Information Systems Frontiers
https://doi.org/10.1007/s10796-020-10017-4

DeepRan: Attention-based BiLSTM and CRF for Ransomware Early Detection and Classification

Krishna Chandra Roy[1] · **Qian Chen**[1]

Abstract

Ransomware is a self-propagating malware encrypting file systems of the compromised computers to extort victims for financial gains. Hundreds of schools, hospitals, and local government municipalities have been disrupted by ransomware that already caused 12.1 days of system downtime on average (Siegel 2019). This study aims at developing a deep learning-based detector **DeepRan** for ransomware early detection and classification to prevent network-wide data encryption. DeepRan applies an attention-based bi-directional Long Short Term Memory (BiLSTM) with a fully connected (FC) layer to model normalcy of hosts in an operational enterprise system and detects abnormal activity from a large volume of ambient host logging data collected from bare metal servers. DeepRan also classifies abnormal activity as one of the candidate ransomware attacks by extending attention-based BiLSTM with a Conditional Random Fields (CRF) model. The Term Frequency-Inverse Document Frequency (TF-IDF) method is applied to extract semantic information from high dimensional host logging data. An incremental learning technique is used to extend the model's existing knowledge to prevent DeepRan quality degradation over time. We develop a testbed of bare metal servers and collect normal host logs of two users for 63 days (IRB-approved). 17 ransomware attacks are executed on the victim hosts, and the infected host logging data is used for validating DeepRan. Experimental results present that DeepRan produces 99.87% detection accuracy (F1-score of 99.02%) for ransomware early detection. The detector also achieves 96.5% accuracy to classify abnormal events as one of 17 candidate ransomware families. The application of incremental learning is validated as an efficient technique to enhance model quality over time.

Keywords Ransomware detection · Classification · Testbed design · Dataset collection · Deep learning

1 Introduction

Ransomware, a type malware enabling cyber extortion for financial gain, is normally distributed and propagated by using social engineering (e.g., malspam and USB drop) techniques, exploiting network protocol vulnerabilities (e.g., SMB and RDP) and initiating drive-by downloads from compromised websites (Challita 2018). A total of 850.97 million ransomware infections were detected in 2018 (Dobran 2019). In contrast to the 2017 ransomware WannaCry that infected 300K machines across the globe, the majority of ransomware attacks in 2018 targeted small businesses (Davis 2019), and approximately two-thirds of 2019 ransomware attacks in the U.S. have targeted state and local governments (Shi 2019). According to Coveware's Q3 Ransomware Marketplace report (Siegel 2019), the average size of the companies that were attacked by ransomware in Q3 of 2019 was 645 employees, which is down from 925 in Q2 of 2019.

Modern ransomware utilizes advanced and sophisticated encryption algorithms, which are hard or impossible to regain access to data if victims refuse to pay the ransom. It is reported that nearly 40 percent of victims paid a relatively smaller amount of ransom to unlock their data than spending thousands to millions of dollars to rebuild their information systems and infrastructure (Cook 2019). In particular, we observed that critical infrastructure sectors such as healthcare and communication (i.e., local government) sectors had no choice but to pay off criminals to end the attacks. For example, three Alabama hospital

✉ Krishna Chandra Roy
krishna.roy@utsa.edu

Qian Chen
guenevereqian.chen@utsa.edu

[1] Electrical and Computer Engineering Department, University of Texas at San Antonio, 78249, San Antonio, TX, USA

Published online: 14 June 2020

computer systems were compromised by Ryuk ransomware attacks in October 2019, which resulted in diverting existing patients (expect the most critically ill patients) and turning away new patients. The hospital normal operation has been disrupted by the ransomware attack for more than a week before the hospitals paying off the ransom (Andone 2019). Big cities such as Atlanta and Baltimore can afford more than $10 million costs to recover their computer servers from ransomware attacks. Small cities usually face stricter financial constraints (e.g., Lake City and Riviera City in Florida (Cimpanu 2019)) quickly resumed their operations by paying the criminals.

Moreover, the highly profitable cyber extortion has brought forward the ransomware-as-a-service (RaaS) business model. Ransomware authors sell their malware for the criminals who are technically unable to develop their variants. Nowadays no enterprise systems are immune to those intensive and sophisticated ransomware attacks. Ransomware normally conducts a series of activity (e.g., checking keyboard layout, escalating privilege, changing registry, and loading *dll* files) before encrypting victim's data, which begs the question, "How to detect ransomware attacks as early as possible to prevent data loss and to stop ransomware self-propagation?" Ransomware's rapid evolution reveals that a newly found ransomware attack is possible to be a variant of existing (known) ransomware families. These specific properties of ransomware attacks motivate this study to develop a ransomware early detection and classification tool. To our knowledge, no methods for extracting the footprint of ransomware attacks from the ambient and/or bare metal server-generated host logging data is known. Such a ransomware early detection and classification tool is needed in practice to benefit three primary use cases—(1) to expedite currently timely (hundreds of man-hours) manual analysis of host logs used to identify malware events, (2) to detect ransomware activities timely and correctly from logging data of bare metal servers, and (3) to determine families of newly discovered ransomware samples.

The contributions of the present paper are three-fold. First, from a *conceptual* point of view, we motivate the new problem of ransomware early detection and classification, which could assist information technology (IT) operators to identify few infected hosts immediately before the self-propagating ransomware attack infecting the entire network.

Second, to realize ransomware early detection and classification, we develop a $\underline{\text{Deep}}$-learning based $\underline{\text{Ran}}$somware detector (or DeepRan) uses the Term Frequency-Inverse Document Frequency (TF-IDF) approach as an information relative term-weighting scheme to extract semantic information from time series host logs that are collected from bare metal servers. Attention-based BiLSTM, a specialized Recurrent Neural Network (RNN) is used to model the host normalcy. Host activity that is deviated from the normal region is identified as anomalies. The DeepRan classifier is deployed using the BiLSTM-CRF technique to classify the abnormal activity as one of the candidate ransomware attacks. DeepRan is an easy tool for IT operators to use can fulfill some needs as identified in paper (Bridges et al. 2018).

Third, the existing host logging datasets have many limitations and are not capable of developing an advanced cyber threat detector. In this paper, we design an experimental testbed for host log data collection from bare metal servers. We recruit two users in a large enterprise network to collect 63 days of host logs for modeling normalcy. We also execute 17 ransomware samples on the bare metal servers to evaluate DeepRan efficiency. We compare DeepRan, which is trained by the attention-based BiLSTM-FC model with another five commonly used deep learning models. Experimental results validate attention-based BiLSTM-FC is the most efficient model for ransomware early detection, which produces detection accuracy of 99.87%. DeepRan applying the BiLSTM-CRF model classifies 17 different ransomware attacks with an accuracy of 96.5%.

This paper is organized as follows: Section 2 introduces background information of this study. Section 3 illustrates research questions and methodologies for developing an early detection and classification tool. Section 4 presents a case study, experiment and results. Section 5 introduces related prior work, and we conclude the paper and discuss future work in Section 6

2 Background

2.1 Host Logs

Host logs (or host logging data) are records of computer events, either triggered by a user or by a running process. Compared with the traditional network-based intrusion detection systems, which fail to provide a full picture of end-to-end activity that occurs for an event, the host logs can be used to facilitate malware forensic efforts, early detection of malicious activities, and determine the breadth of a compromised operational enterprise network. In particular, the events in which behaviors deviate from normal regions might indicate an initial attack vector (e.g., external threat or insider threat). IT operators should be alerted immediately to protect their operational enterprise networks from cyber attacks. These events and the interrelated nature between a series of malicious events are unique patterns for IT operators to discover new threats that bypass traditional network intrusion detection mechanisms. Moreover, IT operators can use host log to compare network data indicating potential compromise with host logging data to confirm or refute the previous evidence.

2.1.1 Existing Host Log Dataset and Limitation.

The very few existing and public cybersecurity relevant datasets have many limitations as summarized in Los Alamos National Laboratory's (LANL) recent publication (Turcotte et al. 2017). For instance, many datasets are collected from specific and pseudo real-world events rather than daily operational environments. Most datasets are synthetic and created using models intended to represent specific phenomenons of relevance. Some commonly used datasets are egregiously outdated; their underlying systems, networks, and attacks represented are 30 years old, which can no longer represent cyber phenomenon under modern computing environments.

In particular to LANL open-sourced host logging dataset, 20 events are captured from 13,000+ hosts located in LANL's operational network over 90 days. The 20 events record the host and user's activities such as log on, log off, computer shut down and some selected processes. As discussed by the authors, the dataset aims at providing "a thorough understanding of the context, normalization processes, idiosyncrasies and other aspects of the data (Turcotte et al. 2017)." The LANL's dataset is not sufficient for us to model normal behavior of end-to-end activity in an operational enterprise network. To create a detector that could identify malicious activities of ransomware attacks in the early stage and facilitate malware forensics efforts, it is necessary to collect our data.

Windows Logging Service. In this study, host logging data is collected from each computer (or bare metal server) running Microsoft Windows 7 OS by the software called Windows Logging Service (WLS) (Windows Logging Service 2020). WLS is a Windows service installed on the Windows host, which augments traditional logging and forensic analysis (i.e., host event logs and network traffic) with real-time reporting of contextual OS information. This collected OS information of each host is sent to one (or more) Linux *syslog* server(s) via standard *syslog* messages. In addition to event logs, WLS monitors a variety of system details such as *CertificateMonitor*, *DeviceMonitor*, *RegistryMonitor*, *FileMonitor* that are often cited in incident reports as indicators of compromise (IoC) (Campus 2017).

