in the samples are not equally important. Therefore, we use attention models to get more concentration on important words by calculating a weight for each word.

Attention model used in this paper comprises both LSTM and attention network. While LSTM learns the relation between words in a sequence and returns a vector for every word, the attention network indicates the importance of each word by calculating their weights [50].

Figure 4 shows the architecture of our considered attention-based LSTM (ATT-LSTM) for this research.

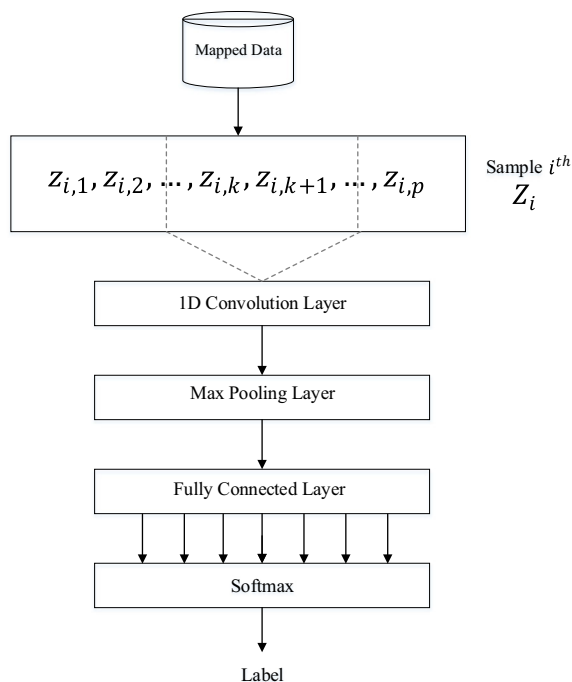


**Fig. 4** ATT-LSTM model for detecting cryptomining malware

In the first step, it receives the embedded data and utilizes an LSTM layer to produce word level features. Then, the attention layer converts word level features into sample level feature by calculating the weighted sum of each vector. Finally, a Softmax uses the sample level feature vectors for final classification.

### 2.3 CNN

We also used CNN to detect cryptomining malware. CNN can work on sequential data by utilizing one or more 1D sliding window to extract precious features from sequential data. Figure 5 depicts our considered



**Fig. 5** CNN model for detecting cryptomining malware

CNN model, which consists of a 1D convolution layer, a max pooling layer, and a fully connected layer [14] for extracting valuable features and training our classifier. The embedding layer is performed using the Glove method, as shown in Fig. 1. We used the configurations from our previous work [33] depicted in Fig 5.

### 3 Dataset Preparation and Environment Setup

Typically when machine learning algorithms are applied, it is essential to represent the data in some appropriate form. To create our dataset, we focused on Windows Portable Executable (PE32) cryptominer samples registered with virustotal.com in 2018. The downloaded samples are designed to mine different cryptocurrencies especially Bitcoin and Monero. Due to the domain-name filtering or inconsistent execution platform (or Operating System), some of the collected samples failed to perform mining. Therefore, we selected 1500 active samples as our final dataset

of malware. For this purpose, Cuckoo Sandbox<sup>1</sup> was used.

We built the environment shown in Fig. 6 to collect reports of cryptomining samples at runtime. Cuckoo Sandbox is the leading open-source malware analysis tool that allows getting the detailed behavioral reports of any file or URL in a matter of seconds. The Cuckoo host is responsible for guest and manages analysis tasks, collects network traffic, and generates behavioral reports. Cuckoo guest is a clean environment connected to Cuckoo host through a virtual switch to execute a sample and send collected reports to the host. The host system passes the traffics generated by the guest to the Internet to provide the possibility of testing malware samples requiring an Internet connection. In this paper, we used Cuckoo Sandbox 2.0.6 installed on Ubuntu 16.04 Desktop on a system with Core i5 CPU and 8 GB of RAM. We installed Windows 10 on Virtualbox 6.0 and disabled all applications/configurations making noisy network traffics such as Windows Update Manager.

