Research Article

Fingerprinting Network Entities Based on Traffic Analysis in High-Speed Network Environment

Xiaodan Gu, Ming Yang, Yiting Zhang, Peilong Pan, and Zhen Ling

School of Computer Science and Engineering, Southeast University, Nanjing, China

Correspondence should be addressed to Ming Yang; yangming2002@seu.edu.cn

Received 24 August 2018; Accepted 28 October 2018; Published 16 December 2018

Guest Editor: Yuan Yuan

For intrusion detection, it is increasingly important to detect the suspicious entities and potential threats. In this paper, we introduce the identification technologies of network entities to detect the potential intruders. However, traditional entities identification technologies based on the MAC address, IP address, or other explicit identifiers can be deactivated if the identifier is hidden or tampered. Meanwhile, the existing fingerprinting technology is also restricted by its limited performance and excessive time lapse. In order to realize entities identification in high-speed network environment, PFQ kernel module and Storm are used for high-speed packet capture and online traffic analysis, respectively. On this basis, a novel device fingerprinting technology based on runtime environment analysis is proposed, which employs logistic regression to implement online identification with a sliding window mechanism, reaching a recognition accuracy of 77.03% over a 60-minute period. In order to realize cross-device user identification, Web access records, domain names in DNS responses, and HTTP User-Agent information are extracted to constitute user behavioral fingerprints for online identification with Multinomial Naive Bayes model. When the minimum effective feature dimension is set to 9, it takes only 5 minutes to reach an accuracy of 79.51%. Performance test results show that the proposed methods can support over 10Gbps traffic capture and online analysis, and the system architecture is justified in practice because of its practicability and extensibility.

1. Introduction

With the rapid development and widespread application of computer networks, mobile communications, smart devices, and the Internet of Things technology, cyberspace is becoming more and more integrated into people's social life. People can access services through various devices anytime and anywhere, thereby realizing the interconnection between people

identifiers are reasonably combined. The potential selections are instantiated as the SSID information in the 802.11 active probe frame, the plug-in installed in the browser, the font library in the system, etc. Although these implicit identifiers cannot uniquely identify a network entity individually, they are hard to be concealed because they usually reflect the user's personalized configuration, historical behavior records, or subtle differences between entities. Therefore, in practical applications, a combination of the implicit identifiers is usually utilized to generate a fingerprint for device identification and user tracking. The fingerprinting technology based on implicit identifiers is essentially a method of traffic analysis and side-channel attack. Although its effectiveness has been initially verified by existing work, there are still many problems to be solved in practical application, including the selection of efficient features, realization of real-time processing of network traffic in a high-speed network environment, and quick identification of devices and users in a short period of time.

In view of the problems mentioned above, we use PFQ kernel module to realize high-speed capture of network packets and use Storm, a well-known distributed real-time streaming data processing technology, to realize online analysis of network traffic. Based on these, a device identification method based on runtime environment analysis and a network user identification method based on behavioral fingerprinting are proposed separately. The main contributions of the paper are as follows:

(i) A distributed traffic analysis framework for high-speed network environments is designed. The framework uses the PFQ kernel module to implement packet capture, Kafka for packet distribution, and Storm for packet content analysis and information extraction of applications, operating systems, HTTP User-Agents, domain names, and Web access records.

(ii) An online device identification technology based on the analysis of user device operating environment is proposed. This technology selects 961 features such as applications, operating systems, and HTTP User-Agent fields to constitute the fingerprints of the devices, while a variety of offline classification models are trained and verified. Finally we select the logistic regression algorithm to identify the user device in a sliding window manner.

(iii) In order to realize network user identification, Web access records, domain names, and HTTP User-Agent fields are selected to constitute user behavioral fingerprints according to the user's network behavior habits. These fingerprints, containing a total of 57593 feature columns, are trained and verified by the two offline classification models using machine learning, which are Naive Bayes and random forest models. By comparison, the Multinomial Naive Bayes model is found to outperform the random forest model, so it is chosen as the classification algorithm for identifying online users in a sliding window manner.

(iv) In order to achieve high-efficiency identification,

intruder occupies an authorized device, we cannot detect the intrusion by using the device identification technologies. So it is necessary to carry out the research on the user identification technology based on behavioral fingerprints.

Essentially, user identification technologies based on behavioral fingerprinting are biometric, which use the inherent physiological characteristics or behavioral characteristics of the human body for identification. They can be categoried into two types. The former one has been widely used by employing the characteristics of human body parts, such as fingerprint identification, face identification, and iris identification.

The behavior-based identification technology [21] extracts the features for identification with the information of the user's operation skills, knowledge, styles, preferences, and strategies revealed in behaviors. For example, researchers have found that different users differentiate from each other in moving, clicking, dragging, and releasing the mouse [22]. Some may be different in key stroking when keyboarding [23]. All of these differences can help to extract fingerprints for effective identification. In the network area, users have different behavioral patterns for network access due to different preferences, habits, etc. Different behavior patterns lead to different traffic flows. Therefore, researchers believe that the network traffic generated by the user can be regarded as biometric for user identification [24].

In order to identify users based on network traffic, early solutions are implemented by extracting explicit identifiers such as IP addresses or MAC addresses [3]. However, such method based on explicit identifiers is not reliable for that it will fail once the user changes the IP address or devices. So far, the dynamic address allocation scheme adopted by ISP makes users change the IP more frequently. To address this issue, the researchers apply the behavior fingerprinting technology to user identification based on network traffic. Padmanabhan et al. [25] find that different users may have different behaviors when browsing the same website. By analyzing the real data, they extract the users' clickprints to generate behavioral fingerprints. Pang et al. [3] propose to explore the destination address, network name, 802.11 option configuration, and broadcast frame length so as to identify the user from the perspective of protocol and user preferences. This is actually a comprehensive application of user-related and device-related implicit identifiers.