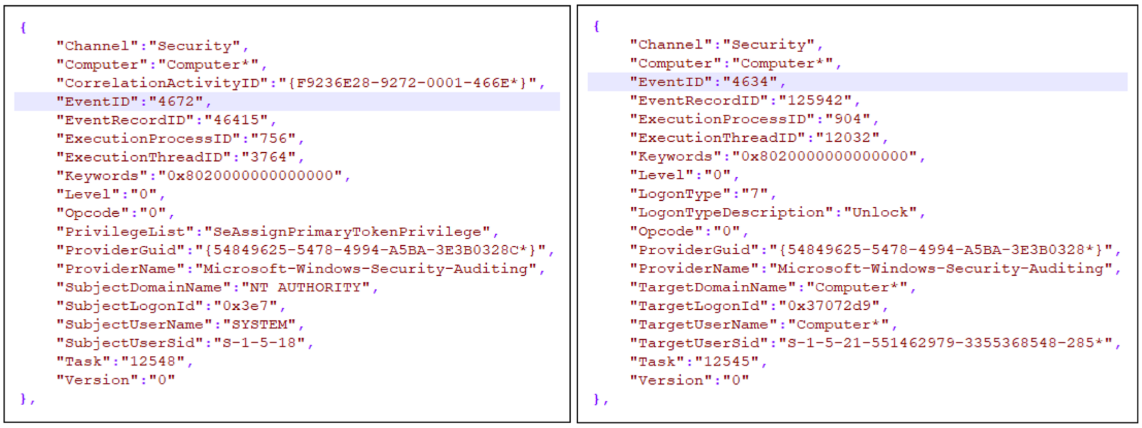
Host Logging Data Pre-Processing. The collected *syslog* messages of each host are first separated by network traffic and host logs. We further on filtering out the host logging data that have no *EventID* field. The processed host logging dataset is stored in JSON files, where objects contain information about events. Event information is described by a number of *string: value* pairs, where "string" is the event information field, and "value" is the value of such field. All events in the processed host logging dataset contain the "EventID" field, which represent the processes triggered by users or computer programs. Hundreds of unique "EventID" values have been collected, and the number of unique "EventID" values vary by user's activity (or the different running processes). For example, a sequence of five events have EventID values 4624 → 4672 → 4798 → 4798 → 4634, meaning that (1) the user successfully logon ($EventID = 4624$); (2) special privileges are assigned to new logon ($EventID = 4672$); (3) a user's local group membership is enumerated by "chrome.exe," the Google Chrome application ($EventID = 4798$); (4) the user's local group membership is enumerated by "svchost.exe, " a Windows system process to host from one to many services ($EventID = 4798$); and (5) the user is logged off ($EventID = 4634$). In this sequence, events (3) and (4) have the same "EventID" values (i.e., $EventID = 4798$), but these two events have different values for the field "CallerProcessName," such as *"chrome.exe"* and *"svchost.exe"* for events (3) and (4), respectively.

Therefore, to develop an efficient anomaly detectors from host logs requires us to analyze each field of events. The number of fields and the type of fields are different for each event. For example, the unique number of fields collected from our host logs are more than 3,000. The number and type of fields for the same event (i.e., the events that have the same "EventID" value) are the same. Figure 1 presents two events of JSON structured host logs as examples. Figure 1a presents the first event information, in which "EventID" value is 4672. Besides the "EventID" field, this event has the other 18 fields (i.e., Chanel, Computer, ..., Task, Version). Figure 1b illustrates information about the second event, in which the value of "EventID" is 4634. Although the second event also has 19 fields, some field types are different from the first event's.

2.2 Threat Model: Ransomware Attacks

In contrast to the 2017 ransomware **WannaCry** that infected 300K machines globally, the majority of ransomware attacks are targeting small businesses (Davis 2019). These crypto-ransomware attacks usually use Windows API function calls to read, encrypt and delete files. Ransom messages are displayed on the screen after the ransomware infecting the host. This study selects and analyzes the 17 disruptive ransomware attacks/families that were first observed between 2015 and 2019.

1. **Vipasana***(2015, 2016)* encrypts files with the public key in a hybrid approach of symmetric and asymmetric encryption. Vipasana deployment does not require Internet connectivity, but sends the private key for decryption to the attack server. The private key is



```
{
    "Channel":"Security",
    "Computer":"Computer*",
    "CorrelationActivityID":"{F9236E28-9272-0001-466E*}",
    "EventID":"4672",
    "EventRecordID":"46415",
    "ExecutionProcessID":"756",
    "ExecutionThreadID":"3764",
    "Keywords":"0x8020000000000000",
    "Level":"0",
    "Opcode":"0",
    "PrivilegeList":"SeAssignPrimaryTokenPrivilege",
    "ProviderGuid":"{54849625-5478-4994-A5BA-3E3B0328C*}",
    "ProviderName":"Microsoft-Windows-Security-Auditing",
    "SubjectDomainName":"NT AUTHORITY",
    "SubjectLogonId":"0x3e7",
    "SubjectUserName":"SYSTEM",
    "SubjectUserSid":"S-1-5-18",
    "Task":"12548",
    "Version":"0"
},
```

(a) Example: EventID 4672

```
{
    "Channel":"Security",
    "Computer":"Computer*",
    "EventID":"4634",
    "EventRecordID":"125942",
    "ExecutionProcessID":"904",
    "ExecutionThreadID":"12032",
    "Keywords":"0x8020000000000000",
    "Level":"0",
    "LogonType":"7",
    "LogonTypeDescription":"Unlock",
    "Opcode":"0",
    "ProviderGuid":"{54849625-5478-4994-A5BA-3E3B0328*}",
    "ProviderName":"Microsoft-Windows-Security-Auditing",
    "TargetDomainName":"Computer*",
    "TargetLogonId":"0x37072d9",
    "TargetUserName":"Computer*",
    "TargetUserSid":"S-1-5-21-551462979-3355368548-285*",
    "Task":"12545",
    "Version":"0"
},
```

(b) Example: EventID 4634

Fig. 1 Sample of Host Logging Data in JSON Format

given back to victims after the ransom is paid (Olbrich 2016).

2. **Xorist** *(2016)* is a Ransomware-as-a-Service (RaaS) attack which propagates via a `JS` file (i.e., a text file containing JavaScript code) of an email attachment. Once the victim executes the malicious `JS` file containing the obfuscated code, Xorist's payload is automatically downloaded and saved to `%temp%` folder of the victim PC. Xorist initiates the encryption process by filtering and matching their extension criteria to modify registry. After encrypting data, Xorist drops ransom notes in all scanned drives and folders (Meskauskas 2019; Detailed technical analysis of xorist ransomware (ransomware report) 2018).
3. **TeslaCrypt** *(2015, 2017)* is a member of Xorist family that encrypts game-play data of specific computer games (Teslacrypt ransomware attacks 2020). This attack is distributed by the popular Angler browser exploit kit, encrypts files using AES-256 encryption algorithm and extorts victims for a ransom of $250 to $1000. The malware uses the `Tor` anonymity network for command and control (C2) without requiring network connectivity for data encryption (INTELLIGENCE 2015).
4. **Fantom** *(2016)* displays a fake Windows Update screen to fool victims while secretly encrypting the victim hosts' file system. Fantom generates a random AES-128 key, and encrypts the key with the RSA encryption algorithm. The key is uploaded to the C2 server. The extension of the encrypted file names are ".locked4", ".fantom" or ".locked". After completely encrypting the files, Fantom changes the wallpaper and keeps a "DECRYPTYOURFiles.html" ransom note on the desktop and in folders of encrypted files(Meskauskas 2017).
5. **JigSaw** *(2016)* has spread around the globe via malicious spam emails. It encrypts files with the AES cipher. JigSaw has more than 45 versions, and some variants delete files every 60 minutes (Morparia 2016; Kiguolis 2019).
6. **WannaCry** *(2017)* is a ransomware attack with historic world-wide effect that launched on May 12, 2017 (Perlroth and Boeing possibly hit by 'wannacry' malware attack 2018). The WannaCry dropper is a self-contained program consists of three components, an application for data encryption and decryption; an encryption key file; and a copy of Tor. WannaCry exploits vulnerabilities in Windows Server Message Block (SMB) and propagates malicious code to infect other vulnerable machines in connected networks.
7. **Petya** *(2016, 2017)* self-propagates by exploiting Windows EternalBlue and SMB vulnerabilities. Petya is initially executed via *rundll32.exe*, removes itself from the infected system, and then creates the C:/Windows/perfc as a flag to indicate the host has been infected. Petya encrypts specific file types in user-mode, and the encryption key is further on being encrypted with an embedded public key and being appended to the *README.TXT* file (Team 2017).
8. **GoldenEye** *(2016, 2017)* is a combination of MISCHA and Petya (Meskauskas 2018), and disguises itself as a Trojan with a *.xls* extension. GoldenEey is widely distributed via phishing emails, which installs its copy in the `%APPDATA%` directory under the name of a random application found in the system. The high-level attack (MISCHA) is deployed to encrypt the files and drop *TXT* format ransom notes. The attack bypasses UAC and elevates its own privileges to make the second attack at a low-level by installing Petya at the beginning of the disk. After Petya is

deployed, the system crashes and starts with a fake CHKDSK (Sponchioni 2016; Labs 2017).

