Information Sciences 601 (2022) 162–174

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

A novel approach based on adaptive online analysis of encrypted traffic for identifying Malware in IIoT

Zequn Niu [a], Jingfeng Xue [a], Dacheng Qu [a], Yong Wang [a], Jun Zheng [b,*], Hongfei Zhu [c,*]

[a] School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[b] School of Cyberspace Security, Beijing Institute of Technology, Beijing 100081, China
[c] College of Computer and Technology, Chongqing University of Posts and Telecommunications, Chongqing 400065, China

ARTICLE INFO

Article history:
Received 8 November 2021
Received in revised form 20 January 2022
Accepted 7 April 2022
Available online 11 April 2022

Keywords:
Malware
Encrypted traffic
IIoT
Online learning
Adaptive random forests

ABSTRACT

The continuous emergence of new malware has been a severe threat to Industrial Internet of Things (IIoT), while identifying malware through detecting malicious traffic in encrypted, drift, and imbalanced traffic streams is a challenge. This paper proposes an approach based on adaptive online analysis to accurately determine the families of malware by analyzing traffic streams which are encrypted, drift, and imbalanced. This approach is based on Improved Adaptive Random Forests (IARF), to obtain the ability of adaptive update of parameters when processing new types of malware traffic in traffic streams and being sensitive to families of malware with few samples in imbalanced traffic. We build a prototype of this approach and evaluate the performance through experiments. The experiments are based on a mixed dataset composed of data from malware-traffic-analysis.net, Lastline Inc, MCFP dataset, and CTU-13 dataset. In addition, our approach is also compared with three state-of-the-art methods. The results of the experiments show that we have obtained a 99.66% F1-score in the classification of malware families, and our classifier also performs better than the other classifiers.

1. Introduction

The continuous emergence of new malware is a serious threat to Industrial Internet of Things (IIoT), and with the encryption of traffic in IIoT, detecting new malware through traffic becomes more challenging [29]. Traditional malicious traffic detection methods such as Deep Packet Inspection (DPI) [1] need to detect the plaintext content of the traffic, whereas these methods are not effective when the traffic packets are encrypted. Some methods for detecting encrypted traffic like Man-In-The-Middle (MITM) [3] need to decrypt traffic packets and then check the plaintext content, which conflicts with the purpose of privacy protection and consumes a lot of resources in the process of decryption and re-encryption.

To detect malware in encrypted traffic without decrypting the packets, researchers have used machine learning techniques to analyze the unencrypted headers and statistical features of encrypted traffic data [4,5]. However, in practical applications of machine learning-based models to identify malware through detecting encrypted traffic, the following challenges remain:

* Corresponding authors.
E-mail addresses: zhengjun@bit.edu.cn (J. Zheng), zhuhf@cqupt.edu.cn (H. Zhu).

https://doi.org/10.1016/j.ins.2022.04.018
0020-0255/© 2022 Elsevier Inc. All rights reserved.

(1)Traffic data for model training may not be fully obtained in the initial stage, but continuously provided in the form of streaming data, which requires the model learning from traffic streams and updating frequently. Also, the continuous emergence of new malware traffic will cause the feature distribution of the traffic to change continuously, which requires adaptive adjustment of model to capture the new malware traffic. At present, most models are based on offline machine learning algorithms, which means if there are new samples for learning, expensive costs and time overhead have to be paid for offline models to retrain and redeploy [7,9].

(2)The traffic data are imbalanced, benign traffic flows are much more than malicious traffic flows. Because of the imbalanced data, machine learning-based models will bias towards the frequent records, which leads to the degradation of model classification performance [11,16].

In response to the above challenges, we propose an algorithm, i.e., Improved Adaptive Random Forests (IARF), based on Adaptive Random Forests (ARF) [12], to identify malicious traffic streams which are drift and imbalanced. Compared with ARF, IARF has a sampling strategy that samples are pre-classified with the classifier and only the misclassified samples in pre-classification are used for training. IARF is constructed in online mode and able to update the parameters by training with one sample, which means once samples of new malware arrive, IARF can update the parameters timely to handle the new malware. At the same time, IARF is more flexible and economical than offline learning-based methods on updating because IARF does not require re-training and re-deploying. In addition, IARF is more sensitive than ARF to malware families with few samples in imbalanced traffic, which is important to discover cyber attack that more hidden but could cause serious damage such as controlled information stealing attack [30,31]. Further, with feature extraction to encrypted traffic packets through Brim, an open source network analysis tool, we propose a new encrypted malicious traffic detection approach based on IARF and Brim. The approach aims to accurately detect malware families in traffic streams which are encrypted, drift and imbalanced in IIoT.

We validated our approach on real datasets. The data are collected from public encrypted traffic datasets on the Internet. The malicious traffic data are generated by a variety of malware in different families running in a simulated environment and the benign traffic data are captured in the network by the creators of these datasets. We have validated our approach on the mixed datasets, which consists of the data from malware-traffic-analysis.net, Lastline Inc, MCFP dataset, and CTU-13 dataset. The F1 score of 99.66% was obtained in the experiments, which proved the feasibility of our approach.

The contributions of our work are as follows:

(1)We propose a new encrypted malicious traffic detection approach based on adaptive online analysis, which can identify the families of malware through detecting encrypted traffic streams. The size of input data for training the model can be 1.

(2)We propose an Improved Adaptive Random Forests algorithm, by changing the sampling method on training data for Adaptive Random Forests. The IARF-based model has better performance on imbalanced encrypted traffic data than the ARF-based model.

(3)We compare our approach with three state-of-the-art approaches in term of classification performance on encrypted malicious traffic and the experimental results show the superiority of our approach.

The rest of the paper is organized as follows: section 2 discusses the related work, section 3 presents preparatory knowledge about our approach, section 4 details the design methodology of our approach, section 5 evaluates the performance of our proposed approach, and section 6 concludes the paper.

2. Related work

Malware identification through anomaly traffic detection is one of the network behavior based malware detection methods [32]. Traditional anomaly traffic detection method such as Deep Packet Inspection (DPI) [1,2] is widely used to detect non-encrypted traffic. DPI determines the legitimacy of traffic by deeply inspect the payload of the packets and has a high accuracy for anomaly detection. However, because of the need to view the content of the traffic, DPI is not applicable for encrypted traffic detection. Callegeti et al. [3,6] propose to decrypt the encrypted traffic data using a Man-In-The-Middle approach, so that the traditional anomaly detection method can still work on the decrypted data. But, this approach is not consistent with the purpose of traffic encryption to protect privacy, and the process of decrypting and re-encrypting packets consumes a lot of resources.

Compared with the detection method with decrypting ciphertext, methods based on machine learning algorithms have shown a unique advantage in detecting encrypted traffic. Machine learning-based models usually use the features extracted from the traffic without decryption. The features that are widely used in the research can be divided into three categories: metadata features, statistical features, and unencrypted TLS header features. Metadata features are the basic features, containing the basic information of data streams such as five-tuples. Statistical features are features obtained by statistical analysis of traffic flows, which are usually not displayed directly, but need to be extracted by statistical calculation. Unencrypted TLS header features are introduced when the traffic is encrypted with TLS (Transport Layer Security) protocol, which is a cryptographic protocol that provides privacy for applications. These features contains SSL connection features, certificate features, etc. McGrew et al. [4,8] extracts features including metadata features, packet length, byte distribution, and TLS header information features from encrypted traffic, and then uses logistic regression-based classifiers to identify encrypted traffic flows. Further, Anderson et al. [10,17] introduces background traffic such as DNS context flows linked to encrypted flows and headers of HTTP context flows from the same source IP within a 5-min window as additional features for learning, which able to obtain a 0.00% false discovery rate when detecting malicious traffic in their experiments. Torroledo et al. [5] extracts self-signed or free-generated certificates in encrypted network sessions and classifies the certificates using RNN + LSTM models, the experimental results show that their method has an accuracy of 94.87% in identifying malicious certificates and 88.64% in identifying phishing certificates.