Yang [26] uses data mining techniques on the Web browsing dataset in order to mine association rules for each user's behavior, proposes three strength evaluation criteria based on support and lift to generate fingerprints, and finally calculates the distance between fingerprints for identification. Kumpošt et al. [27] believe that the websites visited by the user and the corresponding frequencies, which reflect individual preferences, can be identified as a behavioral fingerprint. They store the source IP, destination IP, and frequency in a two-dimensional matrix and perform the inverse document frequency and cosine similarity algorithm to identify users. Similarly, Herrmann et al. [28] extract user's destination domain names and the corresponding visiting frequency to derive the behavioral fingerprints and use Multinomial Naive Bayes classifier to classify them. Experiments are conducted on a dataset containing HTTP traffic generated by 28 volunteers, and a 73% accuracy rate is obtained.

Since the data set used in the experiments [28] is not big enough to prove the feasibility of the method, the author conducts a larger-scale experiment in the later work [29]. He tests a dataset containing more than 2,100 users' DNS requests, uses the cosine similarity algorithm to filter the noise data, and finally obtains an accuracy of 88%. In addition, Herrmann et al. [30] also compare and evaluate three classification methods through a large number of experiments, including the Multinomial Naive Bayes classifier, the nearest neighbor algorithm, and association rules mining technology. Kim et al. [31] get the user's behavioral fingerprints based on DNS traffic by analyzing the domain name, the sequence of domain names, and the requested periods. Gu et al. [32] infer the users' preferences through the semantic analyses of search records and achieve an accuracy of 93.79% on the dataset of 509 network users.

Overall, the current device and user identification researches have some defects. For example, only a few features are explored and the identification effect is prone to jitter. In addition, the existing studies are not time-efficient enough as it needs to aggregate a whole day's traffic as a fingerprint.

To avoid the aforementioned problems, we adopt the distributed processing technology to extract information such as applications, operating systems, HTTP User-Agent, domain names, and Web access records in real time from high-speed network traffic. Then we propose two online identification methods based on the runtime environment and the behavioral fingerprints, respectively, which are made possible in the pattern of sliding windows. In addition, we also focus on testing the impact of different traffic window sizes on the identification rate and thereby prove the high efficiency and practicality of the proposed technologies.

3. Over ll Design of Fingerprinting

The overall design of our network entities fingerprinting technology is shown in Figure 1. The initial step leverages the PFQ-based high-speed packet capture module to capture high-speed network traffic and then forwards the packets to distributed high-speed network traffic processing modules through the Kafka message queue. Then the processing module parses the message content, extracts the relevant feature data, and stores it in the HBase. Finally, the Spark-based online identification module periodically reads the feature data from the distributed database and realizes the online identification of network devices and users by employing the machine learning algorithms in a sliding window mechanism. The working mechanism and functions of each module are specified as follows:

(i) **PFQ-based high-speed packet capture module.** Configure a mirror port on the switch or router, or use an optical splitter to mirror the traffic to a data distribution server. The high-speed packet capture module on this server achieves highly efficient packet capture by adopting the memory mapping mechanism, zero

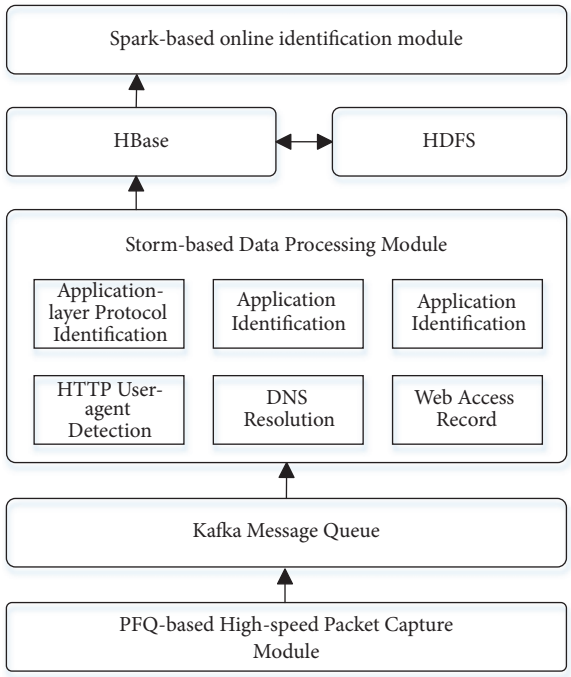


Figure 1: Overall design of network entities identification.

copy technology, and double-buffering mechanism based on the PFQ Linux kernel module.

(ii) **Kafka message queue.** The distributed message queue is a data transmission channel between the packet capture module and the distributed processing module. More specifically, it is a buffer zone for coordinating the producer and consumer. We use Kafka to implement distributed message publish, which has a high linear extensibility in adapting to the high-speed data transmission scenario.

(iii) **Storm-based data processing module.** The distributed data processing module, as a core functional module, undertakes all processing and analysis tasks for network traffic, including parsing of network messages, identification of application protocols, identification of applications, and finally extracts and stores data of applications, operating systems, Web access records, domain names, and HTTP User-Agent fields. We realize distributed streaming data processing based on the Storm platform, which can achieve high-speed data reading combined with Kafka queues and can achieve high-speed data writing combined with HBase. Meanwhile, data transmission performance between the internal components of Storm is also very efficient.