9. **BTCware** *(2017)* is one of the ten most common ransomware families in 2017. It brute-force weak passwords of the Windows Remote Desktop Protocol (RDP) . The attackers break into the compromised hosts and execute the ransomware executable file manually. Some variants of BTCWare also distributes by spam emails or campaigns through attached files containing JS or malicious macro (Anand Ajjan 2018).
10. **Defray** *(2017)* propagates via phishing emails with an attached *Word* document embedded on an *OLE* package object. The attack has targeted healthcare, education, manufacturing and technology industries 6 (Threat Spotlight 2017). Once the victim executes the OLE file, the Defray payload is dropped in the %TMP% folder and disguises itself as a legitimate executable (e.g., taskmgr.exe or explorer.exe). Defray encrypts the file system but does not change file names or extensions, and also deletes volume shadow copies of the encrypted files (Crowe 2017).
11. **nRansom** *(2017)* blocks access to the infected computers rather than encrypting victim's data (This Ransomware Demands Nude instead of Bitcoin - Motherboard 2017). It demands ten nude photos of the victim instead of digital currency to regain access. As recovery from nRansom is relatively easy, it is not a sophisticated malware but a "test" or a "joke".
12. **RedEye** *(2018)*, is a member of the same ransomware family as Annabelle and JigSaw (Arghire 2018). RedEye is distributed through spam email messages, encrypts files using AES-256, and wipes out the MBR (Master Boot Record). The encrypted files are appended with the RedEye extension, and are overwritten with zeros entirely.
13. **Saturn** *(2018)* is an RaaS attack that terminates itself if the victim host is detected running in a virtual machine environment. The ransomware deletes volume shadow copies, disables Windows startup repair, and clears the Windows backup catalog of the victim hosts. .saturn is appended to the names of encrypted files (Abrams 2018).
14. **Scarab** *(2017-2019)* uses AES-256 encryption, and appends various extensions to the names of the encrypted files. Necurs botnet is used by Scarab to spread malicious software in November 2017. More than 20 variants of the ransomware continue to appear until early 2019 (McAfee 2019; Hioureas 2018).
15. **GandCrab** *(2018, 2019)* is also an RaaS attack that has rapidly spread across the globe since January, 2018. In 2019 a newer version of GrandCrab uses *Salsa20* stream cipher to encrypt files offline instead of applying RSA-2048 encryption or connecting to the C2 server (Salvio 2018). GandCrab scans logical drives from A: to Z:, and encrypts files by appending a random Salsa20 key and a random initialization vector (IV) (8 bytes) to the contents of the file. The private key is encrypted in the registry using another Salsa20 key. IV is encrypted with an RSA public key embedded in the malware. This new encryption method makes GandCrab very hard to decrypt, and only GandCrab creators can decrypt the files (Mundo and Gandcrab ransomware puts the pinch on victims 2018).
16. **Ryuk** *(2018, 2019)* is a highly targeted attack that has detrimental effects to data centers and enterprises globally. The victim PCs are first infected by Emotet or Trickbot Trojan. Afterward, the threat actors map and assess the victim's network; deliver Ryuk ransomware to encrypt files and network drives using AES and RSA encryption techniques; kill hundreds of processes and services; drops ransom notes; and deletes shadow copies, backups and the encryption key(Itay Cohen 2018; Infoblox 2019). Three Alabama hospital IT networks were attacked by Ryuk in October 2019, forcing the victim hospitals to turn away non-critical patients and obliging ambulances.
17. **Sodinokibi** *(2019)* is a huge risk to businesses and organizations after its initial attacks in Asia and Europe. The ransomware is distributed via phishing emails containing a malicious link. It downloads a zip file of the Sodinokibi payload containing an obfuscated JS file, which can bypass the detection of the majority of antivirus vendors. The malware module is loaded into memory functions, and the payload is injected into an AhnLab antivirus process. The ransomware iterates through all folders on the victim machine encrypt files with the RC4 encryption technique and leaves a ransom note in each folder. After encryption, the ransomware changes the desktop wallpaper, and all shadow files are deleted (Fakterman 2019; Nocturnus 2019).

2.3 Language Modeling Using Recurrent Neural Networks

Host logging data containing time series host activity/events is similar to natural languages. This motivates us to create a detector to identify malicious activity timely by applying natural language models.

Recurrent Neural Network (RNN) has been validated as an efficient method to model natural language (Brown et al. 2018). RNN is a class of artificial neural sequence model that learns from historical input information and uses its internal state to process sequences of inputs to predict the next input from the previous observation. RNN consists of the input layer, hidden units, and the optional output layer. Let x_t denote the current step of the input sequence, h_{t-1} denote the previous hidden state. The next hidden state h_t can be calculated as follows (Chen et al. 2017).

$$h_t = f(Ax_t + Wh_{t-1}) \tag{1}$$

where f is a non-linear activation function (e.g., softmax function). A and W represent weight matrices of the current input vector x_t and the previous hidden state h_{t-1}.

Although RNN performs well in time series analysis, the vanishing gradient problem makes RNN models hard to train. It is a time consuming procedure to get the final results of model training or updated weights. Long Short Term Memory (LSTM) is a type of RNN network that uses gate functions to solve the vanishing gradient problem (Hochreiter and Schmidhuber 1997). LSTM has similar control flow as RNN, but LSTM's cell operation is different from RNN networks'. The LSTM network has many LSTM cells. Cell states (in a block) are regulated (i.e., protected and controlled) by input, forget and output gates (Hochreiter and Schmidhuber 1997). The LSTM network captures temporal information from a sequence of inputs and creates a directed graph in the time domain.

Bidirectional recurrent neural networks (BiLSTM) is a bidirectional variant of LSTM that connects two hidden layers of opposite directions to the same output. Compared with LSTM, the hidden layer of BiLSTM is split in two directions, forwards (from past to future) and backwards (from future to past) (Huang et al. 2015). The BiLSTM network is selected for this study to train the host logging data as the BiLSTM network model can facilitate information extraction from host logging data sequential vectors in both directions. Fig. 2 illustrates the functional diagram of BiLSTM. One LSTM network processes the sequence from the top to the bottom (forwards) and the other processes the sequence from the bottom to the top (backwards). At each time step t, the forward pass calculates the hidden state h_t by considering the previous hidden state h_{t-1} and the new input sequence x_t. At the same time, the backward flow calculates the hidden state h_t considering the future hidden state h_{t+1} and the current input x_t. Afterward, the forward h_t and the backward h_t are concatenated to obtain the combined vector representation.

As we introduced above, event information captured in host logging data has multiple fields, and some fields might have a higher impact on identifying an anomalous event than others. Therefore, we employ **attention mechanism** to give different focus to the information outputted from the hidden layers of BiLSTM. BiLSTM includes the attention mechanism that can prioritize the significance of the important fields while penalizing the "noise" fields. The attention mechanism assigns weights to the event

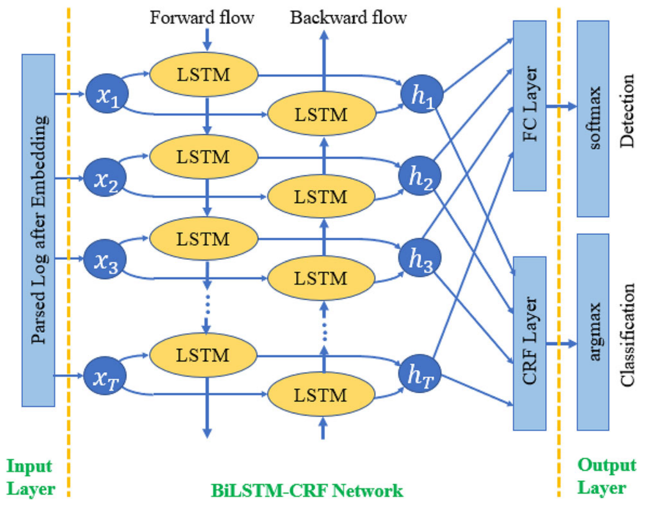


Fig. 2 Functional Diagram of BiLSTM

fields according to their significance using a `fully connected layer` (or FC layer). Each hidden state h_t at the time step $t \in [1, T]$ of the BiLSTM network is passed through the FC layer to obtain an individual trainable weight α_t at time step t as:

$$\alpha_t = tanh(w_t^{\alpha} \cdot h_t) \tag{2}$$

where w_t^{α} is the weight of the attention layer at time step t. A large α_t value means the input event field at time step t is more important. Therefore, we can obtain a trainable weight vector α=$[\alpha_1, \cdots, \alpha_T]$.

Finally, the model predicts the input event sequence as normal behavior or attacks (anomaly detection) depending on the output attention score. The attention score can be obtained by applying a `softmax` to the alignment score vector $W_{align} \cdot \alpha$, where W_{align} is a vector of alignment weights (Zhang et al. 2019). The attention score is calculate as follows.

$$\texttt{AttentionScore} = \texttt{softmax}(W_{align} \cdot \alpha) \tag{3}$$

The model detects ransomware attacks if the predicted `AttentionScore` value is lower than the pre-defined threshold (i.e., 0.5) that is obtained by training normal behavior.

When the FC layer output indicates the input event sequence are abnormal, we can apply an attention-based BILSTM-CRF model by adding a `Conditional Random Field (CRF)` layer on the top of the attention-based BiLSTM model for multi-class classification. Attention-based BILSTM-CRF is a type of discriminative undirected probabilistic graphical model (Chen et al. 2017; Ma and Hovy 2016), which passes the context based information output from the original attention-based BiLSTM model to its sequential CRF layer. The structured output of attention-based BiLSTM-CRF is represented as a single log-linear distribution as a function of each observation sequence (Ma and Hovy 2016). Using the maximum conditional likelihood estimation (i.e., the logarithm of the likelihood), the attention-based BILSTM-CRF predicts the input event sequence to the output classes that have the highest conditional probability. The output of our BILSTM-CRF model predicts the input events as one of the known ransomware attacks.

3 Research Questions and Methodologies

This study aims at developing a high-accuracy malware detector to identify abnormal host behavior from a large volume of ambient (un-attacked) host logs as early as possible. Once the detector detects the host is compromised, it will further on classifying the malicious behavior to one of the known malware families. The following constraints and research questions must be addressed to develop an efficient detector for malware early detection and classification.

1. Host logging data is a sequential ordering of host events in plain text format. These host events are initiated by users or computer processes, and some events are a consequence of successive response of related processes. The detector should be able to process massive amounts of raw text and perform well in time series analysis and prediction.
2. Host logging data is very sparse as most fields only occur with a given value of the `EventID` field. The number of unique fields can be hundreds or even thousands, depending on the types and numbers of different host activities initiated by (il)legitimate users. Also, some fields can have thousands of different values. The detector, therefore, must be capable to discover the interrelated nature of the events and their specific fields and parameter values.
3. Host logging data of an operational enterprise network normally contains much fewer infected hosts than normal hosts. The detector must identify the abnormal events or the sequence of abnormal events induced by malware from a few hosts that are infected with malware.
4. The proposed early detection function requires the detector to immediately and accurately determine the host security status. The detector, therefore, should achieve a high detection accuracy in time.
5. The detector should also be highly efficient for multi-class classification to classify abnormal host events into known/existing malware families.

We explore answers to these questions by developing a specific ransomware detector using deep learning techniques. Deep-learning based Ransomware detector (or DeepRan) uses Term-Frequency-Inverse-Document-Frequency (TF-IDF) as an information relative term-weighting scheme to extract semantic information from each event in the host logging data and trains semantic information using an attention-based BiLSTM, a specialized Recurrent Neural Network (RNN) to model normalcy. The events that are deviated from the normal region are anomalies. DeepRan will also predict abnormal events as one of the candidate ransomware attacks. Fig. 3 presents the framework of DeepRan. Details of DeepRan components and methodologies are described as follows.

3.1 Log Parser

Log Parser converts the host logs of a JSON structured text to the semantic information vector format. The output event vectors are training data or test data, which are inputs to deep learning models for ransomware early detection and classification.