In practical application, the evolution of benign and malicious traffic changes the distribution of features extracted from traffic data, which requires detection techniques with the ability to capture the drift of traffic, and thus encrypted traffic detection based on incremental learning and online learning has started to gain attention in the industry. Deep learning techniques are widely studied in cyber attack detection [33,34], however, the offline training mode of deep learning makes deep learning based detection models unable to handle evolving traffic streams, so researchers are focus on employing online learning techniques to detect streaming traffic. Lee et al. [14,18] compares the effectiveness of representative incremental learning algorithms for encrypted traffic detection and finds that the Srochastic Gradient Descent (SGD)-based method is effective in detecting anomalous encrypted traffic. Liu et al. [19] proposes MalDetect to detect the malicious traffic. To improve the detection efficiency, MalDetect capture only the first 8 packets of the traffic flows for feature extraction, and finish the test task before the encrypted application data transmission, which gives the control system the opportunity to reduce the malicious behavior in real time. However, the dataset used in Liu et al. [15,19] is balanced, but in practical application the benign flows are much more than the malicious flows.

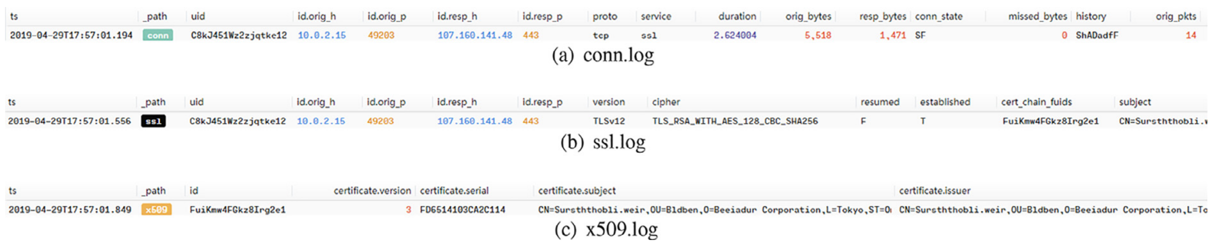
In our approach, a sample filtering strategy is used to choose the misclassified samples in pre-classification for training instead of using the whole data streams, to reduce the impact of imbalanced data on the classifier. We propose the IARF algorithm based on the ARF algorithm with this sample filtering strategy, and experimental results show IARF-based classifier has a better performance when dealing with the imbalanced traffic.

3. Preliminaries

3.1. Brim

Brim is an open source network analysis tool developed by Brim Security, Inc. and built from several open source components, including: Zed, a structured log query engine; Electron and React for multi-platform user interfaces; and Zeek, which generates network analysis data from packet capture files. Brim reads pcap files and converts them into Zeek logs, and then exports the logs in the form of csv or other format files. Zeek is an open source network traffic analysis software, formerly known as BrO. Zeek/BrO generates different network traffic logs to record different aspects of traffic properties, for example, conn.log records the information about IP, TCP, and ICMP connection details, ssl.log records the information about SSL/TLS handshake details, and x509.log records the information about the x509 certificate analysis data. The conn.log, ssl.log and x509.log are also the log files used in this paper for feature extraction, which show the status of encrypted traffic from different perspectives. Fig. 1 shows the records of same session from conn.log, ssl.log, and x509.log in Brim.

In conn.log, each record corresponds to a network session, i.e., a client–server communication process that opens with three handshakes and closes with a closed connection. For the same encrypted session, different logs are linked by correlation keys. For example, the same encrypted session shown in Fig. 1 has the same value in record "uid" in both conn.log and ssl.log, and the connection between conn.log and ssl.log with the same "uid" enables the aggregation of records of the same session. Similarly, the connection between ssl.log and x509.log relies on the record "cert_chain_fuids" in ssl.log, which contains the id of each certificate in the certificate chain that validates the encrypted session, corresponding to the record "id" in x509.log. This article takes the first record in "cert_chain_fuids" that represents the end-user certificate and connects it to the corresponding "id" in x509.log.



ts	_path	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	proto	service	duration	orig_bytes	resp_bytes	conn_state	missed_bytes	history	orig_pkts
2019-04-29T17:57:01.194	conn	C8kJ451Wz2zjqtke12	10.0.2.15	49203	107.160.141.48	443	tcp	ssl	2.624004	5,518	1,471	SF	0	ShADadfF	14

(a) conn.log

ts	_path	uid	id.orig_h	id.orig_p	id.resp_h	id.resp_p	version	cipher	resumed	established	cert_chain_fuids	subject
2019-04-29T17:57:01.556	ssl	C8kJ451Wz2zjqtke12	10.0.2.15	49203	107.160.141.48	443	TLSv12	TLS_RSA_WITH_AES_128_CBC_SHA256	F	T	FuiKmw4FGkz8Irg2e1	CN=Surstthobli.

(b) ssl.log

ts	_path	id	certificate.version	certificate.serial	certificate.subject	certificate.issuer
2019-04-29T17:57:01.849	x509	FuiKmw4FGkz8Irg2e1	3	FD6514103CA2C114	CN=Surstthobli.weir,OU=Bldben,O=Beeiadur Corporation,L=Tokyo,ST=O	CN=Surstthobli.weir,OU=Bldben,O=Beeiadur Corporation,L=To

(c) x509.log

Fig. 1. Records of the same session in Brim.

3.2. Adaptive random forests

Adaptive Random Forests (ARF) [12] algorithm is an adaptation of the classic random forest algorithm to streaming data. The original random forest grows decision trees as base classifiers, and decides the final output by voting on each decision tree. The presence of the random mechanism in random forest makes it resistant to overfitting and insensitive to the default values of the samples. The random mechanism of the random forest comes from two components: (1) If the feature dimension of the sample is M, specify a constant $m \ll M$, randomly select m features from M features as the set of features acting on the tree, and select the best feature from this feature set each time the tree is going to split. (2) If the size of the training set is N, for each tree, n training samples ($n < N$) are taken from the training set in a bootstrap way to form a subset for training [13].

However, the prerequisite to make this random mechanism effective in random forest is having the full dataset, which is in conflict with streaming data. The adaptability of random forest to streaming data requests a proper online bootstrap aggregation process and limiting the leaf split decision to a subset of the features.

For the requirement of online bootstrap aggregation process, ARF adopts online bagging algorithm [20] with Poisson ($\lambda = 6$) to achieve online bootstrap sampling. The requirement for the restriction of feature splitting is achieved by modifying the induction algorithm of the base tree in ARF. The base tree in ARF is based on the Hoffeding tree algorithm [21] but changed in leaf split strategy. The full set of features is used during leaf split in Hoffeding tree algorithm, but in ARF, whenever a new node is created in the base tree, a random subset of features is selected and the next splitting selection is restricted to this subset of features.