(iv) **HBase.** The online identification module is proposed to read and analyze the data periodically. Thus, as a distributed column-oriented database, the HBase functions as a data medium between the distributed data processing module and the online identification module.

(v) **Spark-based online identification module.** Our identification module is based on the Spark platform and designed to fit into two scenarios. For the device identification scenario, the module extracts device features about runtime environment to generate the fingerprints. For the cross-device identification scenario, the user behavior data are collected to implement fingerprinting for network users. The relevant feature data are read from the HBase distributed database, and the machine learning algorithms in Spark MLlib along with the sliding window mechanism are employed to identify the network devices and users online.

4. Collection nd Distributed Processing of High-Speed Network Tr ffic

4.1. Collection of High-Speed Network Traffic. Compared to Pcap, which is a traditional packet sniffing toolkit, PFQ is a better designed network packet capture framework customized for the optimization of multicore CPUs and multihardware queue network interfaces. It is primarily used for efficient packet capture and transmission on Linux. In its internal implementation, PFQ eliminates the cost of copying packets from kernel space to user space by adopting a memory mapping mechanism, and performs concurrency operations of user-space applications and PFQ kernel packet grabbing programs on the buffers by means of double-buffering technology. The core components of PFQ fall into three parts: packet extracting program, packet forwarding module, and socket queue. First, the packet extraction program directly obtains the packets from the network interface card (NIC) driver and transfers them to the batch queue. Then, the packet forwarding module selects the socket and sends packets to the user-space applications.

After the packets are captured, librdkafka is used to write them into the Kafka message queue. In this paper, we decouple the high-speed packet capture module and the Kafka message queue through the open source project Blockmon [33].

4.2. Distributed Processing of Network Traffic. When the captured packets are written into the Kafka message queue, we implement distributed analysis and processing of packets based on the Storm platform. The following are the key concepts related to the Storm:

(i) Topologies: the logic of the application which defines various components and the ways of communication between them.

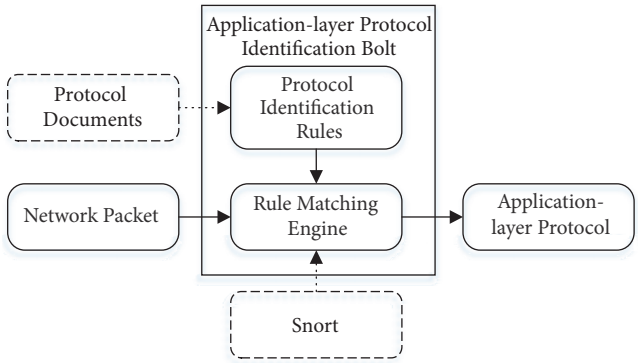


Figure 2: Application layer protocol identification bolt.

transmitting the processing results to the external system for storage or display.

4.2.1. Input of Flow Data. Data transmission between Spouts and Bolts, as well as that among Bolts, is in the form of Streams, while Spouts obtain data from external sources in different ways. In this paper, we use KafkaSpout to read packets from the Kafka message queue and transmit tuples to the packets parsing Bolts.

In the distributed processing environment, KafkaSpouts retrieve data from Kafka partitions in parallel on all slave nodes. And the degree of parallelism has a crucial influence on the throughput of the system. Meanwhile, it is affected by the number of Kafka partitions, because each Kafka partition can only be consumed by one KafkaSpout. Namely, the key to improving Storm throughput is to increase the number of Kafka partitions.

4.2.2. Packets Analyzing and Filtering. The inputting packet tuples are analyzed and filtered to obtain the content of messages, and the header information in each layer of the network protocol is extracted, including PFQ pkthdr header, Ethernet frame header, IP header, TCP header, and UDP header. In the process of packet analysis, several rules are set to filter packets unrelated to network device identification, such as the network control protocol packets and routing protocol packets to improve the processing efficiency of the system.

4.2.3. Application Protocol Identification. Traditionally, in order to identify application protocols, the port numbers of the captured packets are always matched with the well-known ones. Such method is deficient due to a high false positive rate. Therefore, we aim to improve the identification accuracy by employing DPI technology to analyze the packet payloads. Although such method is less efficient, its accuracy rate is significantly higher than that of the former. The module of application protocol identification is developed in two phases: (a) designing the rule matching engine and (b) writing protocol identification rules. The first step extracts the rule matching engine of the Snort core components and introduces multithreading. Then, based on referencing protocol documents, we summarize the characteristics of a typical protocol and write an appropriate rule according to the writing specification of Snort rules.

```
alert udp $EXTERNAL_NET any -  $HOME_NET any
(msg: "snmp"; content: "  "; offset:0; depth:1;
byte_test:1,  ,0x80,1; content: "  "; offset:2; depth:1;
sid:70; rev:1;)
```

Box 1

The process of identifying application protocols is shown in Figure 2. The rule matching engine generates a rule tree according to the specified protocol identification rules and performs rule matching for network traffic. When a match occurs, it indicates that the packet is identified as a specific type of protocol.

Box 1 shows an identification rule for the SNMP protocol and corresponding explanation ensues.

This statement indicates that the rule is released as the first version, with the id of 70. All the UDP packets sent from any port or IP address are detected without exception. According to the identification rule, the first byte value of the application layer payload is 0x30. The second byte value must be less than 0x80, and the third is 0x02. When the match is successful, the alert action is triggered and the protocol type value SNMP is returned to the caller.