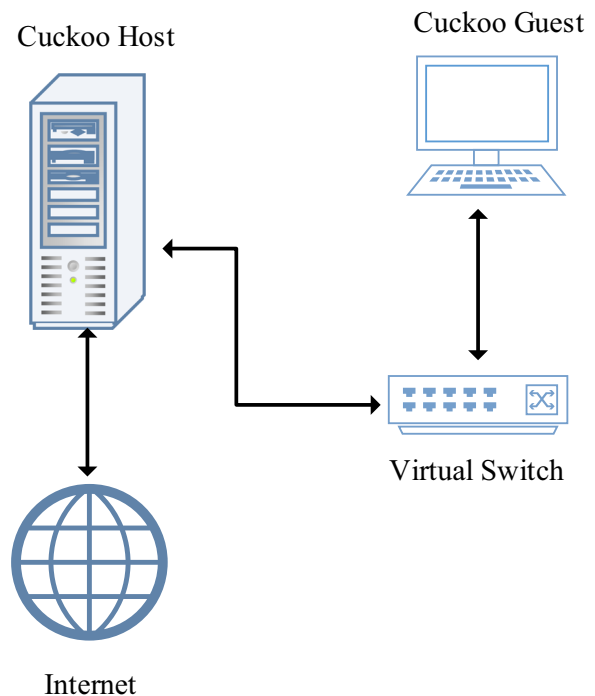
The reports generated by the sandbox, describing the behavioral data of each sample were preprocessed, and malware features were extracted from there. In the following, we describe the tools and preprocessing task to provide appropriate data for static and dynamic analysis.

### 3.1 System Calls Data

One of the most important symptoms of cryptominer malware behavior is the frequent calls of cryptographic libraries so that benign programs may not use these library functions to such an extent. Table 1 describes some of the Windows cryptographic libraries, which may be used by a cryptominer malware.

We have created a sequence of runtime events (system calls) for each cryptomining malware. Therefore, we have a dataset in the form of  $D = \{S_1, S_2, \dots, S_n\}$  which  $S_i$  represents a sequence of all events by launching sample  $i$ .

As all of our selected cryptominer programs are standalone samples, portable and standalone benign apps are suitable for a normal dataset. Most of the benign applications support graphical user interface, and they need human activities such as clicking on



**Fig. 6** Environment setup for executing samples

a button to show their normal behavior (generate a sequence of normal events). Homayoun et al. [25] developed a python tool named PyWinMonkey to mimic human actions for launching desktop applications. Therefore, we use their benign dataset consisting of the system calls from 220 portable applications as our normal dataset. Table 2 shows the final datasets of system calls with the number of sequences in each dataset.  $D_{crypto\_sys\_calls}$  represents system call sequences of cryptominer malware samples while

**Table 1** Some cryptographic libraries which a cryptomining malware uses

Library	Description
bcryptprimitives.dll	Windows cryptographic primitives library
crypt32.dll	Crypto API32
cryptbase.dll	Base cryptographic API DLL
cryptdll.dll	Cryptography manager
cngaudit.dll	Windows cryptographic next generation audit library
rsaenh.dll	Microsoft enhanced cryptographic provider
cryptsp.dll	Cryptographic Service Provider API
ncrypt.dll	Windows cryptographic library

<sup>1</sup><http://https://cuckoosandbox.org>

**Table 2** System call events dataset

Dataset	Number of sequences
$D_{crypto.sys.calls}$	1500
$D_{benign.sys.calls}$	220

$D_{benign\_sys\_calls}$  represents system call sequences of normal samples.