One advantage of ARF is the adaptation to data streams with concept drift. Concept drift describes unforeseeable changes in the underlying distribution of streaming data over time [28]. To deal with the drifting data streams, the ensemble algorithms are often coupled with drift detectors, and the default approach is to reset learners immediately after a drift is signaled [12]. However, the resettable learners may decrease the classifier performance of ensemble models due to lack of training. In order to solve this issue, Adaptive Random Forests does not reset the base classifier once the concept drift is detected, but instead, a two-stage setup of drift and warning detection is performed. The two-stage setup contains warning threshold and drift threshold. When a warning threshold is triggered, a new tree is created in the "background" as a background tree which is trained together with the working tree, and when a drift detection threshold is triggered, the working tree that triggers the drift detection threshold is replaced with the background tree. By this method, the ARF ensures that the base classifier in the forests can effectively detect the evolving data streams.

4. Method

4.1. Overall architecture

Our approach consists of a log transformation component, a preprocessing component, and an IARF-based model, as shown in Fig. 2. We use Brim in the log transformation component to write the records of the pcap files to the logs, then use the preprocessing component for feature extraction, and finally use our proposed IARF-based model to make a classification on the type of traffic flows.

Combined with the overall architecture diagram in Fig. 2, the main process of our approach is as follows:

(1)Import the pcap file to Brim.
(2)Process the pcap file through Brim and output the Zeek log. The conn.log, ssl.log and x509.log in the output logs are selected as the input of the preprocessing component.
(3)The tasks of the preprocessing component include record connection, feature extraction, and label encoding. In the preprocessing component, the records describing the same session in different logs are connected into an aggregated sample of the session through correlation keys. Then, features related to traffic encryption are extracted from the aggregated sample. Label encoding is also performed for the models in next step to recognize the sample.
(4)Sent the sample into every base tree in the forest for test.
(5)The base trees send their own test results to the voting&judging module, and the voting&judging module makes the prediction based on the result of the majority vote of base trees.
(6)Output the prediction result.
(7)The prediction result of the model is compared with the true label of the sample, if different, the sample is sent to the online bagging module, so that the model can be trained with the wrong-predicted sample.
(8)The online bagging module distributes the samples to the base trees according to a Poisson ($\lambda = 6$) distribution and all the background trees in the forest, then the trees that receive the samples are trained.
(9)The trained base tree performs warning and drift detection of concept drift.
(10)The drift detection value of each base tree is compared with the thresholds of warning/drift detection. If there is a base tree that reaches the warning threshold, a background tree is built; if any base tree reaches the drift threshold, then the tree will be replaced by the background tree.

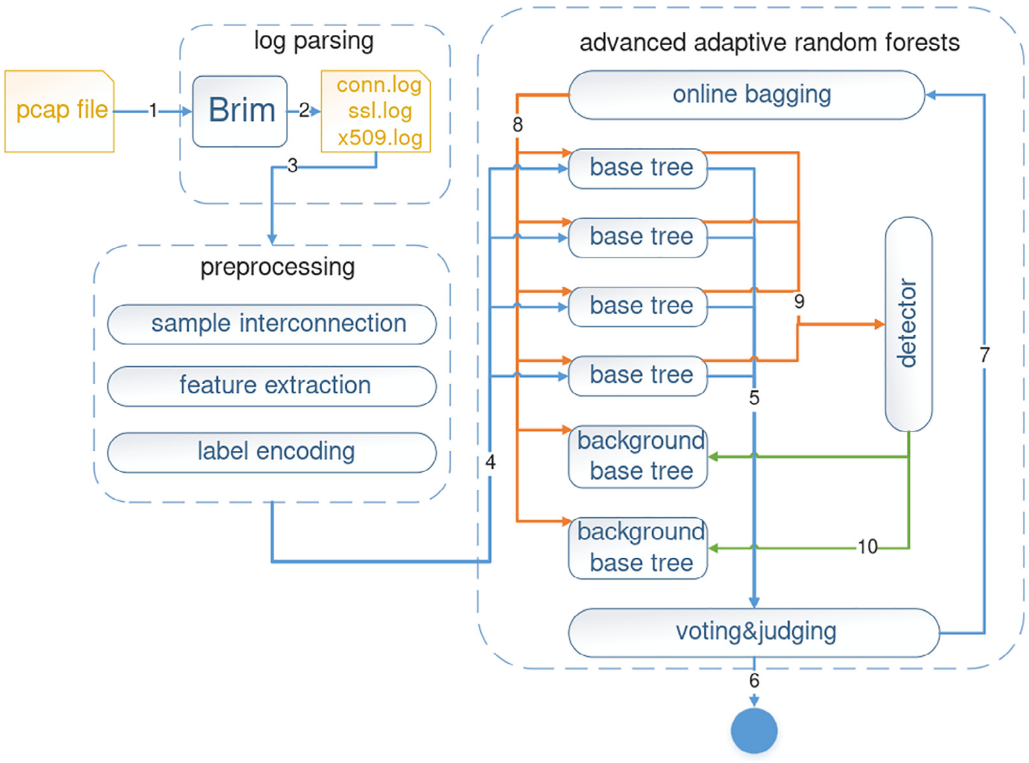


Fig. 2. The architecture of IARF-based detector.

4.2. Log transformation and preprocessing

The task of the log transformation component is to convert pcap files into Zeek logs using Brim, and the task of the preprocessing component is to concatenate records scattered in different Zeek logs describing the same session and then perform feature extraction. In our feature extraction work, the network session is taken as the basic unit of traffic data. The process of network session encrypted with the TLS protocol is shown in Fig. 3. First, the client sends a request to the server side to establish an encrypted connection with the web server. After the web server receives the request from the client, the certificate information (public key information, etc.) of the website will be sent to the client, then the client sends the symmetric key encrypted by the public key to the server, and after the server decrypts it with its own private key, it can communicate with the client in cipher text. The encrypted traffic is the application data that the client interacts with the server, and features cannot be effectively extracted from this cipher text. Therefore, our feature extraction focuses on the protocol negotiation process before the encryption interaction and the overall statistics of the session, such as the size of the upstream and downstream traffic packets.

We use Brim to analyze pcap files and generate Zeek logs for feature extraction. In this paper, we focus on encrypted traffic analysis, so we selected conn.log, ssl.log and x509.log which are related to encrypted traffic as the source logs for the feature extraction. For an encrypted session, the records are scattered in these three logs, so we interconnect the different logs by the key values which is described in Section 3. Based on experiment, we extract the features as shown in Table 1. Because "server_name" in ssl.log, "san.dns" and "san.ip" in x509.log are in string type that cannot be processed by the classifier and needs to be converted to a numeric form. inspired by Anderson et al. [17], we converted "server_name" to "server_name_len" which represents the length of the server domain, "san.dns" to "san_dns_num" for the length of dns, and "san.ip" to "san_ip_num" which represents the length of IP address. Also, since some malicious websites can be disguised by registering free legitimate certificates, which is a nonprofit certificate authority providing TLS ceriticates to websites, we add the feature "lets_encrypt" to determine if the certificate of the session was registered at letsencrypt.org. Since our model cannot be feed directly with samples with categorical features, we convert the all the categorical features into integer ones with the Label-Encoding technique, such as 0 representing "OTH", 1 representing "REJ" and 2 representing "RSTO" in the feature "conn_state".

