electronics

MDPI

Article

Efficient Fingerprinting Attack on Web Applications: An Adaptive Symbolization Approach

Xue Yang [1], Jian Xu [2] and Guojun Li [3,*]

[1] Computer and Information Security Department, Zhejiang Police College, Hangzhou 310053, China
[2] Computer Application Technology Department, School of Computer Science, Hangzhou Dianzi University, Hangzhou 310018, China
[3] Basic Courses Department, Zhejiang Police College, Hangzhou 310053, China
* Correspondence: lgj_zjjcxy@163.com

Abstract: Website fingerprinting is valuable for many security solutions as it provides insights into applications that are active on the network. Unfortunately, the existing techniques primarily focus on fingerprinting individual webpages instead of webpage transitions. However, it is a common scenario for users to follow hyperlinks to carry out their actions. In this paper, an adaptive symbolization method based on packet distribution information is proposed to represent network traffic. The Profile Hidden Markov Model (PHMM exploits positional information contained in network traffic sequences and is sensitive to webpage transitional information) is used to construct users' action patterns. We also construct user role models to represent different kinds of users and apply them to our web application identification framework to uncover more information. The experimental results demonstrate that compared to the equal interval and K-means symbolization algorithms, the adaptive symbolization method retains the maximum amount of information and is less time-consuming. The PHMM-based user action identification method has higher accuracy than the existing traditional classifiers do.

Keywords: network traffic; adaptive symbolization; PHMM; user action patterns; web application identification

Citation: Yang, X.; Xu, J.; Li, G. Efficient Fingerprinting Attack on Web Applications: An Adaptive Symbolization Approach. *Electronics* **2023**, *12*, 2948. https://doi.org/10.3390/electronics12132948

Academic Editors: Manuel Palomo-Duarte, Marko Horvat, Igor Mekterović and Juan Antonio Caballero-Hernandez

Received: 21 April 2023
Revised: 18 June 2023
Accepted: 23 June 2023
Published: 4 July 2023

1. Introduction

Being able to automatically associate a portion of network traffic to a particular web application is desired by network administrators or attackers. With the growth in the usage of end-to-end encryption protocols (such as SSL/TLS), attackers can not inspect the content of communications. However, traditional encryption obscures only the content but does not hide information such as the traffic volume and direction. This allows an attacker to exploit information leaked by the side channel, such as the packet length, timing, and order.

Recent studies have proposed a number of potential solutions to analyze encrypted traffic. A proposed framework [1] monitors network traffic between users and network resources to identify the associated web application. Many machine learning algorithms (e.g., random forest) and deep learning methods, such as convolutional neural networks (CNNs), are used to uncover what applications are running on users' smartphones [2–4] or what webpages/websites users are visiting [5–7]. Among them, the technology that identifies webpages/websites from network traffic is referred to as Website Fingerprinting Attack (WFA).

Despite many WFA methods having been proposed, previous studies primarily focused on fingerprinting individual webpages (most existing methods simply refer to homepages as representative webpages) to identify whether users have accessed a monitored website. They usually ignore sequence visits, such as webpage transitions via clicking hyperlinks. However, for most websites, users often follow hyperlinks to carry out their actions. For example, users follow hyperlinks to read/post blogs on a social forum. In this

paper, we take webpage transitional information into consideration when we profile the "user-web application" interaction (i.e., build users' action patterns). These patterns can be applied to determine whether a sequence of network traffic belongs to a monitored web application (static websites are not considered in our work).

Identifying a web application via interaction patterns is practically significant. It is not a difficult job to create a web application today since there are many ready-made templates to choose from. A website builder named Wix [8] provides different types of templates (ranging from e-business, and album, to social forum ones, and so on) and publicizes that customers can create a website in just four steps without any coding skills. Reports [9] surfaced that a police officer provided a source code seized during the investigation of a case to other criminals to create a new gambling website and illegally obtain huge profits. In reality, even if a gambling website is targeted by law enforcement officers, criminals may modify its appearance (e.g., the website title and pictures) and rebuild a new one easily. In order to block these slightly modified illegal websites, an approach that can detect a template-based web application is needed.

Intuitively, web applications that derive from the same template may share similar functional logic. Even if some elements, such as the titles or pictures, change due to environmental factors, we argue that some key elements remain. These unchanged elements can be seen as the functionality genes of those web applications. A web application is often designed to provide users with different capabilities, i.e., users can perform certain actions. Criminals might modify the appearance of an illegal website to avoid punishment, but they cannot change those capabilities. Based on intuition, we can identify the modified web application by identifying associated users' action patterns from the network traffic, which is an advantage in today's model-driven engineering applications [10].