### 3.2 Opcodes Data

An opcode specifies an atomic operation to be performed in an application. As applications opcodes have been widely employed in static analysis of malware samples, we evaluate the possibility of distinguishing cryptominer malware from benign samples. We extracted opcodes of each sample by disassembling them using IDA Pro version 7.0 (Fig. 7). We pruned the output in a sequential order to create our datasets. As opcode analysis is a static approach and all files can be subjected to the static analysis, we also considered Dynamic Link Library (DLL) of Microsoft Windows files as well as executable files to provide our benign dataset. We gathered our collection of executable and DLL samples [23], which are known harmless, from a standard Windows installation. Their hashes were cross checked with ESET Nod32<sup>2</sup> and Kaspersky<sup>3</sup> to ensure they were not malicious.

At the end of this session, we created two datasets of sequences, as shown in Table 3.  $D_{crypto\_opcodes}$  denotes opcode sequences of cryptomining malware samples, and  $D_{benign\_opcodes}$  contains all opcode sequences of benign samples.

## 4 Findings

This section investigates the possibility of detecting cryptomining malware samples by training and evaluating each prepared dataset in Section 3. We first trained the deep learning models on opcodes and analyzed the results; then we applied deep learning models on system calls dataset and performed the analysis. All experiments were performed on a machine with Intel Core i7-930@2.6GHz processor with 8 threads,

<sup>2</sup><http://www.eset.com>

<sup>3</sup><http://www.kaspersky.com>

```
.text:00401000      push     lpModuleName      ; lpModuleName
.text:00401006      call    ds:GetModuleHandleA
.text:0040100C      push     lpProcName       ; lpProcName
.text:00401012      mov     dword_40427C, eax
.text:00401017      push     eax              ; hModule
.text:00401018      call    ds:GetProcAddress
.text:0040101E      push     off_4040C8
.text:00401024      mov     dword_40424C, eax
.text:00401029      push     dword_40427C
.text:0040102F      call    eax
.text:00401031      push     off_40407C
.text:00401037      mov     dword_404248, eax
.text:0040103C      push     dword_40427C
.text:00401042      call    dword_40424C
.text:00401048      push     off_404088
.text:0040104E      mov     dword_4041F8, eax
.text:00401053      push     dword_40427C
.text:00401059      call    dword_40424C
.text:0040105F      push     off_404150
.text:00401065      mov     dword_404258, eax
.text:0040106A      push     dword_40427C
.text:00401070      call    dword_40424C
.text:00401076      mov     dword_40420C, eax
.text:0040107B      ret     0
```

**Fig. 7** The output of IDA Pro disassembler consists of irrelevant data such as operands and line of codes

8.0GB of RAM and 7200RPM HDD running Ubuntu 16.10. We installed Python 3.7, and sklearn machine-learning python package [36] for classification tasks.

## 4.1 Models Configuration

We used Adam as the weight updating algorithm in all deep models for our evaluation. The Adam algorithm is an optimizer for stochastic gradient descent [28]. Because overfitting is a common problem in deep neural networks, we utilized the dropout [41] technique with 0.3 value to avoid this problem. We set the window size to 5 in the CNN model and 100 in LSTM and ATT-LSTM models. Batch size (number of samples that pass through a neural network in each propagation

**Table 3** System call events dataset

Dataset	Number of sequences
$D_{\text{crypto\_opcodes}}$	1500
$D_{\text{benign\_opcodes}}$	1500



to feed each model and find the optimum parameters) was set to 20 for all configurations. Finally, we used categorical cross-entropy as the loss function of deep models.

## 4.2 Evaluation Metrics

Similar to existing approaches, such as those of [33, 51], we used the well-known metrics of True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) to evaluate performance of the classifiers. TP denotes malware samples that are correctly predicted, TN denotes the number of samples correctly identified as benign, FP denotes benign samples incorrectly detected as malware, and FN denotes malware samples incorrectly identified as benign. Accuracy, precision, recall, and f-measure metrics are calculated using these four criteria. Accuracy is the proportion of the correct results (malware and benign) in the observed samples (see (1)). The precision of a classifier reflects the ratio of correctly predicted malware samples to the total predicted malware observation (see (2)). Recall or sensitivity is the fraction of successful prediction of relevant malware samples (see (3)). F-measure is indicating the performance of the classification algorithm (see (4)).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F - measure = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