4.3. IARF

We propose an Improved Adaptive Random Forests algorithm based on ARF with changing the sampling method. ARF is able to perform uninterrupted learning when traffic is continuously reached in the actual environment, which makes it suitable for the traffic streams detection, and its strategies for concept drift also makes it adaptable to evolving data streams. However, when detecting the malicious traffic, ARF-based models will bias towards benign class because of the imbalance between benign and malicious samples. In traffic, benign flows are much more than flows made by malware, but most of the benign flows are redundant to the models. For classifier, learning the whole traffic streams which include a large number of benign samples does not help to improve the performance, but makes the model favor the benign class in classification. Moreover, a lot of time is taken when learning a large number of benign samples because ARF needs to update the parameters every time a new sample is received. Therefore, we design an Improved Adaptive Random Forests algorithm based on the ARF algorithm. The idea is to make the classifier to pre-classify the sample when receiving a new sample, and if the prediction is correct, the sample is discarded and not used for training the classifier, only the sample that the model classifies incorrectly is used for training.

Algorithm 1: Improved Adaptive Random Forests Algorithm

Symbols: **m**:maximum features for per split; **n**:number of base trees; **α**:warning threshold; **β**:drift threshold; **F**:forest of base trees; **D**:data stream;
BT:background trees; **W(t)**:weight of Tree(t) **P**: learning performance estimation function; **nt**:new background tree.

```
1: Function TrainImprovedAdaptiveRandomForests(m,n,$\alpha$,$\beta$)
2: while HasNext(D) do
3:    $(x, y) \leftarrow next(D)$
4:    $y^p \leftarrow F.predict(x)$
5:    if NotSame($y, y^p$) then
6:       for all $t \in F$ do
7:          $y' \leftarrow t.predict(x)$
8:          $W(t) \leftarrow P(W(t), y', y)$
9:          BaseTreeTrain(x, y, t, m)
10:           if WarningDetected(x, y, t, $\alpha$) then
11:              $nt \leftarrow CreateTree()$
12:              $B(t) \leftarrow nt$
13:           if DriftDetected(x, y, t, $\beta$) then
14:              $t \leftarrow B(t)$
15:           for all $nt \in B$ bf do
16:              BaseTreeTrain(x, y, t, m)
17: End function
```

The pseudo-code of IARF is shown Algorithm 1. We pre-classify the samples before using them for training (lines 3,4,5), and if the prediction of model matches the true label of the sample, this sample will not be used for training, otherwise this sample is used to learn (lines 6,7,8,9). Through experiments, we found this approach can significantly improve the efficiency without reducing the detection performance, and can reduce the impact of data imbalance on the model at the same time. For malware families that account for less than 1% of the full dataset, the IARF-based model can still obtain more than 0.8 F1-score detection effect.

5. Experiment

5.1. Dataset

The data used in this paper are collected and combined from malware-traffic-analysis.net, Lastline Inc, MCFP dataset, and CTU-13 dataset. These data sources all contain encrypted traffic data, which are captured in real network or by running the malware in simulated environment such as sandboxes. We use encrypted malicious traffic packets collected by Olivier Roques from malware-traffic-analysis.net, Lastline Inc, and the MCFP dataset. Since publicly available data collected by Olivier Roques only contains malicious traffic packets, which makes that benign samples need to be obtained from other sources, we add benign data from CTU-13 dataset and MCTP dataset to expand our dataset. Our goal is encrypted malicious traffic detection, which requires samples of encrypted traffic, so we only use data with "ssl" in the "service" feature of the samples which means the session is encrypted with SSL or TLS protocol. The specific components of our dataset are as follows.

(1)Malware-traffic-analysis.net. This is a website that has a long history of focusing on traffic made by malware and available for downloading malicious traffic packets. The data collected from this website and used in this paper contains encrypted traffic packets from 2014.1 to 2019.6 with a total of 323 pcap packet files containing 17,405 encrypted sessions.
(2)Lastline Inc. Encrypted malicious traffic collected in network of Lastline, containing 2823 encrypted sessions.

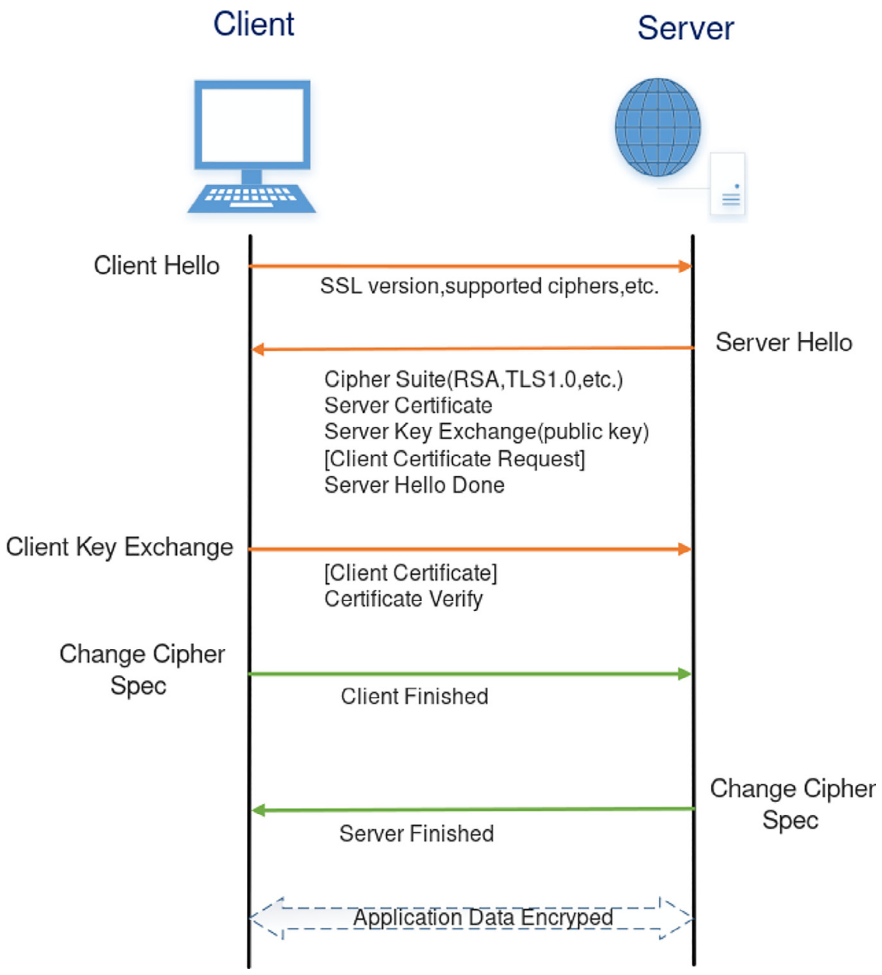


Fig. 3. The process of session encrypted with TLS protocol.

(3)MCFP [22]. MCFP(Malware Capture Facility Project) provides a number of long-term capture packets of benign and malware that use TLS for communication. The malicious traffic data from MCFP contains 31 pcap files, including 675689 encrypted malicious sessions. For benign sessions, we used packets from CTU-Normal-20 to CTU-Normal-32 of MCFP, which are all benign encrypted traffic captures, containing 69,358 encrypted sessions.
(4)CTU-13 [23]. This is a tagged dataset with botnet, normal and background traffic. Since the logs generated by Zeek/BrO are already available in the dataset, we directly use the logs from CTU-13, which contains 487249 encrypted sessions.