Host Logging Data Processing: Tokenization and Vectorization. In a similar fashion to Natural Language Processing (NPL) that converts a group of sentences into tokens, the events in the host logs can be converted to tokens $T = [t_1, t_2, \cdots, t_N]$. As presented in Fig. 1, in a host logging file, the "string" (or fields) of the dictionary structure (e.g., Channel, Computer) are the tokens, and the number of unique tokens selected from the event fields are N. Parameter values of these selected tokens are converted to word vectors. The multi-dimensional events in a host logging file, therefore, can be converted to a sequence of event vectors $V = [V_1, V_2, \cdots, V_K]$, where K represents the $K-th$ events (entries) at the time step K. Every event vector V_i

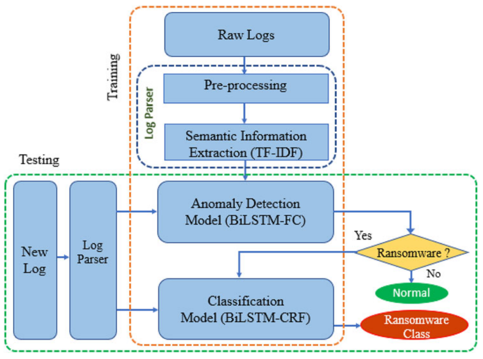

Raw Logs
Training
Log Parser
Pre-processing
Semantic Information Extraction (TF-IDF)
Testing
New Log
Log Parser
Anomaly Detection Model (BiLSTM-FC)
Yes
Ransomware ?
No
Normal
Classification Model (BiLSTM-CRF)
Ransomware Class

Fig. 3 DeepRan Framework

($i \in [1, K]$) has a fixed number of N tokens. An event vector can be represented as $V_i = [v_1, v_2, \cdots, v_N]$. It is possible that one or more tokens' values are not available for some events. In this scenario, the default value "N/A" is assigned to the token. It is also possible that some events have more than N keys/fields. In this scenario, these event fields and values are ignored as the fields are not in the set of the selected tokens.

Word Embedding and Featurization. We use the FastText model, an extension to Word2Vec, for word embedding due to the value of tokens in the host logs are not formal English words. For example, the token/field "SubjectLogonID" of the event which $EventID = 4672$ (as presented in Fig. 1) has a hex value "0x3e7" in a string format. Models such as a one-hot vector or Word2Vec may not have a good performance of rare word representations. FastText, an embedding model using character-level information can appropriately represent rare words such as token parameter values of host events. Therefore, each token parameter value v_j ($j \in [1, N]$) of host events can be converted to a d dimension vector u_j (i.e., $v_j \rightarrow u_j = [a_1, a_2, \cdots, a_d]$). If we transpose of V_i as $V_i^{(T)}$, where $i \in [1, K]$ then $V_i^{(T)}$ can be converted as U_i, which is a $N \times d$ matrix. Thus, V is converted to $U = [U_1, U_2, \cdots, U_K]$.

Weights of Words. Not all words equally represent the meaning of a particular sentence (or event information). TF-IDF is a weighting method that identifies the relative importance of a word in a particular document out of a collection of documents (Salton and Buckley 1988). "Given two sets of documents, and let $f(t, d)$ denote the frequency of term t in document d, and M is the size of the corpus. The TF-IDF weight of a word is the product of the Term Frequency, $\mathrm{tf}(t, d) = f_{t,d} / \sum_{t' \in d} f_{t',d}$ (giving the likelihood of t in d) and the Inverse Document Frequency, $\mathrm{idf}(t, D) = \log[M/(1 + |\{d \in D : t \in d\}|)]$ (giving the Shannon's information of the document containing t). Intuitively, given a document, those terms that are uncommonly high frequency in that document are the only terms receive high scores (Chen and Bridges 2017; Chen et al. 2019)."

By applying the TF-IDF approach, the numerical weight vector of an event word vector $V_i = [v_1, v_2, \cdots, v_N]$ ($i \in [1, K]$) is converted as $W_i = [w_1, w_2, \cdots, w_N]$, where w_i is the TF-IDF score of v_i. The semantic information vector $X = [X_1, X_2, \cdots, X_K]$ is the output of Log Parse, and $X_i = \frac{1}{N} \sum_{i \in [1,N]} W_i * U_i$. Fig. 4 illustrates the workflow of Log Parsing that converts host logs (plain texts) to numerical semantic information vectors.

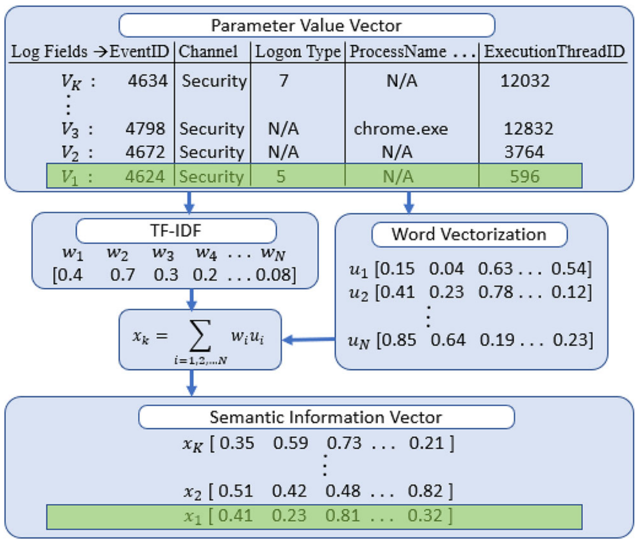

Parameter Value Vector
Log Fields →EventID | Channel | Logon Type | ProcessName ... | ExecutionThreadID
V_K : 4634 Security 7 N/A 12032
V_3 : 4798 Security N/A chrome.exe 12832
V_2 : 4672 Security N/A N/A 3764
V_1 : 4624 Security 5 N/A 596
TF-IDF
w_1 w_2 w_3 w_4 ... w_N
[0.4 0.7 0.3 0.2 ... 0.08]
Word Vectorization
u_1 [0.15 0.04 0.63 ... 0.54]
u_2 [0.41 0.23 0.78 ... 0.12]
u_N [0.85 0.64 0.19 ... 0.23]
$x_k = \sum_{i=1,2,...,N} w_i u_i$
Semantic Information Vector
x_K [0.35 0.59 0.73 ... 0.21]
x_2 [0.51 0.42 0.48 ... 0.82]
x_1 [0.41 0.23 0.81 ... 0.32]

Fig. 4 Log Parsing Workflow

3.2 Anomaly Detection Model for Ransomware Early Detection

We use attention-based BiLSTM with a fully connected (FC) layer shown in Fig. 2 to create training models because sequential information in the host logging data can give away ransomware abnormal patterns. This study aims at detecting ransomware infected hosts as soon as possible, but training input semantic information of event vectors only cannot detect ransomware abnormal behavior immediately. Therefore, we train the attention-based BiLSTM-FC models with two sliding window sizes. The input of the first model (BiLSTM-FC model 1) is a sequence of event's semantic information vectors, and its sliding window size is N (the size of the vector). The input of the second model (BiLSTM-FC model 2) is a sequence of previous N components (or event fields), and its sliding window size is 1. Both BiLSTM-FC models predict the next component's (or field's) AttentionScore based on the input sequences.

As presented in Fig. 5, the true event sequences V_i are host logging data captured by WLS at time step $i \in [1, K]$. $X_i = [x_{i,1}, \cdots, x_{i,N}]$ is semantic information of the true event V_i that has N selected components (or event's fields). Fig. 5 gives three events' semantic information vectors X_1, X_2, X_3 as an example. From the time step 2 onward the host is compromised by a ransomware attack. As we discussed in Section 2.1, the same event has the same number and type of fields, but the values of the same fields may be different when the host performs normally and when the host is attacked by ransomware. The red components of the vectors X_2 and X_3 in Fig. 5 such as $[x_{2,2}, \cdots, x_{2,N}]$ and $[x_{3,1}, \cdots, x_{3,N}]$ represent the fields' semantic information is abnormal caused by the ransomware attack. BiLSTM-FC model 1 is in the orange block below the *True Event Sequences*, which illustrates the model with the sliding window size N predicting normal behaviors of the host at time step 2. The model predicts the host behaves abnormally at time step 3 because of the input of the model is the true and abnormal vector X_2. From the analysis, we observe that the BiLSTM-FC with the N sliding window size has some delays of detection, and at least one-time step delay.

On the other hand, the black block above the *True Event Sequences* in Fig. 5 illustrates the BiLSTM-FC with the sliding window size 1 can predict the host's abnormalcy at time step 2. This is because the majority of the predicted AttentionScores of X_2 components are abnormal when comparing the predicted values with the normalcy threshold 0.5. (The host is normal if the majority of the component's AttentionScores are greater than or equal to the threshold; otherwise, the host is predicted as abnormal.) Thus, we use BiLSTM-FC with the sliding window 1 to predict the host's abnormal behaviors immediately, and use the sliding window N to keep the track of each event for ransomware early detection.

3.3 Ransomware Attack Classification

Polymorphic malware constantly changes its identifiable features to evade antivirus detection. Ransomware attacks usually apply the same technique to continuously evolve and generate multiple variants before the end of its lifespan. We also observed that most ransomware attacks belong to the same ransomware families or inherit encryption/propagation mechanisms from their ancestors. To facilitate malware forensics efforts, the DeepRan detector should also perform as a multi-class classifier that is capable to classify the detected anomalies to appropriate ransomware families. The correct classification of system abnormal events can facilitate forensics efforts, which will help the IT operators to make decisions in malware protection and mitigation.

When the BiLSTM-FC model predicts the input sequence of events are abnormal, DeepRan passes the output of the BiLSTM network to a new CRF layer (instead of the FC layer) to predict which ransomware attacks are compromising the host. The BiLSTM-CRF model determine the class of ransomware attacks (or families) from the candidate classes based on the predicted conditional probability of each class. The abnormal events are classified as the candidate that has the highest probability. Algorithm 1 presents how DeepRan detects anomalies from host logging data, and then determines the compromised host is under which ransomware attacks from the 17 candidate ransomware attacks/families.