To ensure the classifiers are far from a random classifier, we also compute Matthew Correlation Coefficient (MCC) values for our experiments (using (5)) [26]. MCC provides a measurement of the quality of binary classification [7]. While precision, recall, F-measure or accuracy metrics in a random guessing would be higher than 0.5, the MCC value would be around 0 for random guessing. Therefore, MCC is generally considered as a balance measurement that

can be applied on imbalanced datasets. MCC value can be between -1 and +1 for a classifier. While -1 shows an inverse prediction, +1 indicates a perfect prediction.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(FP + TP)(FN + TP)(FP + TN)(TN + FN)}} \quad (5)$$

## 4.3 Applying Deep models on Opcodes

After the embedding process presented in Fig. 1, we applied the LSTM, ATT-LSTM, and CNN presented in Figs. 3–5 on  $D_{opcodes}$  that is created by combining  $D_{crypto\_opcodes}$  and  $D_{benign\_opcodes}$ . We followed 10 times 3 folds evaluation where 70% of data was selected for training phase on 20 epochs, 15% of data was chosen for validation phase at the end of each epoch and the remainder of 15% was selected for the final test of the model after the training phase was completed. Table 4 shows the results of trained classifiers on the selected test data. The results show that LSTM classifier was somewhat better than CNN classifier since the LSTM is able to memorize previous inputs due to its internal memory. Moreover, ATT-LSTM outperformed our LSTM model as it considers the arrangement of opcodes in sequential order as well as assigning higher weights to more important opcodes.

## 4.4 Applying Deep Models on System Calls

We first created  $D_{sys\_calls}$  by combining  $D_{crypto\_sys\_calls}$  and  $D_{benign\_sys\_calls}$  as our main dataset of system calls and then embedding process introduced in the Fig. 2 was run on  $D_{crypto\_sys\_calls}$ . Similar to opcodes sequential structure, we applied deep learning models proposed in Figs. 3–5 on  $D_{crypto\_sys\_calls}$  to investigate the possibility of detecting cryptominer malware using system calls. To

**Table 4** Classifiers performance on the  $D_{opcodes}$

Classifier	Accuracy	F-measure	MCC	FPR
LSTM	0.94	0.94	0.92	0.06
ATT-LSTM	0.95	0.94	0.92	0.05
CNN	0.93	0.92	0.90	0.07



**Table 5** Classifiers performance on the  $D_{sys\_calls}$ 

Classifier	Accuracy	F-measure	MCC	FPR
LSTM	0.99	0.98	0.98	0.006
ATT-LSTM	0.99	0.988	0.99	0.006
CNN	0.99	0.97	0.98	0.007

apply the models on the sequences of system calls, we needed to perform a word embedding. Figure 2 shows the word embedding process on the sequence of system calls events. The evaluation method of classifiers is similar to what we have followed on opcodes. Table 5 presents the average performance of trained models that experimented on the unseen data. As can be seen in Table 5, it is clear that our trained classifiers showed high accuracies (0.99), high MCCs (about 0.99), and low false positive rates (less 0.007), which are close to perfect predictions for cryptomining malware detection. Although our dataset is unbalanced and accuracy rate may not be an ideal metric to judge the models, the high values of MCC and F-measure imply the effectiveness of models to detect cryptomining malware samples.

## 5 Conclusion

As cryptocurrency becomes increasingly popular, cryptomining and the associated malware will remain a growing threat. In this paper, we used deep learning techniques to facilitate the static and dynamic analysis of cryptomining malware. Specifically, for static analysis, using LSTM, Attention-based LSTM, and CNN on opcodes of the cryptomining malware we achieved an accuracy rate of 0.95 with a low false positive rate. In the dynamic analysis, we utilized the Cuckoo sandbox to execute the malware and capture sequences of system call events. Using the deep learning approaches on the sequences of system call events, we achieved a detection rate of 0.99 with a low false positive rate of about 0.006.