Because in the real environment there are much more benign traffic than malicious traffic, in order to get closer to reality, we limit the size of malicious samples. For malware families with more than 10,000 sessions, the number of malicious sessions is limited to 10,000 by random sampling, so that the ratio of benign sessions to malicious sessions is about 10:1. We selected 9 malware families of which the number of sessions is more than 300 as malicious samples to participate in our experiments. The distribution of the participating samples is shown in Table 2, where Normal is the benign sample and the rest are the individual malware families.

5.2. Experiment environment

This paper was conducted in windows 10 with Intel i7-6700@3.40 GHz and 32G RAM. The prototype of our approach was written in Python 3.7 using the sklearn library and River library [24].

5.3. Experiment description and results

Firstly, we evaluate ARF, IARF and several offline learning algorithms on malware family classification. Both ARF and IARF are implemented based on the estimator”AdaptiveRandomForestClassifier” in the River library, and the number of base trees

Table 1
Features extracted from conn.log, ssl.log and x509.log.

Num	Feature name	Log	Description
1	proto	conn.log	The transport layer protocol of the connection
2	service	conn.log	An identification of an application protocol being sent over the connection
3	duration	conn.log	How long the connection lasted
4	conn_state	conn.log	Connection state
5	history	conn.log	Records the state history of connections
6	orig_bytes	conn.log	The number of payload bytes the originator sent
7	resp_bytes	conn.log	The number of payload bytes the responder sent
8	orig_pkts	conn.log	Number of packets that the originator sent
9	orig_ip_bytes	conn.log	Number of IP level bytes that the originator sent
10	resp_pkts	conn.log	Number of packets that the responder sent
11	resp_ip_bytes	conn.log	Number of IP level bytes that the responder sent
12	missed_bytes	conn.log	Indicates the number of bytes missed in content gaps
13	version	ssl.log	SSL/TLS version that the server chose.
14	cipher	ssl.log	SSL/TLS cipher suite that the server chose
15	server_name_len	ssl.log	Length of the Server Name Indicator SSL extension
16	resumed	ssl.log	Flag to indicate if the session was resumed
17	established	ssl.log	Flag to indicate if this ssl session has been established
18	next_protocol	ssl.log	successfully
19	validation_status	ssl.log	Result of certificate validation for this connection
20	certificate.version	x509.log	The version of the encountered SCT
21	certificate.key_alg	x509.log	Name of the key algorithm
22	certificate.sig_alg	x509.log	Name of the signature algorithm
23	certificate.key_type	x509.log	Key type, if key parseable openssl
24	certificate.key_length	x509.log	Length of the key
25	curve	x509.log	Curve, if EC–certificate
26	san_dns_num	x509.log	Numbers of DNS entries in the SAN
27	san_ip_num	x509.log	Numbers of IP entries in the SAN
28	basic_constraints.ca	x509.log	Basic constraints extension of the certificate
29	lets_encrypt	x509.log	Flag to indicate if the certificate was registered at letsencrypt.org

Table 2
Samples of different families in our dataset.

Family	Sessions	Percentage
Normal	509398	92.07%
Dridex	10000	1.81%
Trickbot	10000	1.81%
Vawtrak	10000	1.81%
Miuref	7175	1.30%
Zeus	2391	0.43%
Hancitor	2194	0.40%
Zeus-panda	1194	0.21%
Dreambot	481	0.09%
Gootkit	412	0.07%
Total	553245	100.00%

used in the algorithm is 10, with the rest parameters using default parameters. Secondly, we compare the performance of ARF and IARF on each malware family, and provide a deep analysis for the different performance between ARF and IARF by comparing the actual number of samples used by the two algorithms to train the models. Thirdly, we design an experiment to evaluate ARF and IARF when concept drift happened on traffic streams (new malware family occurs). Finally, we compared our method with three other representative methods.

5.3.1. Comparison with offline learning algorithms

We compared ARF, IARF and some offline machine learning algorithms in terms of accuracy and time overhead. IARF and ARF are online learning algorithms where the training set is in the form of streaming data, and the rest of the algorithms are offline machine learning algorithms where the whole training set is obtained at the beginning stage. For family classification, we use 10-fold cross validation for the experiments. The results are shown in Table 3.

As seen in Table 3, both ARF and IARF perform well in terms of precision, recall and F1score, with IARF slightly outperforming the ARF algorithm. Whereas in terms of training time, the time IARF consumed is much less than ARF. In the comparison with the offline learning algorithm, IARF performs second only to random forest. It is worth mentioning that IARF learns from the streaming data, whereas random forest-based model is trained with the complete training set at once.

Table 3
Result of algorithms on our dataset.

	Precision	Recall	F1score	Training Time	Learning Sample
IARF	**99.68%**	**99.67%**	**99.66%**	**425.05s**	**3179**
ARF	98.78%	98.95%	98.73%	2395.57s	497920
LRCV	84.76%	92.07%	88.26%	2686.92s	497920
Linear SVC	96.83%	97.46%	96.83%	1390.87s	497920
R.Forest	99.96%	99.96%	99.96%	6.80s	497920
GBDT	99.36%	99.18%	99.24%	1488.68s	497920

Table 4
Comparison of the family classification results of ARF and IARF.

	IARF			ARF			
Family	Pricision	Recall	F1-score	Pricision	Recall	F1-Score	Support
Normal	99.85	99.89	99.87	98.99	99.99	99.49	50940
Dridex	99.24	99.51	99.38	98.55	99.28	98.90	1000
Trickbot	99.52	98.81	99.16	99.69	96.63	98.13	1000
Vawtrak	99.97	99.73	99.85	98.14	99.98	99.04	1000
Miuref	94.19	95.25	94.66	99.66	70.83	82.10	718
Zeus	95.76	90.51	92.51	99.92	59.22	71.42	239
Hancitor	89.48	94.82	91.88	72.19	62.82	66.53	219
Zeus-panda	91.48	73.64	80.51	78.80	34.87	44.31	119
Dreambot	98.22	94.61	96.35	59.13	13.05	17.88	48
Gootkit	97.53	88.72	92.69	58.75	25.67	31.81	41