Algorithm 1 Pseudo Code of DeepRan.

```
Input: Normal and Ransomware log Tokens (T)
Output: Ransomware detection probability and class of ransomware if detected.
1:  $parameterVector(V) \leftarrow (T)$
2:  for $i \leftarrow 1, 2...K$ do
3:      $U_i \leftarrow wordEmbedding(parameterVector(V_i))$
4:      $W_i \leftarrow TF-IDF(parameterVector(V_i))$
5:  end for
6:  $semanticInfoVector(X) \leftarrow SumofProduct(W, U)$
7:  $trainData, testData \leftarrow TrainTestSplit(X)$
8:  if $Attention-basedBiLSTM-FC$ then
9:      for $seq \in trainData$ do
10:         $modelDetection \leftarrow BiLSTM-FC(seq)$
11:     end for
12: end if
13: if $Attention-basedBiLSTM-CRF$ then
14:     for $seq \in trainData$ do
15:         $modelClassification \leftarrow BiLSTM-CRF(seq)$
16:     end for
17: end if
18: for $seq \in testData$ do
19:     $pobabilityRansomware \leftarrow modelDetection(seq)$
20:     if $Ransomware(pobabilityRansomware)$ then
21:         $ransomwareClass \leftarrow modelClassification(seq)$
22:     end if
23: end for
```

4 Case Study

We design the following experiments to collect training data and test data, and validate the efficiency of DeepRan.

4.1 Testbed Design and Bare Metal Host Log Collection

As described in Section 2.1.1, the existing datasets have many limitations and are not capable of developing an advanced cyber threat detector. In this study, we design an experimental testbed for (1) providing a realistic and isolated environment to deploy malware executable; (2) collecting real-time bare metal host logs when the physical machines operate normally and/or infected by malware

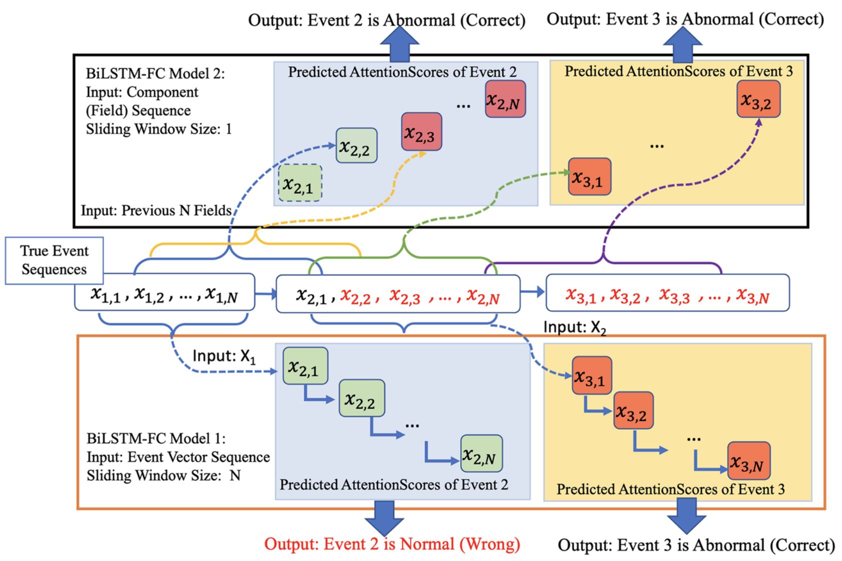

Output: Event 2 is Abnormal (Correct)
Output: Event 3 is Abnormal (Correct)
BiLSTM-FC Model 2:
Input: Component (Field) Sequence
Sliding Window Size: 1
Predicted AttentionScores of Event 2
Predicted AttentionScores of Event 3
Input: Previous N Fields
True Event Sequences
Input: X1
Input: X2
BiLSTM-FC Model 1:
Input: Event Vector Sequence
Sliding Window Size: N
Output: Event 2 is Normal (Wrong)
Output: Event 3 is Abnormal (Correct)

Fig. 5 Attention-based BiLSTM-FC for Ransomware Early Detection

(i.e., ransomware); and (3) processing raw logs to harvest reproducible and shareable host logging data.

As presented in Fig. 6, the testbed has three physical/bare-metal machines: one Linux server (the PC on the left side of the figure) and two Windows 7 clients, Host1 and Host2. The Linux server has two functionalities such as (1) performs as a Fog Server (FOG Project 2020) to re-image associated OS for its clients (Host1 and Host2). We conduct experiments to collect host logging data when the host is under different ransomware attacks. After executing one ransomware sample, the host is re-imaged by the Linux Server to assure the host configuration is always the same and the host security status is normal before the host is infected by another ransomware sample again. (2) The Linux server also performs as a system message logging server, which receives real-time logging data that are sent from its clients (Host1 and Host2) via the Windows Logging Services (WLS) software installed on the clients.

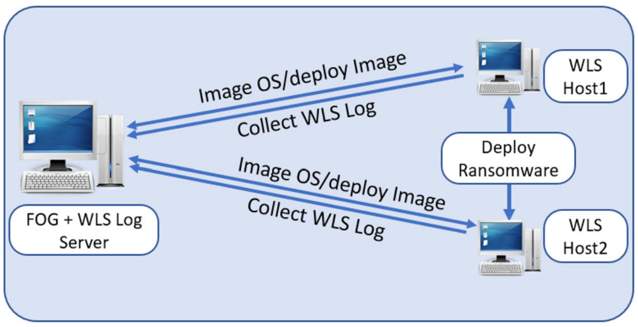
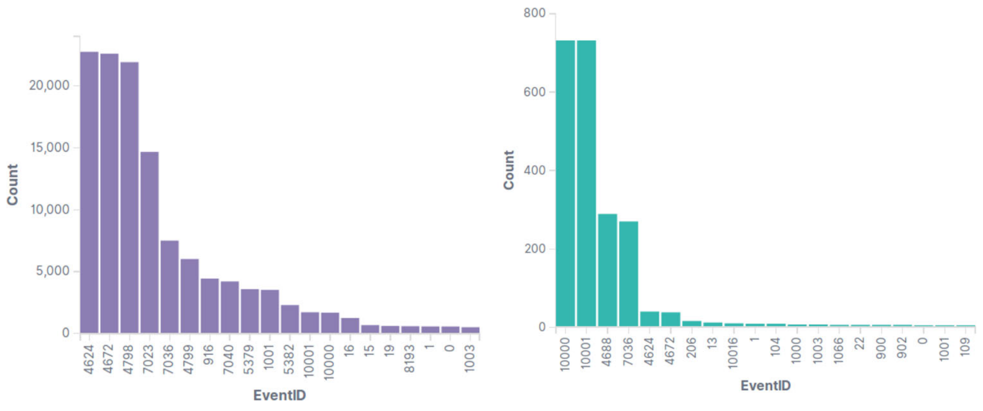


Fig. 6 Testbed: Real-Time Host Log Collection

Host Log Collection—Normal Activity. We recruit two legitimate users from a large enterprise network and collect their working PCs' host logs over 63 days (IRB-approved). User's normal activity includes reading, writing and deleting files, opening websites, watching video streams, uploading/downloading files into cloud storage, sending and receiving emails, (un)installing software. The host logs are collected by WLS installed on the two client PCs, and we obtain 126 JSON files (one file per user per day) of normal host logs that contain more than one million logged events. Among the one million logged events, we observe 250 unique values of `EventID`. Fig. 7 lists the 20 most frequently seen events and their frequency. Note that the number of unique fields of the one million logged events is more than 3,000.

Host Log Collection—Ransomware Activity. The 17 ransomware executables are fully deployed on two Windows 7 physical machines for us to collect infected host logs. The infected hosts are re-imaged by the Fog Server to assure the host performs normally before deploying the next ransomware samples. Therefore, the initial state of the victim host is always normal, and we continuously collect host logs for 10 minutes after ransom notes display on the screen. We obtain 17 JSON files of infected host logs, and each file contains the host behavior of one ransomware sample. More than 4,800 events have logged, and 79 events are unique. Note that, not all of the events are abnormal. For example, the events representing host reboot/start before the ransomware samples are executed are normal. These numbers of normal events are much fewer than ransomware events in each file. To simplify the labeling process, we label all events in the 17 ransomware host logs as abnormal. The

Fig. 7 Most frequently Occurred Events



(a) Top 20 events of normal activity

(b) Top 20 events of ransomware activity

top 20 most frequent events captured in ransomware logs are presented in Fig. 7.

4.2 Training and Test Datasets

DeepRan can early detect ransomware attacks compromised the victim host, and then classify the abnormal events into one of the known ransomware families. To detect anomalies, we use the unsupervised attention-based BiLSTM-FC algorithm to model the normal region of host behavior, which is trained by 10% of total normal events (i.e., 103k+ events). The test dataset consists of the rest 90% of normal events and 100% ransomware logged events. If the test events deviate from the normal region, DeepRan continuously classifies the events into one of the 17 ransomware attacks. The classifier adopting the supervised BiLSTM-CRF model is trained and tested by 50% of logged events of every ransomware attack. Table 1 lists the event numbers of the training and test data for validating DeepRan detection and classification efficiency. Note that we evaluate detection and classification models offline separately. Therefore, the number of ransomware events for testing the detector and classifier is different in size.

4.3 DeepRan Implementation

1. **Token Numbers:** The top 50 most frequently seen fields out of total 3,000 unique fields of the training data are selected as tokens ($N = 50$) to form the parameter value vectors.
2. **Word Embedding Dimension:** The value $d = 100$ is passed to the FastText model to featurize and embed word parameter value vectors to semantic information vectors, which are the inputs of the attention-based BiLSTM model. The training data of the detector contains 103,330 normal events. Each event is converted to a 50×100 matrix. The size of training data, after being converted to semantic information vectors is $(103,330 * 50) \times 100$
3. **Attention-based BiLSTM Model Deployment:** We use the open source machine learning library **pytorch** (Pytorch: From research to production 2020) to train the BiLSTM model by selecting the minibatch stochastic gradient descent (SGD) algorithm with Adam optimizer on an NVidia Tesla K80 GPU server. We set the minibatch size to 128, and the initial learning rate as 0.001. During training the model we used cross-entropy as the loss function(LeCun et al. 2015).