Future research will include exploring the use of other machine/deep learning techniques (e.g. multi-view learning) in our proposed approach to facilitate extraction of other features for classification.

**Acknowledgements** The authors thank the anonymous reviewers and the handling editor for providing constructive feedback.

## References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467 (2016)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014)
3. Bahrami, P.N., Dehghantanha, A., Dargahi, T., Parizi, R.M., Choo, K.R., Javadi, H.H.S.: Cyber kill chain-based taxonomy of advanced persistent threat actors: Analogy of tactics, techniques, and procedures. J. Inf. Process. Sys. **15**(4), 865–889 (2019). <https://doi.org/10.3745/JIPS.03.0126>
4. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
5. Baldwin, J., Dehghantanha, A.: Leveraging support vector machine for opcode density based detection of cryptoransomware. In: Cyber Threat Intelligence, pp. 107–136. Springer (2018)
6. Bishop, C.M.: Pattern Recognition and Machine Learning, chap. 2, pp. 113–116. Springer, Berlin (2006)
7. Boughorbel, S., Jarray, F., El-Anbari, M.: Optimal classifier for imbalanced data using matthews correlation coefficient metric. PLOS ONE **12**(6), e0177678 (2017). <https://doi.org/10.1371/journal.pone.0177678>
8. Brown, S.D.: Cryptocurrency and criminality. The Police Journal: Theory Practice and Principles **89**(4), 327–339 (2016). <https://doi.org/10.1177/0032258x16658927>
9. Carlin, D., O'kane, P., Sezer, S., Burgess, J.: Detecting cryptomining using dynamic analysis. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST), pp. 1–6. IEEE (2018)
10. Carlin, D., OrKane, P., Sezer, S., Burgess, J.: Detecting cryptomining using dynamic analysis. In: 2018 16th Annual Conference on Privacy, Security and Trust (PST). IEEE (2018). <https://doi.org/10.1109/pst.2018.8514167>
11. Choo, K.K.R., et al.: Cyber threat landscape faced by financial and insurance industry. Trends and issues in crime and criminal justice (408), 1–6 (2011)
12. Choo, K.R.: The cyber threat landscape: Challenges and future research directions. Computers & Security **30**(8), 719–731 (2011)
13. Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y.: Attention-based models for speech recognition. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, pp. 577–585. MIT Press, Cambridge (2015). <http://dl.acm.org/citation.cfm?id=2969239.2969304>
14. Cireşan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two, IJCAI'11, pp. 1237–1242. AAAI Press (2011). <https://doi.org/10.5591/978-1-57735-516-8/IJCAI11-210>
15. Conti, M., Dargahi, T., Dehghantanha, A.: Cyber Threat Intelligence: Challenges and Opportunities. Springer, Berlin (2018)