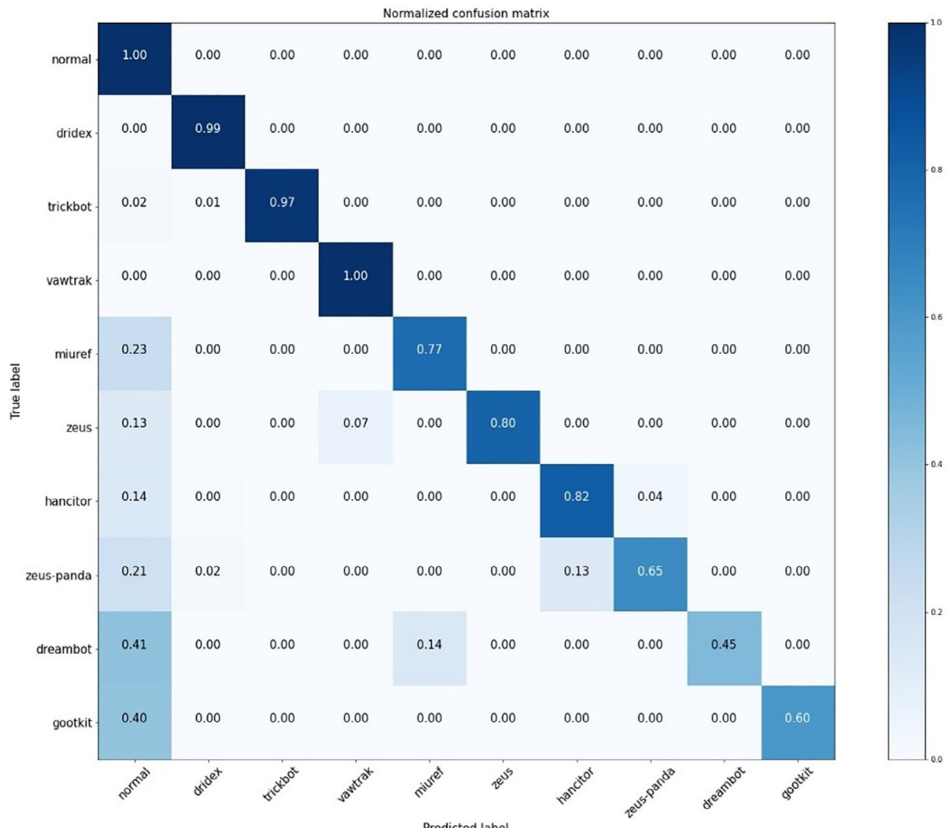
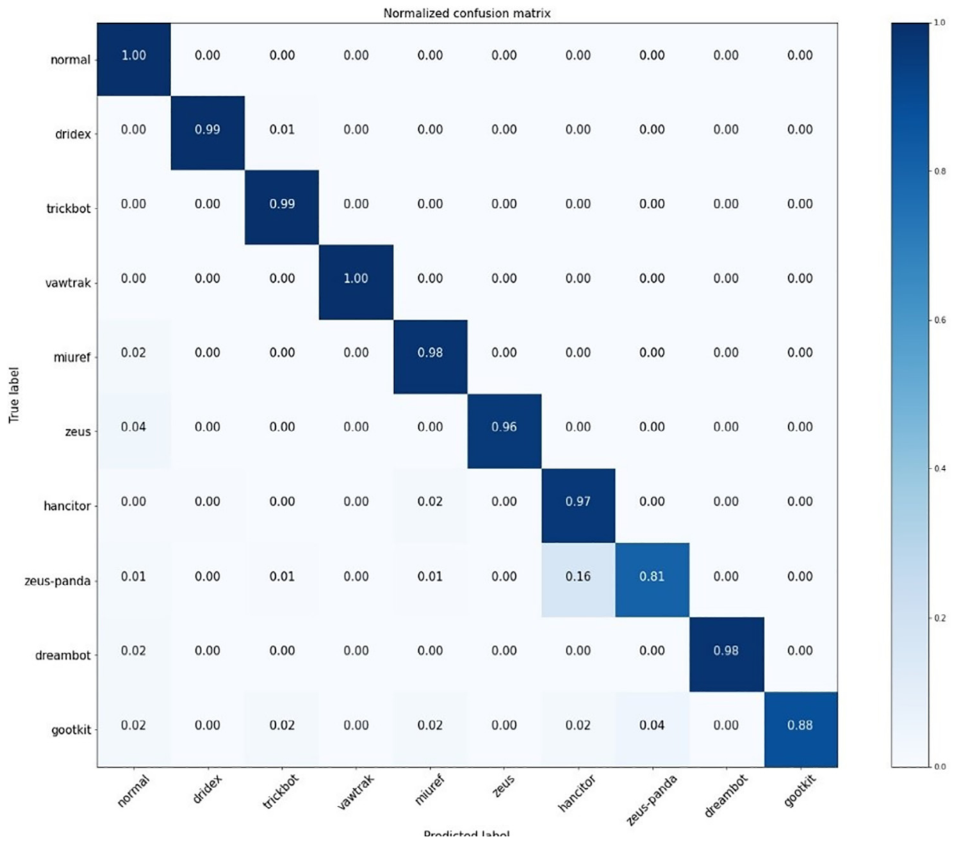


Fig. 4. Confusion matrix of ARF.

5.3.2. Evaluating the impact of imbalanced traffic streams

We made a comparison between ARF and IARF on each malware family, for deeply analysis of ARF and IARF when facing imbalanced traffic streams. Table 4 shows the classification result of ARF and IARF for each family, where support is the number of samples accounted for by each family in the test set. As seen in Table 4, ARF and IARF perform similarly and well on Normal, Dridex, Trickbot and Vawtrak, however in the classification performance of the other families, the IARF can be maintained at more than 80%, but the ARF has a significant decrease. To get more detail, the confusion matrix of the classification results of the two models are shown in Fig. 4 and Fig. 5.

From Fig. 4 and Fig. 5, it can be seen that on IARF-based model, most of the samples are correctly classified except for zeus-panda, of which the samples are misclassified as hancitor families. In contrast, in metric of the ARF, all the families with sample size less than 10,000 suffer from severe sample imbalance, and some samples of minority malware families are mis-



	normal	dridex	trickbot	vawtrak	miuref	zeus	hancitor	zeus-panda	dreambot	gootkit
normal	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dridex	0.00	0.99	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
trickbot	0.00	0.00	0.99	0.00	0.00	0.00	0.00	0.00	0.00	0.00
vawtrak	0.00	0.00	0.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
miuref	0.02	0.00	0.00	0.00	0.98	0.00	0.00	0.00	0.00	0.00
zeus	0.04	0.00	0.00	0.00	0.00	0.96	0.00	0.00	0.00	0.00
hancitor	0.00	0.00	0.00	0.00	0.02	0.00	0.97	0.00	0.00	0.00
zeus-panda	0.01	0.00	0.01	0.00	0.01	0.00	0.16	0.81	0.00	0.00
dreambot	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.98	0.00
gootkit	0.02	0.00	0.02	0.00	0.02	0.00	0.02	0.04	0.00	0.88

Fig. 5. Confusion matrix of IARF.

Table 5
Training samples of ARF and IARF actually learning.

	IARF		ARF	
Family	Sessions	Percentage	Sessions	Percentage
Normal	998	0.296847	458461	0.920756
Dridex	208	0.061868	9026	0.018127
Trickbot	273	0.081202	8998	0.018071
Vawtrak	50	0.014872	8984	0.018043
Miuref	697	0.207317	6425	0.012904
Zeus	312	0.092802	2162	0.004342
Hancitor	308	0.091612	1974	0.003965
Zeus-panda	297	0.08834	1076	0.002161
Dreambot	92	0.027365	434	0.000872
Gootkit	127	0.037775	378	0.000759
Total	3362	1.000000	497918	1.000000

classified as benign. For deeper analysis, we counted the samples effectively learned by the two models in one batch of training, and the results are shown in Table 5.

As seen in Table 5, the number of samples that IARF learned is much less than the one of ARF, due to the sampling strategy in IARF that the model only learns misclassified samples in pre-classification. It can be seen that the number of benign samples learned by IARF is only 998, which only accounts for 29.68% of the training set IARF actually used, whereas benign samples have the percentage of 92.07% in the training set ARF learned. Compared with ARF, the samples of malware family such as Miuref account for 20.73% of the training set in IARF, and the percentage of other family samples has also increased, which makes the gap between the number of benign samples and the number of malicious samples not too huge. The influence of the sampling strategy on IARF is reflected in the improved detection rate of minority malware families, which can be seen in Table 3 and Table 5.