After a review of various typical protocol documents, we have completed the writing of recognition rules for 25 typical application layer protocols such as BITTORRENT, DNS, DROPBOX, HTTP, SMTP, and SSH.

4.2.4. Application Identification. After the identification of the application protocol, we further figure out the type of application that generates the packets. As we all know, apart from the traffic generated by the user interaction, the background process of the application communicates with the server periodically, thereby generating more traffic. We analyze the traffic and extract various features to identify the applications.

The data transmission between an application and the server is generally divided into two cases. First, the application uses a custom data transmission protocol, such as the OICQ protocol, which is designed by Tencent and is merely used as the data transmission protocol of QQ. In this case, as long as the application protocol has been identified, the application is sure to be identified. Second, multiple kinds of applications share an application layer data transmission protocol, such as the HTTP protocol, to encapsulate data transmitted between the client and the server. For this situation, we distinguish different applications by extracting multiple field values from the traffic. For example, in the HTTP protocol, the HOST field represents the combination of the domain name and the port number of the server address. Usually, the addresses and corresponding domain names of applications devised by different companies are not identical. Even though different applications provided by the same company share the same server, the addresses are still distinguished from each other with different HTTP request parameters. Therefore, the combination of the HOST field

of HTTP protocol and request parameters can be used to identify different applications.

This paper observes and analyzes the application traffic in the experimental network and summarizes a collection of 116 commonly used application identification rules under 21 categories, such as browser, e-mail, remote management, online game, instant messaging, social networking, Web disk, input tool, online video, P2P video, and stock software. These identification rules cover traffic generated from users' clicks, login activities, automatic updates, and background process communications.

4.2.5. HTTP User-Agent Detection. As the name suggests, the User-Agent field in HTTP traffic contains the user agent information. Generally, the User-Agent field generated by a browser contains the information like the types and versions of the browser and the operating system. As an important piece of information in the User-Agent field, a specific operating system has its own structure-mapping rules, diversifying the types of all the existing operating systems. For instance, the prefixes of the Windows operating systems are normally Windows NT, and suffixes represent specific operating system versions. The identification rules of the operating systems presented in this paper cover the mainstream operating systems such as Windows, Mac OS, OpenBSD, and Ubuntu.

4.2.6. DNS Resolution and Web Access Records.

device fingerprint, where the Boolean type device fingerprint indicates whether features such as applications or operating systems appear in the network traffic, and the numeric device fingerprint indicates how often these features appear.

Note that all the identifiable application type sets are $\mathcal{S} \quad \mathcal{S}_1\ \mathcal{S}_2 \cdots\ \mathcal{S}_N$ and the ith application's version set is $V_i \quad V_i^1\ V_i^2 \cdots\ V_i^{C_i}$. C_i refers to the total number of recognizable versions of the ith application and OS represents the operating system type. The feature vector of the device fingerprint can be presented by formula (1).

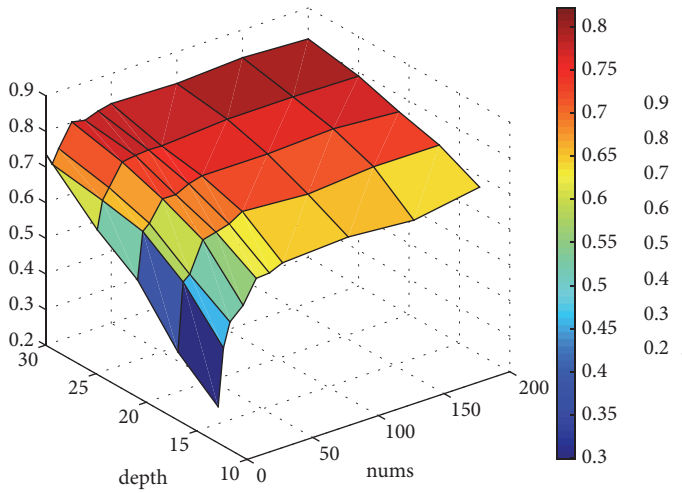
$$\overrightarrow{FP_{dev}} \quad \left\{ \underbrace{\mathcal{S}_1\ \mathcal{S}_1V_1^1 \cdots\ \mathcal{S}_1V_1^{C_1}}_{S_1}\ \underbrace{\mathcal{S}_2\ \mathcal{S}_2V_2^1 \cdots\ \mathcal{S}_2V_2^{C_2}}_{S_2} \cdots \right. \underbrace{\mathcal{S}_N\ \mathcal{S}_NV_N^1 \cdots\ \mathcal{S}_NV_N^{C_N}}_{S_N}\ O\mathcal{S} \left. \right\} \tag{1}$$

When the value of the attribute in fingerprint $\overrightarrow{FP_{dev}}$ is numerical, it is a numerical type device fingerprint. When the value of the attribute is only 0 or 1, it is a Boolean type one. Then the device identification problem can be modeled as a multiclassification problem in machine learning.

5.3. Offline Model Training and Verification. Since the efficiency of identification depends greatly on the classification algorithm and the dimension of the device fingerprint vector is relatively small, the multiclassification algorithm can generally be used to train the identification model. We compare the classification effects of Multinomial Naive Bayes algorithm, random forest algorithm, and the logistic regression algorithm. After that, the best performed algorithm is selected for online identification.

We collect network traffic of 118 user devices for 53 days from June 1st to July 23rd, 2016. The network traffic produced on each device is examined per hour. Based on the examination, the features are extracted to form a fingerprint (all zero-vector fingerprints are discarded). Then we get 50,305 valid device fingerprints in total. The data collected in the first 30 days is used to train and verify the offline model, including 30,148 records, while the remaining 20,157 records gathered in the following days are used to evaluate the accuracy of the classification model.