16. Costin, A., Zaddach, J.: Iot malware: Comprehensive Survey, Analysis Framework and Case Studies. BlackHat, USA (2018)
17. Courtois, N.T., Emirdag, P., Wang, Z.: On detection of bitcoin mining redirection attacks. In: 2015 International Conference on Information Systems Security and Privacy (ICISSP), pp. 98–105. IEEE (2015)
18. Darabian, H., Dehghantanha, A., Hashemi, S., Homayoun, S., Choo, K.K.R.: An opcode-based technique for polymorphic internet of things malware detection. *Concurrency and Computation: Practice and Experience*, pp. e5173. <https://doi.org/10.1002/cpe.5173> (2019)
19. Draghicescu, D., Caranica, A., Vulpe, A., Fratu, O.: Cryptomining application fingerprinting method. In: 2018 International Conference on Communications (COMM). IEEE (2018). <https://doi.org/10.1109/icomm.2018.8484745>
20. Gers, F.A., Schmidhuber, J., Cummins, F.: Learning to forget: Continual prediction with LSTM. *Neural Comput.* **12**(10), 2451–2471 (2000). <https://doi.org/10.1162/089976600300015015>
21. Graves, A., Jaitly, N., Mohamed, A.: Hybrid speech recognition with deep bidirectional LSTM. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding. IEEE (2013). <https://doi.org/10.1109/asru.2013.6707742>
22. Hasan, S., Alam, M., Khan, T., Javaid, N., Khan, A.: Extraction of malware iocs and ttps mapping with coas. *Computer and Cyber Security: Principles, Algorithm, Applications, and Perspectives*, p. 335 (2018)
23. Hashemi, H., Azmoodeh, A., Hamzeh, A., Hashemi, S.: Graph embedding as a new approach for unknown malware detection. *Journal of Computer Virology and Hacking Techniques* **13**(3), 153–166 (2016). <https://doi.org/10.1007/s11416-016-0278-y>
24. Hermann, K.M., Kočiský, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1, NIPS'15, pp. 1693–1701. MIT Press, Cambridge (2015). <http://dl.acm.org/citation.cfm?id=2969239.2969428>
25. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R.: Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence. *IEEE Transactions on Emerging Topics in Computing*, pp. 1–1. <https://doi.org/10.1109/tetc.2017.2756908> (2017)
26. Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R., Choo, K.K.R., Newton, D.E.: DRTHIS: Deep Ransomware threat hunting and intelligence system at the fog layer. *Futur. Gener. Comput. Syst.* **90**, 94–104 (2019). <https://doi.org/10.1016/j.future.2018.07.045>
27. Kananizadeh, S., Kononenko, K.: Predictive mitigation of timing channels - threat defense for machine codes. *J. Grid Comput.* **15**(3), 395–414 (2017)
28. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014)
29. Kovács, J.: Supporting programmable autoscaling rules for containers and virtual machines on clouds. *J. Grid Comput.* **17**(4), 813–829 (2019)
30. Längkvist, M., Karlsson, L., Loutfi, A.: A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recogn. Lett.* **42**, 11–24 (2014). <https://doi.org/10.1016/j.patrec.2014.01.008>
31. Parizi, R.M., Dehghantanha, A.: On the understanding of gamification in blockchain systems. In: 2018 6th International Conference on Future Internet of Things and Cloud Workshops (Ficloudw), pp. 214–219 (2018). <https://doi.org/10.1109/W-FiCloud.2018.00041>
32. Ma, Y., Peng, H., Cambria, E.: Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive Lstm. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
33. Milosevic, N., Dehghantanha, A., Choo, K.K.R.: Machine learning aided android malware classification. *Computers & Electrical Engineering* **61**, 266–274 (2017). <https://doi.org/10.1016/j.compeleceng.2017.02.013>
34. Mukhopadhyay, U., Skjellum, A., Hambolu, O., Oakley, J., Yu, L., Brooks, R.: A brief survey of cryptocurrency systems. In: 2016 14th Annual Conference on Privacy, Security and Trust (PST). IEEE (2016). <https://doi.org/10.1109/pst.2016.7906988>
35. O'Shea, K., Nash, R.: An introduction to convolutional neural networks. *arXiv:1511.08458* (2015)
36. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. *J. Mach. Learn. Res.* **12**(Oct), 2825–2830 (2011)
37. Pennington, J., Socher, R., Manning, C.: Glove: Global vectors for word representation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543 (2014)
38. Rütth, J., Zimmermann, T., Wolsing, K., Hohlfeld, O.: Digging into browser-based crypto mining. In: Proceedings of the Internet Measurement Conference 2018, pp. 70–76. ACM (2018)
39. Santos, I., Brezo, F., Nieves, J., Penya, Y.K., Sanz, B., Laorden, C., Bringas, P.G.: Idea: Opcode-sequence-based malware detection. In: *Lecture Notes in Computer Science*, pp. 35–43. Springer, Berlin (2010). [https://doi.org/10.1007/978-3-642-11747-3\\_3](https://doi.org/10.1007/978-3-642-11747-3_3)
40. Sniezynski, B., Nawrocki, P., Wilk, M., Jarzab, M., Zielinski, K.: VM Reservation plan adaptation using machine learning in cloud computing. *J. Grid Comput.* **17**(4), 797–812 (2019)
41. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **15**(1), 1929–1958 (2014)
42. Stokel-Walker, C.: Are you making cryptocurrency for crooks? *New Scientist* **237**(3161), 16 (2018). [https://doi.org/10.1016/s0262-4079\(18\)30115-5](https://doi.org/10.1016/s0262-4079(18)30115-5)
43. Sundermeyer, M., Schlüter, R., Ney, H.: Lstm neural networks for language modeling. In: Thirteenth Annual Conference of the International Speech Communication Association (2012)
44. Taylor, P.J., Dargahi, T., Dehghantanha, A., Parizi, R.M., Choo, K.K.R.: A systematic literature review of blockchain cyber security. *Digital communications and networks*. <https://doi.org/10.1016/j.dcan.2019.01.005>. <http://www>