5.3.3. Evaluating the impact of evolving traffic streams

We conducted experiments to evaluate ARF and IARF when facing evolving data streams. We divided the training set and the test set in a 9:1 ratio, and then divide the training set into two subsets: training-subset1, which play the role of original traffic stream, and training-subset2, which as drift traffic stream. We select one malware family ”Dridex” as an emerging malware family and take the samples of ”Dridex” into training-subset2, while the rest of samples in training-subset1. The mixed training set is built by training-subset2 spliced after training-subset1, indicating that the traffic streams has drift at training-subset2. To simulate the different concept drift scenarios, we select different proportions of samples in training-subset1 to mix with the samples in training-subset2, and then conducted comparison experiments to evaluate the classifiers. Fig. 6 shows the F1score of ARF and IARF under different scenarios. The ratio in Fig. 6 means the ratio between the samples selected from training-subset1 and the samples of ”Dridex” in training-subset2, for example, 0.0 of ratio means that training-subset2 completely consists of samples of ”Dridex”, and 1.0 means half samples in training-subset2 are from training-subset1 and the other half are ”Dridex”.

As can see from Fig. 6 that both classifiers have F1score greater than 0.95 on the test set when ratio is greater than 0. IARF is slightly stronger than ARF. Ratio greater than 0 means that the traffic flows of new malware are interspersed with the original traffic, and the experimental results show that in this case, both classifiers are able to perform effective classification. When ratio equals to 0, IARF-based classifier can still maintain 96.54% of F1score, but the F1score of ARF is missing close to 0. By investigating the output of ARF, we found that all the prediction made by ARF are ”Dridex”. Then we count how many samples of ”Dridex” have been learned by the classifiers, and the results showed that ARF learned 9026 samples of ”Dridex” which are all the samples of ”Dridex” in training set, but IARF only learned 29 samples of ”Dridex”.

We believe that because of model training with samples of the same label for a long time, ARF has reset all the base trees to adapt the drift, thus the prediction of ARF is only ”Dridex”. On the contrary, due to the sampling strategy in IARF, most of the samples of ”Dridex” have been filtered out, making the model much less affected by the evolving traffic than the ARF.

5.3.4. Comparison with other methods

In this paper, we compare our approach with three other representative state-of-the-art methods: (1) D2LAD proposed by Xing et al. [25] is an online encrypted traffic detection method based on deep dictionary learning, which uses deep dictionary learning technique to obtain normal patterns based on the sequence features from the traffic, and finally determines

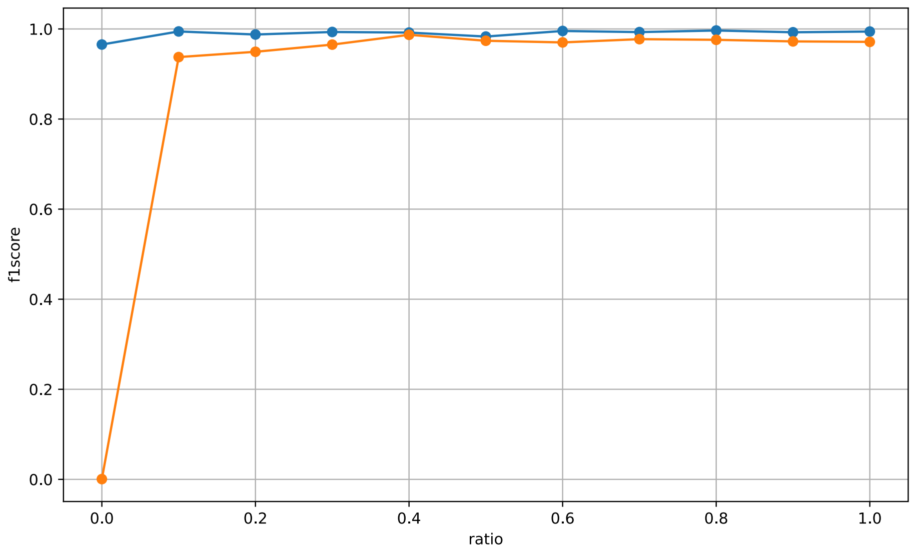


Fig. 6. F1score of ARF and IARF under different ratio.

Table 6
Comparison between IARF and other methods.

Algorithms	Precision	Recall	F1-score
IARF	**99.68%**	**99.67%**	**99.66%**
D2LAD [25]	96.11%	92.87%	94.46%
FS-Net [26]	95.21%	91.44%	93.29%
Method of [27]	97.22%	89.84%	93.11%

whether the traffic is anomalous by calculating the correlation between the original data and the deep dictionary to obtain an anomaly score on the data. (2) FS-Net proposed by C. Liu et al. [26] is an end-to-end classifier based on a multilayer encoder that learns features from the original traffic and then classifies the traffic by a unified framework. (3) method proposed by J. Liu et al. [27] is a distance-based approach that calculates the distance between malware by using an unsupervised learning algorithm Gaussian Mixture Model (GMM) and Ordering Point Identification Clustering Structure (OPTICS) to define the classes of malware and classify the malware. We compare the results of IARF with these three state-of-the-art methods, using the results of Xing et al. [25] who use a malicious/benign traffic ratio of 1:10 without adding noise labels. The comparison is shown in Table 6.

The results in Table 6 show that our method is more effective than other methods.

6. Conclusion

This research aims to identify malware in streaming traffic data which are encrypted, drift and imbalanced. The study contributes to our understanding of malicious traffic detection in complex environment, which is closer to reality. Based on ARF, an online learning algorithm, IARF is proposed to detect malware in encrypted traffic streams. IARF is able to update adaptively once new samples arrive, which is important to track new types of malware timely. In addition, IARF remains sensitive to malware families with few samples, which is crucial for detecting covert but serious threatening cyber attacks. Further, we propose a new encrypted malicious traffic detection approach based on IARF to accurately detect malware families in encrypted, drift and imbalanced traffic. We implement a prototype of the method and conduct experiments to evaluate the approach. The experimental results show that our approach performs significantly better than the original ARF in terms of accuracy and efficiency in the case of imbalanced, drift and encrypted traffic. Also, compared with three representative methods, our approach have a better performance.

A limitation of our study is that the feature extraction works on a completed session, which means the session has been over and the payload including the malicious payload has been transferred. Nevertheless, the best performance of an anomaly detection system is to detect and warn of malicious encrypted sessions before the malicious payload transferred, so that the system can prevent further interaction of the session in time. In this context, a more sophisticated study should be conducted, in feature extraction from only the handshake portion of the session, rather than the entire session.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62172042), the National Key Research & Development Program of China (2020YFB1712104), the Major Scientific and Technological Innovation Projects of Shandong Province (2020CXGC010116), the Natural Science Foundation of Chongqing (Grant No. cstc2020jcyj-msxmX0778), the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN201900608).

References

[1] S. Sen, O. Spatscheck, and D. Wang, Accurate, scalable in-network identification of p2p traffic using application signatures, in Proceedings of the 13th conference on World Wide Web - WWW '04, New York, NY, USA, 2004, pp. 512–521. https://doi: 10.1145/988672.988742..
[2] Q. Zhang, L. Zhu, Y. Li, et al, A group key agreement protocol for intelligent internet of things system, International Journal of Intelligent Systems. 1–24 (2021), https://doi.org/10.1002/int.22644.
[3] F. Callegati, W. Cerroni, and M. Ramilli, Man-in-the-Middle Attack to the HTTPS Protocol, IEEE Security & Privacy Magazine, vol. 7, no. 1, pp. 78–81, Jan. 2009, https://doi: 10.1109/MSP.2009.12..
[4] D. McGrew and B. Anderson, Enhanced telemetry for encrypted threat analytics, in 2016 IEEE 24th International Conference on Network Protocols (ICNP), Singapore, Nov. 2016, pp. 1–6. doi: 10.1109/ICNP.2016.7785325..
[5] I. Torroledo, L.D. Camacho, A.C. Bahnsen, Hunting Malicious TLS Certificates with Deep Neural Networks, in: Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security, Toronto Canada, 2018, pp. 64–73, https://doi.org/10.1145/3270101.3270105.
[6] Q. Zhang, L. Zhu, R. Wang, et al, Group key agreement protocol among terminals of the intelligent information system for mobile edge computing, International Journal of Intelligent Systems. 1–20 (2021), https://doi.org/10.1002/int.22544.