During the offline training process, the device fingerprints in the data set are randomly divided into two subsets, one being the training set containing 70% fingerprints and the other being the verification set conon%□(%)-□□□.□□(□.□□()-□□□.□□(pr)□.□□o□□□(g)-□-□□□.□(e)□.□□(a)□□□.□□()-m□□.□□(□.□(h)-m□(th□(od)□



(a) Boolean type fingerprints

0.8
0.75
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3

Table 1: The effect of the minimum effective feature dimension on the number of device fingerprints.

Minimum Effective Feature Dimension	Validation Set	Test Set
1	100.00%	100.00%
2	89.58%	90.43%
3	82.98%	81.66%
4	75.94%	74.85%
5	70.57%	68.99%
6	63.73%	62.08%

climbs. The classification accuracy values of three models all plateau close to 80% for the test set when the minimal effective dimension is 6. However, despite the top accuracy, only 62.08% of device fingerprints in the test set are retained. In comparison, when the minimal effective dimension is lowered to 4, the classification accuracy of Multinomial Naive Bayes and logistic regression for the test set is higher than 75%, and 74.85% of device fingerprints in the test set are retained. Considering the effect of the minimal effective dimension on fingerprint classification accuracy and traffic coverage, the minimal effective dimension 4 is determined as the threshold for filtering fingerprint data. Since the logistic regression model performs comparatively better for both the validation set and the test set, the logistic regression model is chosen as the online identification model for the Boolean type device fingerprinting.

5.3.2. Training and Verification of Classification Model for Numerical Type Device Fingerprinting. For numerical type device fingerprinting, the same random forest parameters are first trained. The results are shown in Figure 4(b). When *nums* is 100 and depth is 30, the random forest model has the best classification effect. Since the specific value of each feature has an important influence on the classification result of the Multinomial Naive Bayesian classification model, we need to perform the term frequency (TF) transform as shown in formula (2) for each feature value.

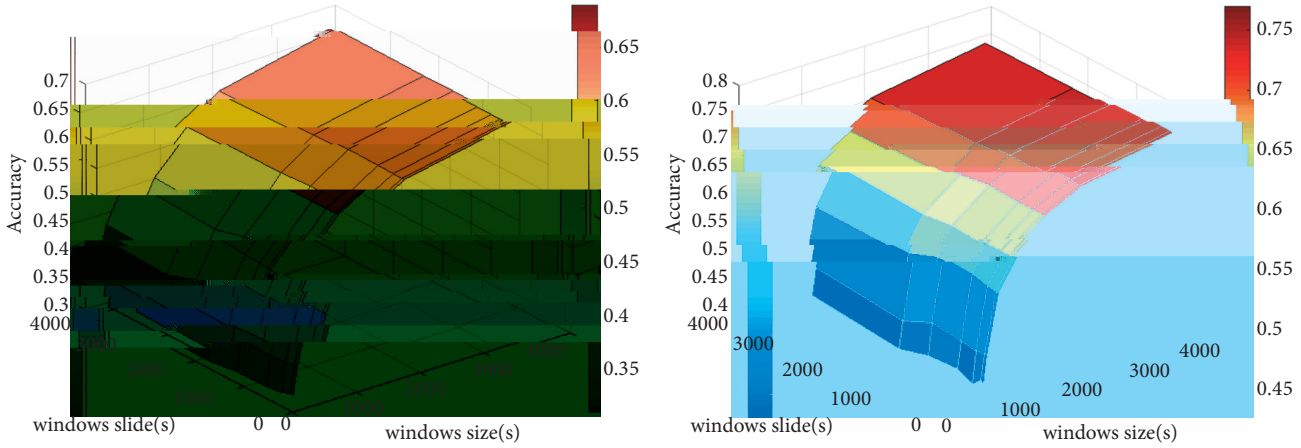
$$\log \qquad (2)$$

To further implement numerical type device fingerprinting, we train the Multinomial Naive Bayes classification model and the logistic regression classification model, respectively, and calculate the classification accuracy on the verification set and the test set. The results are shown in Figure 5(b). We also verify the impact of the minimum effective dimension on the classification accuracy, as shown in Figure 6(b). By comparing Figures 5(a) and 5(b), Figure 6(a) and Figure 6(b), respectively, we can find that the performances of Boolean and numerical type device fingerprinting are basically the same and can both achieve a comparatively high device identification accuracy. However, because the numerical type fingerprints may fluctuate on feature values, we only leverage the Boolean type device fingerprinting to test the online identification accuracy of devices.

5.4. Online Identification of User Devices. The online identification of user devices employs the Boolean type device fingerprinting, and a logistic regression model is taken as the classificatc vstm)-. (e)- .xtyperline

the
the . bwot. r . [the . wid (o) . (e - . s)e

thv e w rlne
. - - c ee
ey e



(a) Setting the minimum effective feature dimension to 1

(b) Setting the minimum effective feature dimension to 4

Figure 7: The Effect of sliding window on the accuracy of online device identification.

the information about the types of browsers, versions, and operating system types achieved through HTTP User-Agent detection, to generate user's online behavioral fingerprints for the identification of network users.

The behavioral fingerprint vector is generated by extracting features from captured traffic in a unit time. For the target IP address in the Web access record, we associate it with the domain name based on the DNS response records and treat all IP addresses and subdomains under the same domain name as one attribute of the vector.