Table 1 Normal and ransomware host log dataset

Host Logs	Detection		Classification	
	Training	Testing	Training	Testing
No. of Normal Events	103,330	929,967	0	0
No. of Ransomware Events	0	4,820	2,410	2,410

4.4 Experimental Results

Anomaly Detection. We compare DeepRan with another five commonly used deep learning models (i.e., CNN, LSTM, GRU-FC, BiLSTM-FC, and attention based BiGRU-FC) to validate its accuracy in ransomware attack detection. The six models are trained with normal events only, and the test dataset contains a large volume of normal events and very few ransomware events. The evaluation results are presented in Table 2, and we observe that DeepRan yielding 99.87% detection accuracy, 100% precision, 98.06% recall, 99/02% F1-score and 0.98 Matthews correlation coefficient (MCC) is better than the other five models for ransomware attack detection. This experimental result validates DeepRan is a high-accuracy detector that can be leveraged to detect few ransomware infected hosts from a

Table 2 Comparing DeepRan Performance with Five Commonly Adopted Deep Learning Models

Model	Accuracy	Precision	Recall	F1-Score	MCC
CNN	99%	99%	95.85%	97.40%	0.91
LSTM	98.92%	98.87%	90.85%	94.69%	0.915
GRU-FC	98.99%	98.99%	90.85%	94.75%	0.92
BiLSTM-FC	98.42%	98.72%	91.22%	94.82%	0.91
Attention-based BiGRU-FC	99.65%	100%	95.42%	97.65%	0.93
DeepRan: Attention-based BiLSTM-FC	99.87%	100%	98.06%	99.02%	0.98

large number of normal hosts in an operational enterprise system.

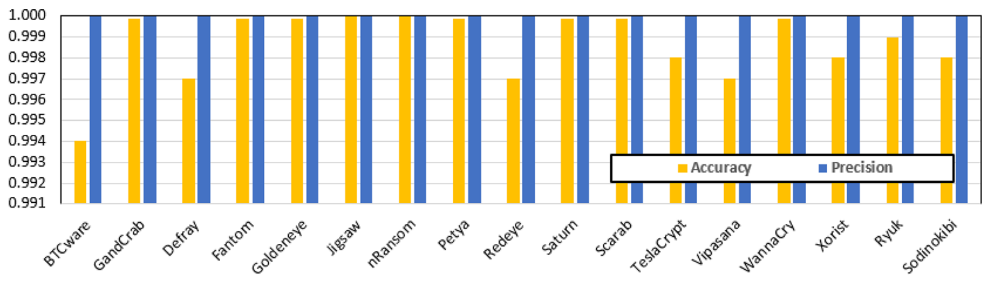
Early Detection. One of the contributions of DeepRan is to detect ransomware attacks before they are fully deployed on the victim host or before ransomware encrypts the victim's data. This is primarily because DeepRan selects sliding window lengths of 1 and 50 to predict the token value for the next event and the next event vector (a sequence of events). These numbers of sliding window lengths are selected as they perform the best among our experiments. If DeepRan detecting anomalies only relies on predicting the next event vector (i.e., sliding window length of 50), it may fail to identify abnormal events if the majority token values of the test event vector are predicted as normal. DeepRan, yielding higher than 99.94% accuracy for detecting 17 ransomware events in the test dataset, proves that DeepRan can help the IT operators/victim users to determine current security status and identify abnormal host behavior since the beginning of the ransomware deployment. Victim users of the infected host and the IT operators of the compromised enterprise network can respond to the ransomware attacks timely to prevent ransomware maliciously encrypts data of the entire network.

DeepRan's efficiency for detecting the hosts that are infected by the 17 ransomware attacks is presented in Table 3. We also provide a bar chart as shown in Fig. 8 for visualizing DeepRan anomaly detection efficiency. The majority of the Matthews Correlation Coefficient (MCC) scores of detecting ransomware attacks are higher than

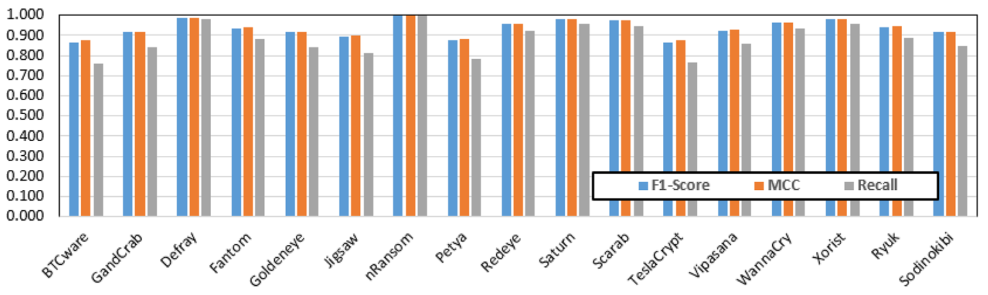
Table 3 DeepRan Anomaly Detection Results for 17 Ransomware Attacks

Ransomware	Accuracy	Precision	Recall	F1-Score	MCC
BTCware	0.994	1	0.762	0.865	0.873
GandCrab	0.999	1	0.842	0.914	0.918
Defray	0.997	1	0.979	0.989	0.989
Fantom	0.999	1	0.88	0.936	0.938
Goldeneye	0.999	1	0.841	0.914	0.917
Jigsaw	0.999	1	0.812	0.896	0.901
nRansom	1	1	1	1	1
Petya	0.999	1	0.782	0.877	0.884
Redeye	0.997	1	0.923	0.96	0.96
Saturn	0.999	1	0.96	0.979	0.979
Scarab	0.999	1	0.945	0.972	0.972
TeslaCrypt	0.998	1	0.763	0.865	0.873
Vipasana	0.997	1	0.858	0.923	0.926
WannaCry	0.999	1	0.932	0.965	0.965
Xorist	0.998	1	0.96	0.979	0.979
Ryuk	0.999	1	0.89	0.942	0.944
Sodinokibi	0.998	1	0.845	0.916	0.919

Fig. 8 Visualization of DeepRan Anomaly Detection



(a) DeepRan Accuracy and Precision for 17 Ransomware Detection



(b) DeepRan F1-Score, MCC and Recall for 17 Ransomware Detection

0.9, which validates the early detection of DeepRan has a relatively high quality. Additionally, DeepRan produces no false positive (i.e., precision=1), meaning the events predicted as ransomware activity by DeepRan are all labeled as ransomware from the collection of ground truth data (ransomware host logs). DeepRan recall rates vary between 0.72 to 1, which means that not all events in the ransomware host logs are detected as "abnormal" events. This is mainly because (a) the labels of some ransomware events inaccurately represent the ground truth. As described in Section 4.1, we label all events captured from the infected hosts as "attacks." These host logs are collected during the period when the victim host was booting and until one of the ransomware samples are fully deployed on the victim host. Therefore, the events collected from the host before the host executes the ransomware samples should be labeled as "normal". (b) The events collected from the victim hosts that describe a ransomware attack's runtime behaviors are not all abnormal. For example, the attack may read/write/copy files like normal operations. Since DeepRan aims at detecting few ransomware infected hosts from a large number of normal hosts in the operational enterprise network, the quality and efficiency of DeepRan are not influenced by the model recall rates, but highly dependent on the accuracy and precision metrics.

Ransomware Attack Classification. DeepRan classifies abnormal events as one of the 17 candidate ransomware attacks to assist IT operators in identifying known ransomware and their new variants. The DeepRan classification engine is trained with 50 percent of total ransomware events. We evaluate DeepRan classification accuracy using a test data of the other 50 percent ransomware events. DeepRan classification evaluation results are presented in Table 4. The average classification accuracy of the 17 ransomware attacks is 96.5%. The highest and lowest classification rates are 99.8% (JigSaw) and 93.2% (Defray), respectively. We also observe that DeepRan yields the highest recall value (0.909) for classifying the *nRansom* attacks. This is because `nRansom` is a malicious blocker, which has different runtime patterns to the other 16 crypto ransomware attacks'. The efficiency of the DeepRan classification results are lower than the anomaly detection's. According to the confusion matrix as presented in Fig. 9, the average false positive rate is 1.8% and the false negative rate is 34.5%.

Misclassification indicates that DeepRan's efficiency is impacted if two (or more) ransomware attacks have similar behaviors. For example, these misclassified ransomware attacks may exploit the same vulnerabilities, use similar technical strategies to infect the host, and/or apply the same encryption mechanisms to encrypt the file systems. The misclassified ransomware attacks illustrate their similarity, which can help IT operators to discover new variants of the same ransomware families to facilitate ransomware forensics efforts. As we illustrate that `Goldeneye` and `WannaCry` exploit the same Windows vulnerability for attack propagation and deployment. From the confusion matrix, we observe that 11 out of 125 `Goldeneye` test data is classified as `WannaCry`. We also observe that DeepRan

Table 4 DeepRan Ransomware Classification Scores

Ransomware	Accuracy	Precision	Recall	F1-Score
BTCware	0.963	0.753	0.44	0.555
GandCrab	0.994	1	0.681	0.81
Defray	0.932	0.917	0.882	0.899
Fantom	0.986	0.666	0.638	0.652
Goldeneye	0.967	0.688	0.672	0.68
Jigsaw	0.998	1	0.7	0.823
nRansom	0.982	0.338	0.909	0.493
Petya	0.961	0.288	0.612	0.392
Redeye	0.969	0.964	0.614	0.75
Saturn	0.99	0.406	0.812	0.541
Scarab	0.942	0.478	0.583	0.525
TeslaCrypt	0.961	0.21	0.54	0.303
Vipasana	0.963	0.622	0.326	0.428
WannaCry	0.943	0.558	0.59	0.574
Xorist	0.946	0.193	0.763	0.308
Ryuk	0.947	0.849	0.656	0.74
Sodinokibi	0.956	0.824	0.695	0.754

misclassifies three ransomware attacks: `Defray`, `Redeye` and `Xorist`. For instance, 55 (or 6.7%) of `Defray` test data is mis-classified as `Xorist`; 9 (or 23.7%) of `Xorist` test data is mis-classified as `Defray`, and 45 (or 25.1%) of `Redeye` test data is mis-classified as `Xorist`). Similar patterns among the three ransomware attacks can be extracted from the mis-classified events, which are normally difficult and too tedious for manually static analysis to investigate.