[7] G. Sun, T. Chen, Y. Su, and C. Li, Internet Traffic Classification Based on Incremental Support Vector Machines, Mobile Networks and Applications, 2018,23(4):789–796. https://doi: 10.1007/s11036-018-0999-x..
[8] Q. Zhang, Y. Li, R. Wang, et al, Data security sharing model based on privacy protection for blockchain-enabled industrial Internet of Things[J], International Journal of Intelligent Systems 36 (1) (2021) 94–111.
[9] B. Anderson and D. McGrew, Machine Learning for Encrypted Malware Traffic Classification: Accounting for Noisy Labels and Non-Stationarity, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax NS Canada, Aug. 2017, pp. 1723–1732. https://doi: 10.1145/3097983.3098163..
[10] N. Zhang, J. Xue, Y. Ma, et al, Hybrid sequence-based Android malware detection using natural language processing[J], International Journal of Intelligent Systems 36 (10) (2021) 5770–5784.
[11] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, PacketCGAN: Exploratory Study of Class Imbalance for Encrypted Traffic Classification Using CGAN, in ICC 2020–2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, Jun. 2020, pp. Conference on Communications (ICC), Dublin, Ireland, Jun. 2020, pp. 1–7. doi: 10.1109/ICC40277.2020.9148946..
[12] H.M. Gomes, Adaptive random forests for evolving data stream classification, Machine Learning, 2017,106(9–10):1469–1495, Oct. 2017, https://doi: 10.1007/s10994-017-5642-8..
[13] Y. Li, S. Yao, R. Zhang, et al, Analyzing host security using D-S evidence theory and multisource information fusion, International Journal of Intelligent Systems. 36 (2) (2020) 1053–1068, https://doi.org/10.1002/int.22330.
[14] Y. Ma, M. Shen, Y. Zhao, et al, Opponent portrait for multiagent reinforcement learning in competitive environment, International Journal of Intelligent Systems. 36 (2021) 7461–7474, https://doi.org/10.1002/int.22594.
[15] Y. Li, X. Wang, Z. Shi, R. Zhang, J. Xue, Z. Wang, Boosting training for PDF malware classifier via active learning, International Journal of Intelligent Systems. 1–19 (2021), https://doi.org/10.1002/int.22451.
[16] Z. Chen et al., Machine learning based mobile malware detection using highly imbalanced network traffic, Information Sciences, 2018, (433–434): 346–364. https://doi: 10.1016/j.ins.2017.04.044..
[17] B. Anderson, D. McGrew, Identifying Encrypted Malware Traffic with Contextual Flow Data, in: Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security - ALSec '16, Vienna, Austria, 2016, pp. 35–46, 10.1145/2996758.2996768.
[18] I. Lee, H. Roh, and W. Lee, Poster Abstract: Encrypted Malware Traffic Detection Using Incremental Learning, in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, Jul. 2020, pp. 1348–1349. https://doi: 10.1109/INFOCOMWKSHPS50562.2020.9162971..
[19] J. Liu, Y. Zeng, J. Shi, Y. Yang, He L. RuiWang, MalDetect: A Structure of Encrypted Malware Traffic Detection, Computers, Materials & Continua 60 (2019) 721–739, https://doi.org/10.32604/cmc.2019.05610.
[20] N.C. Oza, Online Bagging and Boosting, in 2005 IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, HI, USA, 2005, vol. 3, pp. 2340–2345. https://doi: 10.1109/ICSMC.2005.1571498..
[21] O. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00, Boston, Massachusetts, United States, 2000, 71-80. doi: 10.1145/347090.347107..
[22] Stratosphere. 2015. Stratosphere Laboratory Datasets. Retrieved March 13, 2020, from https://www.stratosphereips.org/datasets-overview.
[23] S. Garcia, M. Grill, J. Stiborek, et al, An empirical comparison of botnet detection methods[J], Computers & Security 45 (2014) 100–123.
[24] J. Montiel et al., River: machine learning for streaming data in Python, arXiv:2012.04740 [cs], Dec. 2020, Accessed: May 19, 2021. [Online]. Available: http://arxiv.org/abs/2012.04740..
[25] K. Xing and C. Wu, Detecting Anomalies in Encrypted Traffic via Deep Dictionary Learning, in IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Toronto, ON, Canada, Jul. 2020, pp. 734–739. https://doi: 10.1109/INFOCOMWKSHPS50562.2020.9162940..
[26] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, FS-Net: A Flow Sequence Network For Encrypted Traffic Classification, in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, Paris, France, Apr. 2019, pp. 1171–1179. https://doi: 10.1109/INFOCOM.2019.8737507..
[27] Liu, Z. Tian, R. Zheng, and L. Liu, A Distance-Based Method for Building an Encrypted Malware Traffic Identification Framework, IEEE Access, vol. 7, pp. 100014–100028, 2019, https://doi: 10.1109/ACCESS.2019.2930717..
[28] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, Learning under Concept Drift: A Review, IEEE Transactions on Knowledge and Data Engineering, pp. 1–1, 2018, https://doi: 10.1109/TKDE.2018.2876857..
[29] Jun Zhang, Lei Pan, Qing-Long Han, Chao Chen, Sheng Wen, Yang Xiang, Deep Learning Based Attack Detection for Cyber-Physical System Cybersecurity: A Survey, IEEE/CAA Journal of Automatica Sinica (2021), https://doi.org/10.1109/JAS.2021.1004261.
[30] Y. Miao, C. Chen, L. Pan, Q.-L. Han, J. Zhang, Y. Xiang, Machine Learning Based Cyber Attacks Targeting on Controlled Information: A Survey, ACM Computing Surveys 54 (7) (Sep. 2022) 1–36, https://doi.org/10.1145/3465171.
[31] H. Sun, Y. Tan, L. Zhu, Q. Zhang, Y. Li, S. Wu, A fine-grained and traceable multidomain secure data-sharing model for intelligent terminals in edge-cloud collaboration scenarios, International Journal of Intelligent Systems (2021) 1–20, https://doi.org/10.1002/int.22784.
[32] J. Cheng, J. Zheng, and X. Yu, An ensemble framework for interpretable malicious code detection, International Journal of Intelligent Systems, p. int.22310, Oct. 2020, doi: 10.1002/int.22310..
[33] Junyang Qiu, Jun Zhang, Lei Pan, Wei Luo, Surya Nepal, Yang Xiang, A Survey of Android Malware Detection with Deep Neural Models, ACM Computing Survey 53 (6) (2021) 1–36, https://doi.org/10.1145/3417978.
[34] G. Lin, S. Wen, Q.-L. Han, J. Zhang, Y. Xiang, Software Vulnerability Detection Using Deep Neural Networks: A Survey, Proceedings of the IEEE 108 (10) (Oct. 2020) 1825–1848, https://doi.org/10.1109/JPROC.2020.2993293.