Since users usually click from one webpage to another by following a hyperlink, the generated packet sequences are often very long and contain transitional information. Previous work has demonstrated that the Profile Hidden Markov Model (PHMM, a widely used tool in bioinformatics for DNA sequencing analysis [11]) exploits positional information contained in network traffic sequences and is sensitive to webpage transitional information [12]. Thus, we used the PHMM to construct users' action patterns in this work. In order to utilize the PHMM, one is left with the practical problem of reducing the complexity of modeling the original sequences while retaining the maximum amount of information. In order to achieve this, a novel symbolization method is proposed in this work. A symbolization approach involves partitioning the range of the original sequence into a finite number of regions. Each region is associated with a specific symbolic value, and each original packet is uniquely mapped to a particular symbol depending on the region in which the packet falls.

In previous work, packet sequences were divided into equal-sized intervals [12] or grouped into clusters [13] and then labeled with corresponding symbols. However, the equal-sized interval method ignores the packet distribution information of the original sequences. For example, Figure 1 depicts the packet length distribution of users' scanning of a social forum. It is easy to find out that a few packets fall in regions [0, 600] and [900, 1500]. If the corresponding traffic is symbolized using equal-sized intervals (e.g., 300 bytes), some symbols in the symbol set are not often used. Leveraging the clustering algorithms to symbolize traffic sequences is another straightforward method. However, it is very time-consuming to cluster a huge amount of data (which is very common in network traffic analysis). In order to solve these problems, we propose an adaptive symbolization method, which takes packet distribution into consideration and guarantees efficiency via mathematical proof. Adaptive models are regarded as a potential application in many research areas. For example, in software testing, adaptive models can integrate the study of requirements, building and executing cases [14].

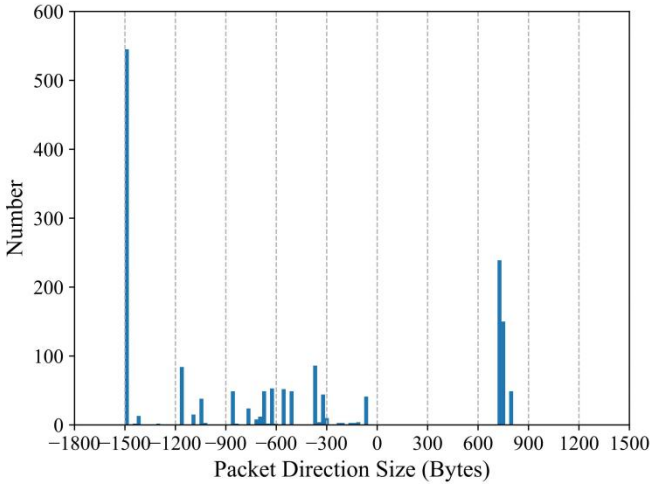

Figure 1. Equal-sized interval method ignores packet distribution information.

In this paper, we describe an approach to construct users' action patterns and use them to identify users' actions from network traffic. An adaptive symbolization method is proposed to represent network traffic, and a profile-hidden Markov model is utilized to build the user action patterns. We then apply the proposed user action identification method to identify whether a sequence of network traffic belongs to a monitored web application.

The novelty of this work includes: (1) The proposed adaptive symbolization method takes packet distribution information into consideration, and its efficiency outperforms existing symbolization approaches and can be supported by mathematical proof. (2) User role models are first considered in web application identification, which reveals more detailed information, e.g., the type of users that carry out a certain action.

The major contributions of our paper are summarized as follows:

- A novel adaptive symbolization method is proposed to reduce the complexity of modeling the original traffic sequences while retaining information to the maximum;
- The PHMM is used to model and identify user actions that hide in network traffic;
- A framework is proposed to identify monitored web applications in which user role models are proposed and utilized to mine more information from network traffic;
- A website is established, and experiments are conducted to demonstrate the effectiveness and efficiency of our proposed approach.

The remainder of this paper is organized as follows. We discuss the state of the art around this research topic in Section 2. The definitions and identification problems are proposed and formulated in Section 3. In Section 4, a method that constructs users' action patterns is presented, and all its different steps are described in detail. In Section 5, the adaptive symbolization method is proposed. In Section 6, we introduce how our approach can be applied to web application identification. In Section 7, we describe the experimental experience of our approach and evaluate the experimental results. Section 8 is about the discussion and limitations of this study. The paper is concluded in Section 9.