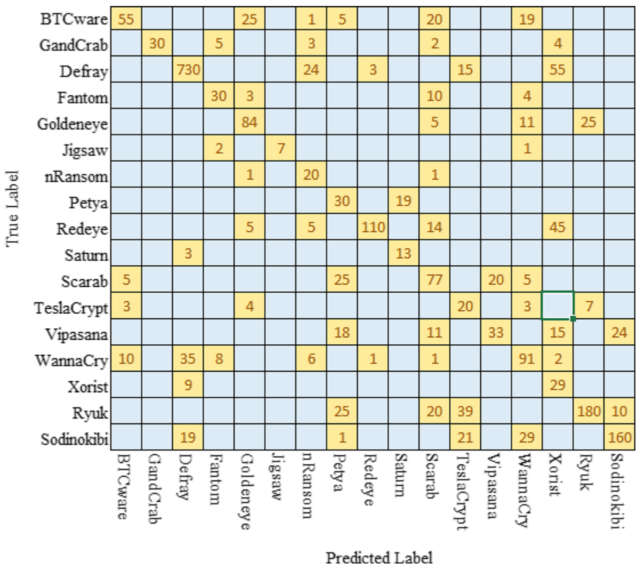


Fig. 9 Confusion Matrix for 17 Ransomware Classification

4.5 DeepRan Quality Degradation Assessment

Usually, models trained by host logging data are unstable due to the system evolution and unknown/new host behavior produced by legitimate users or cyber attackers. To solve the issue of model quality degradation, DeepRan trains host logging data with incremental learning technique by applying gradient descent of the backpropagation algorithm (Schalkoff 1997) to frequently update underlying models with new observations. Therefore, DeepRan can maintain its quality over time without requiring large computation or storage to retrain the model.

To validate the robustness of DeepRan over time, we compare the DeepRan's early detection efficiency when DeepRan is developed with and without applying the incremental/online learning technique. Fig. 10 illustrates that DeepRan's BiLSTM-FC model applying online learning technique has better anomaly detection performance than the training model without applying the online technique. The accuracy of the two models is 99.67% vs. 99.87%; precision is 100% vs. 100%; recall is 98.80% vs. 97.88%; and F1-score is 99.39% vs. 98.93%.

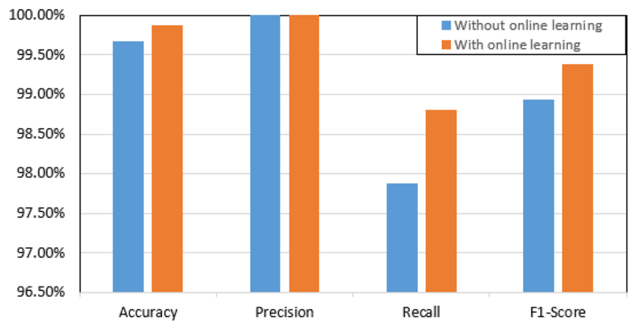


Fig. 10 Assessment of DeepRan Quality Degradation

5 Related Work

5.1 Ransomware Training Data Collection

The state-of-the-art ransomware detectors are trained by either system network traffic or victim host logging data collected in a virtualized/virtual environment (e.g., sandbox or virtual network platform). For example, four ransomware attacks (i.e., WannaCry, Petya, BadRabbit and PowerGhost) propagation network traffic (Fernandez Maimo et al. 2019) is collected from the integrated clinical environment platform, where the victim devices are configured with Windows 7, 10 and Ubuntu 16.04 OS. Humayoun et al. (Homayoun et al. 2017) execute Locky, Cerber and TeslaCrypt samples on Windows 10 virtual machines to collect host logs as training data. Cuckoo Sandbox has also been utilized for harvesting reproducible and shareable host logs (Takeuchi et al. 2018; Chen and Bridges 2017).

5.2 Machine learning for anomaly detection

Machine learning techniques have been widely used for anomaly detection, and model selection is critical for obtaining an efficient detector. Depending on behavior patterns of normal/abnormal system status and indicators of the given datasets, appropriate unsupervised or supervised models have been used to develop high quality anomaly detectors (Bridges et al. 2019). invariant mining (IM), principal component analysis (PCA) and logistic regression are commonly used unsupervised models for anomaly detection. IM determines the linear relationship among system log events by counting the number of events in the datasets(Lou et al. 2010), and malicious log sequences could be identified from the abnormal invariants. PCA divides system log count vectors into normal and abnormal space (Xu et al. 2009), and logistic regression generates a binary regression function based on the count vector of system logs to detect anomaly (Liang et al. 2007). Supervised models such as support vector machine, decision tree and Naive Bayesian are also applied for anomaly detection. However, these stable machine learning models are less efficient for modeling host behavior normalcy in an operational enterprise network because the small variations of input host events can result in a completely different prediction.

Recently, unstable deep learning approaches have been applied to model normalcy from system logs of network traffic to build anomaly detectors. Featurization techniques must be first applied to convert system logs to a numerical vector as the input of deep learning algorithms. RNN models are normally used to develop high accuracy anomaly detectors for time-series input data. For example, Zhang et al. (Zhang et al. 2016) convert system logs to numerical vectors with NPL techniques and filter less significant terms using the TF-IDF method. Afterward, the processed data is trained by an LSTM model to predict malicious logs. An attention-based BiLSTM model that extracts contextual information from the system logs has been validated the efficiency for malware early detection(Zhang et al. 2019).

5.3 Ransomware Attack Detection and Classification

In recent three years, a variety of automated ransomware dynamic analysis and ransomware pattern-based solutions have been developed to facilitate forensics efforts and ransomware detection. For example, the study(Ahmadian and Shahriari 2016) models registry activity to detect ransomware. UNVEIL (Kharaz et al. 2016) and CloudRPS(Lee et al. 2017) detect and identify ransomware attacks by modeling file system activity. Ahmadian et al. (Ahmadian and Shahriari 2016) use Bayesian Network to correctly detect 20 ransomware samples (i.e., F-measure is 0.93). Humayoun et al. (Homayoun et al. 2017) apply the sequential pattern mining method to find maximal frequency patterns (MSP) of ransomware malicious activities instead of generating behavioral features directly from the host. This study compares four machine learning models to classify four ransomware attacks. Experimental results present that the atomic Registry MSPs are the most important sequence of events to detect ransomware attacks with 99% detection accuracy.

Verma et al. (Verma and Bridges 2018) embed host logs into a semantically meaningful metric space. The representation is used to build behavioral signatures of ransomware attacks from host logs for ransomware pre-encryption detection. Morato et al. introduce REDFISH (Morato et al. 2018), a ransomware detection algorithm that identifies ransomware actions when the attack begins to encrypt shared files. REDFISH analyzes SMB traffic and uses three parameters of traffic statistics to detect malicious activity. 19 different ransomware families can be detected by REDFISH in less than 20 seconds. Hardy et al. (Hardy et al. 2016) manually select features from Windows API calls to detect ransomware activity by developing an unsupervised malware detector using deep Stacked AutoEncoders (SAEs). Humayoun et al. (Homayoun et al. 2019) develop a highly accurate ransomware detector called *DRTHIS*. DRTHIS is a deep ransomware hunting and intelligence model that is designed by using both LSTM and CNN networks. Rhode et al. (Rhode et al. 2018) proposed a RNN-based ransomware early prediction algorithm to predict the host's behavior is benign or malicious. The dataset including both network data and OS process logs are collected by executing benign and malicious executables in a virtualization environment. In contrast, our IRB-approved study collects normal datasets from two users' host logging data over 60 days. The host logging data is also collected on bare metal servers by executing 17 ransomware samples.

6 Conclusion and Future Work

This study uses deep learning technique to develop DeepRan for ransomware early detection and classification. An attention based BiLSTM-FC model is validated its efficiency for modeling normalcy of an operational enterprise network using host logging data collected on bare metal servers. DeepRan can also classify the ransomware events to existing ransomware families using an attention based BiLSTM-CFR model. Experimental results validate DeepRan can maintain its efficiency and quality over time. DeepRan can be used for ransomware early detection as it analyses spatial relation of event fields along with temporal relation of time series events. Though DeepRan can detect ransomware before encrypting system resources we didn't included the early detection times in this paper. Future research will extend DeepRan for online malware detection and classification along with early detection time analysis in a real-world large-scale operational enterprise networks.

Acknowledgments This research is sponsored by the National Science Foundation under Grant No.1812599. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

Siegel, B. (2019). Ransomware payments rise as public sector is targeted, new variants enter the market, 11. https://securityboulevard.com/2019/11/ransomware-payments-rise-as-public-sector-is-targeted-new-variants-enter-the-market/.

Challita, A. (2018). The four most popular methods hackers use to spread ransomware, 08. https://www.itproportal.com/features/the-four-most-popular-methods-hackers-use-to-spread-ransomware/.

Dobran, B. (2019). 27 terrifying ransomware statistics & facts you need to read, 01. https://phoenixnap.com/blog/ransomware-statistics-facts.

Davis, J. (2019). 71% of ransomware attacks targeted small businesses in 2018, 03. https://healthitsecurity.com/news/71-of-ransomware-attacks-targeted-small-businesses-in-2018.

Shi, F. (2019). Threat spotlight: Government ransomware attacks, 08. https://blog.barracuda.com/2019/08/28/threat-spotlight-government-ransomware-attacks/.

Cook, S. (2019). 2017-2019 ransomware statistics and facts, 06. https://www.comparitech.com/antivirus/ransomware-statistics/.

Andone, D. (2019). 3 alabama hospitals are accepting patients again after a ransomware attack on its computers, 10. https://www.cnn.com/2019/10/11/us/alabama-hospital-ransomware-attack/index.html.

Cimpanu, C. (2019). Second florida city pays giant ransom to ransomware gang in a week, 06. https://www.zdnet.com/article/second-florida-city-pays-giant-ransom-to-ransomware-gang-in-a-week/.

Bridges, R.A., Iannacone, M.D., Goodall, J.R., Beaver, J.M. (2018). How do information security workers use host data? a summary of interviews with security analysts, arXiv:1812.02867.

Turcotte, M.J., Kent, A.D., Hash, C. (2017). Unified host and network data set, ArXiv e-prints.

Windows Logging Service (2020). https://www.federallabs.org/successes/success-stories/windows-logging-service.

Campus, K.C.N.S. (2017). Windows logging service, 08. https://kcnsc.doe.gov/docs/default-source/kcnsc-software/windows-logging-service-summary_073117.pdf?sfvrsn=26b745c4_2.

Olbrich, C. (2016). A close look at ransomware by the example of vipasana, 10. https://www.boxcryptor.com/en/blog/post/a-close-look-at-ransomware-vipasana-part-i/.