- [sciencedirect.com/science/article/pii/S2352864818301536](https://www.sciencedirect.com/science/article/pii/S2352864818301536) (2019)
45. Vinod, P., Jaipur, R., Laxmi, V., Gaur, M.: Survey on malware detection methods. In: Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security (IITKHACK'09), pp. 74–79 (2009)
46. Vukalovic, J., Delija, D.: Advanced persistent threats - detection and defense. In: 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE (2015). <https://doi.org/10.1109/mipro.2015.7160480>
47. Wang, W., Zeng, G.: Bayesian cognitive model in scheduling algorithm for data intensive computing. *J. Grid. Comput.* **10**(1), 173–184 (2012)
48. Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 606–615 (2016). Association for Computational Linguistics, Austin, Texas. <https://doi.org/10.18653/v1/D16-1058>, <https://www.aclweb.org/anthology/D16-1058>
49. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y.: Show, attend and tell: Neural image caption generation with visual attention. In: Proceedings of the 32nd International Conference on Machine Learning, Proceedings of Machine Learning Research, vol. 37, pp. 2048–2057. PMLR <http://proceedings.mlr.press/v37/xuc15.html> (2015)
50. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489. Association for Computational Linguistics. <https://doi.org/10.18653/v1/n16-1174> (2016)
51. Ye, Y., Li, T., Adjeroh, D., Iyengar, S.S.: A survey on malware detection using data mining techniques. *ACM Comput. Surv.* **50**(3), 1–40 (2017). <https://doi.org/10.1145/3073559>
52. Yin, C., Zhu, Y., Fei, J., He, X.: A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access* **5**, 21954–21961 (2017). <https://doi.org/10.1109/access.2017.2762418>
53. Zhao, Z., Chen, W., Wu, X., Chen, P.C.Y., Liu, J.: LSTM Network: a deep learning approach for short-term traffic forecast. *IET Intell. Transp. Syst.* **11**(2), 68–75 (2017). <https://doi.org/10.1049/iet-its.2016.0208>
54. Zhou, C., Sun, C., Liu, Z., Lau, F.: A c-lstm neural network for text classification. *arXiv:1511.08630* (2015)
55. Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B.: Attention-based bidirectional long short-term memory networks for relation classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), vol. 2, pp. 207–212 (2016). <https://doi.org/10.18653/v1/p16-2034>
56. Zimba, A., Wang, Z., Mulenga, M., Odongo, N.H.: Crypto mining attacks in information systems: an emerging threat to cyber security. *J. Comput. Inf. Sys.* pp. 1–12. <https://doi.org/10.1080/08874417.2018.1477076> (2018)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.