After the features of domain names are determined, combined with the information contained in the HTTP User-Agent field, a network user's behavioral fingerprint is generated, which constitutes the fingerprint vector of the user behavior in a unit time.

The dimension of the user behavioral fingerprint vector is 57,593. According to the value type of feature attributes, behavioral fingerprints can also be divided into two types: boolean and numerical type. However, we can see from the fingerprint classification results of the device that there is no obvious difference between the classification accuracy of the two types of fingerprints, and the classification accuracy of Boolean type device fingerprinting is slightly better than that of numerical type fingerprinting. Therefore, we will test the identification accuracy of network user with Boolean type behavioral fingerprinting.

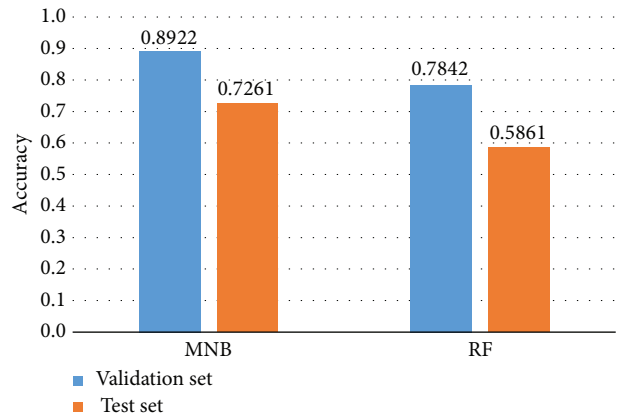
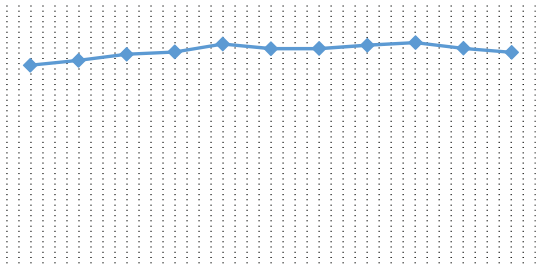
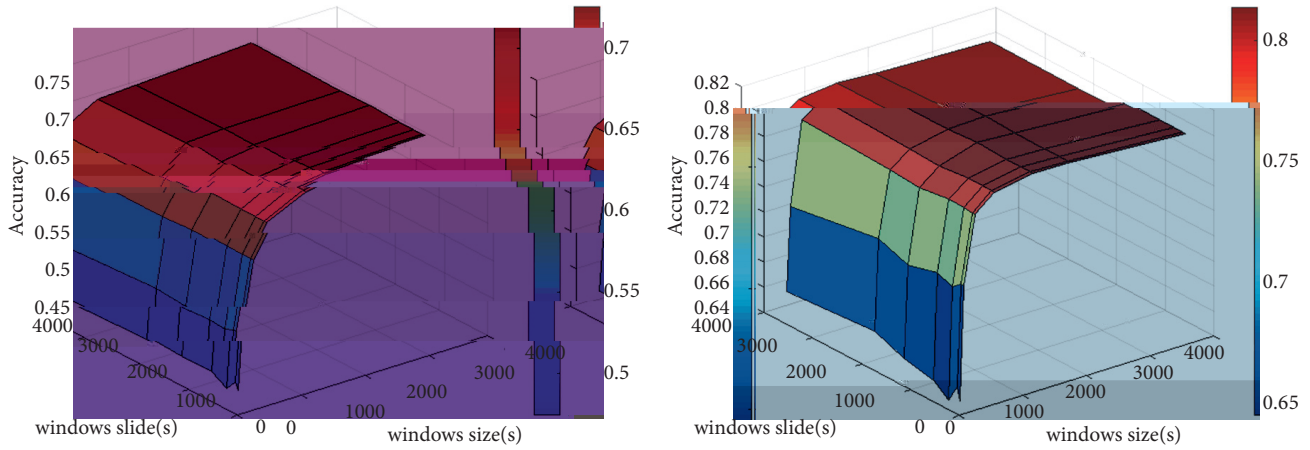


Figure 9: Behavioral fingerprints classification accuracy using MNB and RF.





(a) Setting the minimum effective feature dimension to 1

(b) Setting the minimum effective feature dimension to 9

Figure 11: The effect of sliding window on the accuracy of online user identification.

Table 3: The results of high-speed packets capture (pps).

Packet length (Byte)	Number of physical cores				
	1	2	4	8	12
100	1329465	1566371	3206383	5936376	6272835
200	1130613	1734582	2917599	5352853	5586074
500	922524	1155344	2384589	2385555	2402127
1000	852867	1219714	1219109	1230260	1229943

is 80.74%. Therefore, the time window size of online user identification can be further shortened to 5 minutes.

7. Perform nce Tests nd Results

7.1. Test Environment. In the above we have evaluated and proved the effectiveness of device identification and user identification with different algorithms and parameters, respectively. This section mainly tests the performance of the identification methods. The test environment is as follows:

7.1.1. Hardware

(i) 1 master node: Dell PowerEdge R730 (CPU: 2 6-core E5-2620V2, 2.1GHz, memory: 96 GB, external storage: 3.6TB).

Table 4: The results of high-speed packets capture (Gbps).

Packet length (Byte)	Number of physical cores				
	1	2	4	8	12
100	1.06	1.25	2.57	4.75	5.02
200	1.81	2.78	4.67	8.56	8.94
500	3.69	4.62	9.54	9.54	9.61
1000	6.82	9.76	9.75	9.84	9.84

Table 5: The speed test results of distributed processing framework.