Meskauskas, T. (2019). Xorist ransomware removal instructions, 4. https://www.pcrisk.com/removal-guides/9905-xorist-ransomware.

Detailed technical analysis of xorist ransomware (ransomware report), 0.5. (2018). https://www.howtoremoveit.info/technical-analysis-report-xorist-ransomware/.

Teslacrypt ransomware attacks (2020). https://usa.kaspersky.com/resource-center/threats/teslacrypt.

INTELLIGENCE, D.S.C.T.U.T. (2015). Teslacrypt ransomware, 05. https://www.secureworks.com/research/teslacrypt-ransomware-threat-analysis.

Meskauskas, T. (2017). Fantom ransomware, 06. https://www.pcrisk.com/removal-guides/10418-fantom-ransomware.

Morparia, J. (2016). Ransom.jigsaw, 08. https://www.symantec.com/security-center/writeup/2016-041123-3256-99.

Kiguolis, L. (2019). Remove jigsaw ransomware / virus (removal instructions), 11. https://www.2-spyware.com/remove-jigsaw-ransomware-virus.html.

Perlroth, N., & Boeing possibly hit by 'wannacry' malware attack (2018). [Online] Available: https://www.nytimes.com/2018/03/28/technology/boeing-wannacry-malware.html.

Team, S.R. (2017). Petya ransomware outbreak: Here's what you need to know, 10. https://www.symantec.com/blogs/threat-intelligence/petya-ransomware-wiper.

Meskauskas, T. (2018). Goldeneye ransomware removal instructions, 07. https://www.pcrisk.com/removal-guides/10733-goldeneye-ransomware.

Sponchioni, R. (2016). Ransom.goldeneye, 07. https://www.symantec.com/security-center/writeup/2016-120715-1834-99.

Labs, M. (2017). Goldeneye ransomware – the petya/mischa combo rebranded, 07. https://blog.malwarebytes.com/threat-analysis/2016/12/goldeneye-ransomware-the-petyamischa-combo-rebranded/.

Anand Ajjan, D.P. (2018). Btcware ransomware, 04. https://www.sophos.com/en-us/medialibrary/PDFs/factsheets/sophos-btcware-ransomware-wpna.pdf.

Threat Spotlight (2017). Defray Ransomeware Hits Healthcare and Education. (accessed 16 Aug 2018). [Online]. Available: https://threatvector.cylance.com/en_us/home/threat-spotlight-defray-ransomware-hits-healthcare-and-education.html.

Crowe, J. (2017). Alert: Defray ransomware launching extremely personalized attacks, 08. https://blog.barkly.com/defray-ransomware-highly-targeted-campaigns.

This Ransomware Demands Nude instead of Bitcoin - Motherboard (2017). (accessed 24 Aug 2018). [Online]. Available: https://motherboard.vice.com/en_us/article/yw3w47/this-ransomware-demands-nudes-instead-of-bitcoin.

Arghire, I. (2018). 'redeye' ransomware destroys files, rewrites mbr, 6. https://www.securityweek.com/redeye-ransomware-destroys-files-rewrites-mbr.

Abrams, L. (2018). New saturn ransomware actively infecting victims, 2. https://www.bleepingcomputer.com/news/security/new-saturn-ransomware-actively-infecting-victims/.

McAfee (2019). Threat landscape dashboard scarab - ransomware, 06. https://www.mcafee.com/enterprise/en-us/threat-center/threat-landscape-dashboard/ransomware-details.scarab-ransomware.html.

Hioureas, V. (2018). Scarab ransomware: new variant changes tactics, 1. https://blog.malwarebytes.com/threat-analysis/2018/01/scarab-ransomware-new-variant-changes-tactics/.

Salvio, J. (2018). Gandcrab v4.0 analysis: New shell, same old menace. https://www.fortinet.com/blog/threat-research/gandcrab-v4-0-analysis--new-shell--same-old-menace.html.

Mundo, A., & Gandcrab ransomware puts the pinch on victims, 0.7. (2018). https://securingtomorrow.mcafee.com/mcafee-labs/gandcrab-ransomware-puts-the-pinch-on-victims/.

Itay Cohen, B.H. (2018). Ryuk ransomware: A targeted campaign break-down, 08. https://research.checkpoint.com/ryuk-ransomware-targeted-campaign-break/.

Infoblox (2019). Ryuk ransomware cyber report. https://www.infoblox.com/wp-content/uploads/threat-intelligence-report-ryuk-ransomeware-cyber-report.pdf.

Fakterman, T. (2019). Sodinokibi: The crown prince of ransomware, 08. https://www.cybereason.com/blog/the-sodinokibi-ransomware-attack.

Nocturnus, C. (2019). Sodinokibi: The crown prince of ransomware 08. https://www.cybereason.com/blog/the-sodinokibi-ransomware-attacks.

Brown, A., Tuor, A., Hutchinson, B., Nichols, N. (2018). Recurrent neural network attention mechanisms for interpretable system log anomaly detection. In *Proceedings of the First Workshop on Machine Learning for Computing Systems. ACM, pp. 1*.

Chen, T., Xu, R., He, Y., Wang, X. (2017). Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, *72*, 221–230.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Huang, Z., Xu, W., Yu, K. (2015). Bidirectional lstm-crf models for sequence tagging, arXiv:1508.01991.

Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., Xie, C., Yang, X., Cheng, Q., Li, Z., et al. (2019). Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, pp. 807–817*.

Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf, arXiv:1603.01354.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513–523.

Chen, Q., & Bridges, R.A. (2017). Automated behavioral analysis of malware: A case study of wannacry ransomware. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, pp. 454–460*.

Chen, Q., Islam, S.R., Haswell, H., Bridges, R.A. (2019). Automated ransomware behavior analysis: Pattern extraction and early detection. In *International Conference on Science of Cyber Security, pp. 199–214*. Berlin: Springer.

FOG Project (2020). A free open-source network computer cloning and management solution, https://fogproject.org/.

Pytorch: From research to production (2020). https://pytorch.org/.

LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

Schalkoff, R.J. (1997). Artificial neural networks. McGraw-Hill Higher Education.

Fernandez Maimo, L., Huertas Celdran, A., Perales Gomez, A.L., Clemente, G., Félix, J., Weimer, J., Lee, I. (2019). Intelligent and dynamic ransomware spread detection and mitigation in integrated clinical environments. *Sensors*, *19*(5), 1114.

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R. (2017). Know abnormal, find evil: frequent pattern mining for ransomware threat hunting and intelligence, IEEE transactions on emerging topics in computing.

Takeuchi, Y., Sakai, K., Fukumoto, S. (2018). Detecting ransomware using support vector machines. In *Proceedings of the 47th International Conference on Parallel Processing Companion. ACM, pp. 1*.

Bridges, R.A., Glass-Vanderlan, T.R., Iannacone, M.D., Vincent, M.S., Chen, Q. (2019). A survey of intrusion detection systems leveraging host data. *ACM Computing Surveys (CSUR)*, *52*(6), 1–35.

Lou, J.G., Fu, Q., Yang, S., Xu, Y., Li, J. (2010). Mining invariants from console logs for system problem detection. In *USENIX Annual Technical Conference. pp. 23–25*.

Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I. (2009). Detecting large-scale system problems by mining console logs. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles. ACM, pp. 117–132*.

Liang, Y., Zhang, Y., Xiong, H., Sahoo, R. (2007). Failure prediction in ibm bluegene/l event logs. In *Seventh IEEE International Conference on Data Mining (ICDM 2007). IEEE, pp. 583–588*.

Zhang, K., Xu, J., Min, M.R., Jiang, G., Pelechrinis, K., Zhang, H. (2016). Automated it system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data). IEEE, pp.1291–1300*.

Ahmadian, M.M., & Shahriari, H.R. (2016). 2entfox: A framework for high survivable ransomwares detection. In *2016 13th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC). IEEE, pp. 79–84*.

Kharaz, A., Arshad, S., Mulliner, C., Robertson, W., Kirda, E. (2016). {UNVEIL},: A large-scale, automated approach to detecting ransomware. In *25th {USENIX} Security Symposium ({USENIX} Security 16), pp. 757–772*.

Lee, J.K., Moon, S.Y., Park, J.H. (2017). Cloudrps: a cloud analysis based enhanced ransomware prevention system. *The Journal of Supercomputing*, *73*(7), 3065–3084.

Verma, M.E., & Bridges, R.A. (2018). Defining a metric space of host logs and operational use cases. In *2018 IEEE International Conference on Big Data (Big Data), pp. 5068–5077*.

Morato, D., Berrueta, E., Magañaa, E., Izal, M. (2018). Ransomware early detection by the analysis of file sharing traffic. *Journal of Network and Computer Applications*, *124*, 14–32.

Hardy, W., Chen, L., Hou, S., Ye, Y., Li, X. (2016). Dl4md: a deep learning framework for intelligent malware detection. In *Proceedings of the International Conference on Data Mining (DMIN). The Steering Committee of The World Congress in Computer Science, Computer, p. 61*.

Homayoun, S., Dehghantanha, A., Ahmadzadeh, M., Hashemi, S., Khayami, R., Choo, K.K.R., Newton, D.E. (2019). Drthis: Deep ransomware threat hunting and intelligence system at the fog layer. *Future Generation Computer Systems*, *90*, 94–104.

Rhode, M., Burnap, P., Jones, K. (2018). Early-stage malware prediction using recurrent neural networks. *Computers & Security*, *77*, 578–594.

Krishna Chandra Roy received his B.Sc. degree in Electronics and Communication Engineering from Khulna University of Engineering and Technology (KUET), Bangladesh in 2014. Currently, he is pursuing his Ph.D. degree in Electrical and Computer Engineering from the University of Texas at San Antonio (UTSA). His primary research focused on IoT Security and application of AI in Cyber Security. His research interest also includes Deep Learning applications and Computer Vision.

Dr. Qian Chen is an Assistant Professor with the Department of Electrical and Computer Engineering at the University of Texas at San Antonio (UTSA). She earned her Ph.D. degree in Electrical and Computer Engineering from Mississippi State University in 2014. Dr. Chen's primary research area is autonomic computing and cyber security. Her research topics including Human Factors and their impacts on Cybersecurity, Healthcare Information System and IoMT Security, Industrial Control Systems Security(SCADA and IIoT), Software Vulnerability, and End-to-end Security Solutions.