The Number of Kafka Partitions	Maximum Processing Speed
1	3.76Gbps
2	6.68Gbps
3	10.01Gbps
4	13.35Gbps

Table 6: The time consuming of online identification.

#	Time consuming of online identification (ms)
1	9074
2	8256
3	7480
4	8188
5	6566
6	6780
7	6643
8	

of 5.02Gbps. When the length is changed to 200 bytes, the maximum capture rate is 8.94Gbps. When the packet length is 500 bytes, the maximum packet capture rate is 9.61Gbps.

The experimental results above show that the use of Linux PFQ kernel module can capture packets with a high speed and has robust systematic extensibility.

7.2.2. Speed of Distributed Processing of Packets. When testing the speed of the distributed processing framework, this paper uses KafkaProducer to write the network traffic captured by the packet capture module into each Kafka partition and then calculate the framework's processing speed of reading and analyzing data from Kafka partitions.

The number of KafkaSpouts is consistent with the number of Kafka partitions, which has a crucial influence on the speed of the distributed processing framework. Table 5 shows the speed of the distributed processing framework under different Kafka partition numbers. As we can see, the maximum processing speed is basically proportional to the number of Kafka partitions. It is noteworthy that it exceeds 10Gbps when the Kafka partition number is 3.

7.2.3. Response Speed of Online Application Identification. This paper uses the maximum window size of 60 minutes to test the response speed of the online identification module. This module contains two parts: online device identification based on the runtime environment and online user identification based on network behavioral fingerprints. Through the statistics of the time consumption of online identification modules for 10 trials, the average value is calculated as the response speed of the online identification module. The time consumption of the 10 online identifications is collected in Table 6. By averaging them, the response speed is derived as 7362 ms. This value is much smaller than the minimum step size of the recognition window (1 minute), so it can be considered that the online identification processing speed can meet the performance requirement.

D t Av il bility

The network traffic data used to support the findings of this study have not been made available because they contain a lot of privacy information.

Disclosure

Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by National Key R&D Program of China 2018YFB0803400 and 2017YFB1003000, National Natural Science Foundation of China under grants 61572130, 61502100, 61532013, and 61632008, by Jiangsu Provincial Scientific and Technological Achievements Transfer Fund BA2016052, by Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, by Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 93K-9, and by Collaborative Innovation Center of Novel Software Technology and Industrialization.

References

[1] G. Pedro T, D. Jes·sE V, M. Gabriel F, and V. Enrique, "Anomaly-based network intrusion detection: Techniques, systems and challenges," in *Computers Security*, pp. 18–28, 18–28, 28(1–2, 2009.

[2] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network anomaly detection: methods, systems and tools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[3] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall, "802.11 User fingerprinting," in *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*, pp. 99–110, ACM, September 2007.

[4] Y. Zhang, M. Yang, X. Gu, P. Pan, and Z. Ling, *Proceedings of the 2018 International Conference on Advanced Cloud and Big Data*, LanZhou, China, 2018.

[5] B. Danev, D. Zanetti, and S. Capkun, "On physical-layer identification of wireless devices," *ACM Computing Surveys*, vol. 45, no. 1, article 6, 2012.

[6] T. Kohno, A. Broido, and C. Claffy K, "Remote physical device fingerprinting," in *Proceedings of the 26th IEEE Symposium on Security and Privacy (SP'05)*, Berkeley, CA, USA, 2005.

[7] R. Gerdes, T. Daniels, M. Mina, and S. Russell, "Device Identification via Analog Signal Fingerprinting: A Matched Filter Approach," in *Proceedings of the 13th Annual Network and Distributed System Security Symposium Conference (NDSS'06)*, San Diego, CA, USA, 2006.

[8] E. D. Thomas, J. A. Van Randwyk, E. J. Lee et al., "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proceedings of the 15th conference on USENIX Security Symposium*, Vancouver, B.C., Canada.

[9] T. Yen, Y. Xie, F. Yu, R. Yu, and M. Abadi, "Host Fingerprinting and Tracking on the Web: Privacy and Security Implications," in *Proceedings of the 19th Annual Network and Distributed System Security Symposium (NDSS'12)*, 2012.

[10] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, "Who do you sync you are?" in *Proceedings of the the sixth ACM conference*, p. 7, Budapest, Hungary, April 2013.

[11] P. Eckersley, "How unique is your web browser?" in *Privacy Enhancing Technologies: 10th International Symposium, PETS 2010, Berlin, Germany, July 21–23, 2010. Proceedings*, vol. 6205 of *Lecture Notes in Computer Science*, pp. 1–18, Springer, Berlin, Germany, 2010.

[12] K. Mowery, D. Bogenreif, S. Yilek, and H. Shacham, "Fingerprinting information in JavaScript implementations," in *Proceedings of 2011 Web 2.0 Security and Privacy (W2SP'11*, Oakland, California, 2011.

[13] J. Mayer and J. Mitchell, "Third-party web tracking: Policy and technology," in *Proceedings of the 33rd IEEE Symposium on Security and Privacy (SP'12)*, San Francisco, CA, USA, 2012.

[14] G. Acar, M. Juarez, N. Nikiforakis et al., "FPDetective: Dusting the web for fingerprinters," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013*, pp. 1129–1140, Germany, November 2013.

[15] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "Cookieless monster: Exploring the ecosystem of web-based device fingerprinting," in *Proceedings of the 34th IEEE Symposium on Security and Privacy, SP 2013*, pp. 541–555, USA, May 2013.

[16] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "On the workings and current practices of web-based device fingerprinting," *IEEE Security & Privacy*, vol. 12, no. 3, pp. 28–36, 2014.

[17] P. Laperdrix, W. Rudametkin, and B. Baudry, "Beauty and the Beast: Diverting Modern Web Browsers to Build Unique Browser Fingerprints," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 878–894, USA, May 2016.

[18] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting Mobile Devices Using Personalized Configurations," *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.

[19] W. Wu, J. Wu, Y. Wang, Z. Ling, and M. Yang, "Efficient Fingerprinting-Based Android Device Identification with Zero-Permission Identifiers," *IEEE Access*, vol. 4, pp. 8073–8083, 2016.

[20] Y. Cao, S. Li, and E. Wijmans, "(cross-)browser fingerprinting via os and hardware level features," in *Proceedings of the 24th Annual Network and Distributed System Security Symposium, NDSS*, San Diego, CA.

[21] R. V. Yampolskiy and V. Govindaraju, "Behavioural biometrics: a survey and classification," *International Journal of Biometrics*, vol. 1, no. 1, pp. 81–113, 2008.

[22] N. Zheng, A. Paloski, and H. Wang, "An efficient user verification system via mouse movements," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pp. 139–150, ACM, October 2011.

[23] S. Douhou and J. R. Magnus, "The reliability of user authentication through keystroke dynamics," *Statistica Neerlandica. Journal of the Netherlands Society for Statistics and Operations Research*, vol. 63, no. 4, pp. 432–449, 2009.

[24] N. V. Verde, G. Ateniese, E. Gabrielli, L. V. Mancini, and A. Spognardi, "No NAT'd user left behind: Fingerprinting users behind NAT from NetFlow records alone," in *Proceedings of the 2014 IEEE 34th International Conference on Distributed Computing Systems, ICDCS 2014*, pp. 218–227, Spain, July 2014.

[25] B. Padmanabhan and Y. Yang, *Clickprints on the web: Are there signatures in web browsing data*, 2006, http://knowledge.wharton.upenn.edu/papers/1323.pdf.

[26] Y. Yang, "Web user behavioral profiling for user identification," *Decision Support Systems*, vol. 49, no. 3, pp. 261–271, 2010.

[27] M. Kumpošt and V. Matyáš, "User Profiling and Re-identification: Case of University-Wide Network Analysis," in *Trust, Privacy and Security in Digital Business*, vol. 5695 of *Lecture Notes in Computer Science*, pp. 1–10, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[28] D. Herrmann, C. Gerber, C. Banse, and H. Federrath, "Analyzing Characteristic Host Access Patterns for Re-identification of Web User Sessions," in *Information Security Technology for Applications*, vol. 7127 of *Lecture Notes in Computer Science*, pp. 136–154, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[29] C. Banse, D. Herrmann, and H. Federrath, "Tracking Users on the Internet with Behavioral Patterns: Evaluation of Its Practical Feasibility," in *Information Security and Privacy Research*, vol. 376 of *IFIP Advances in Information and Communication Technology*, pp. 235–248, Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

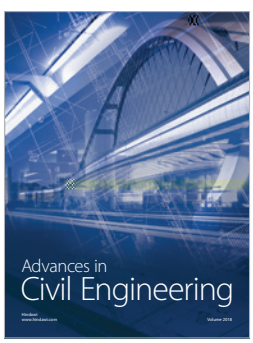
[30] D. Herrmann, C. Banse, and H. Federrath, "Behavior-based tracking: Exploiting characteristic patterns in DNS traffic," *Computers & Security*, vol. 39, pp. 17–33, 2013.

[31] D. W. Kim and J. Zhang, "You Are How You Query: Deriving Behavioral Fingerprints from DNS Traffic," in *Security and Privacy in Communication Networks*, vol. 164 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pp. 348–366, Springer International Publishing, Cham, 2015.

[32] X. Gu, M. Yang, C. Shi, Z. Ling, and J. Luo, "A novel attack to track users based on behavior patterns," *Concurrency and Computation: Practice and Experience*, 2016.

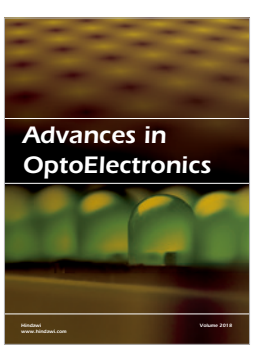
[33] Blockmon, "cnplab/blockmon," https://github.com/cnplab/blockmon.

[34] Tcpreplay, "Tcpreplay development is now being done by AppNeta," URL http, http://tcpreplay.synfin.net.


Advances in
Civil Engineering


Journal of
Robotics

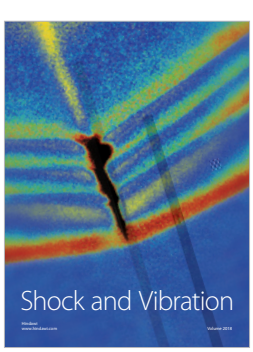

Journal of
Electrical and Computer
Engineering

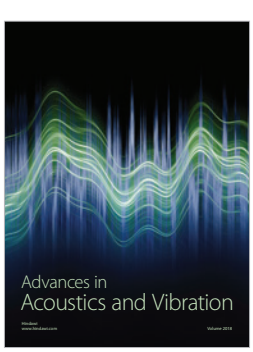

Advances in
OptoElectronics


VLSI De


International Journal of
Aerospace
Engineering


Active and Passive
Electronic Components


Shock and Vibration


Advances in
Acoustics and Vibration