2. Related Work

2.1. Website Fingerprinting Attack

The purpose of a Website Fingerprinting Attack (WFA) is to infer which websites/webpages are visited by users. This type of analysis can reveal the privacy of a user (e.g., interests, habits, sexual and political orientations). WFA was first carried out by Cheng and Avnur [15] in 1998. They demonstrated that the SSL protocol can not address traffic analysis attacks. WFA turns to be a hot research topic in recent years, and many machine-learning techniques have been proven to be very effective.

A work published in 2012 [16] was the first demonstration that application-level defenses, such as HTTPOS and randomized pipe-lining, are not secure. The authors modeled websites using Hidden Markov Models (HMMs), where each state corresponds to a page or a class of pages of the site. To simplify the model, they created it with states corresponding to page templates rather than individual pages. According to their approach, an attacker can construct a HMM for each target website and use the forward algorithm to compute the log-likelihood that a given packet trace would be generated by a user visiting the target website. However, it is not a trivial thing to build a HMM model for a website.

Hayes and Danezis [17] did a systematic analysis of feature importance and filled the gap of a notable absence of feature analysis in the website fingerprinting literature. They proposed the k-fingerprinting attack based on random decision forests and enabled attackers to infer which web page a client is browsing through encrypted or anonymized network connections. They demonstrated that Tor hidden services are easily distinguished from standard web pages, rendering them vulnerable to Website Fingerprinting Attacks.

FLOWPRINT [4] is a semi-supervised mobile-app fingerprinting prototype. The authors observe that mobile apps are composed of different modules that often communicate with a relatively invariable set of network destinations. This property is leveraged to discover patterns in the network traffic. Fingerprints are created based on temporal correlations among network flows between monitored devices and their destinations.

Zhuo and Zhang et al. [12] proposed a website-modeling method based on PHMM; they took advantage of the first tab and the second tab hidden relationship to improve accuracy in identifying a particular website instead of identifying web pages separately. Inspired by their study, we employ the PHMM for modeling user action patterns, which is a finer granularity. We can not only detect whether a user is visiting the targeted website but also identify his/her actions.

2.2. User Action Identification

User action identification has been extensively treated in the domain of personal mobile devices. Apps leverage the Wi-Fi and cellular network of mobile devices to send and receive data. Users perform several actions while interacting with apps and generate data transmissions. The network traffic sequence of a given action typically follows a pattern that depends on the nature of the user–app interaction of that action. These patterns can be used to recognize specific user actions related to a particular app of interest in generic network traces [18].

Conti and Mancini [19] proposed a framework to infer which particular actions the user executes on some apps installed on her mobile phone. Dynamic Time Warping and Random Forest were used to measure the similarity between traffic sequences and classify unseen traffic traces, respectively. The authors considered seven popular apps with different purposes from the official Android market to assess their approach's performance and showed that the accuracy and precision were higher than 95%.

Similar to [19], Fu and Xiong investigated how to exploit encrypted Internet traffic for classifying in-App usages. They developed a system named CUMMA for classifying usages of mobile messaging Apps by jointly modeling user behavioral patterns, network traffic characteristics (packet length and time delay), and temporal dependencies [20]. In their work, traffic flows were segmented into sessions with a number of dialogs; then, the dialogs were classified into single-type usages or outliers. A clustering Hidden Markov Model-based method was used to detect mixed dialogs from outliers to sub-dialogs or single-type usage. Experiments on WhatsApp and WeChat demonstrated the effectiveness and efficiency of their proposed method. In our paper, we not only identify user actions from network traffic but also recognize its corresponding user type based on the constructed user-role models.

2.3. Other Related Works

A few previous papers are notable for using different techniques on similar problems. He and Yang [13] selected features such as burst volumes and directions to represent the application behaviors and leveraged PHMM to model different types of applications (Web, FTP, P2P, and IM) on Tor. Their experimental results demonstrated that PHMM is quite good at modeling network traffic.

Network traffic analysis technology has been extended to the mobile smart home equipment research field. PINGPONG [21] automatically extracts the fingerprints from network traffic generated by the smart home devices and recognizes their actions (such as turning on or off the light). Similarly, HoMonit [22] analyzes the network traffic generated by smart home devices to determine the actions performed on the home device applications. Li and Feng et al. [23] proposed generating fine-grained fingerprints based on the subtle differences between the file systems of various firmware images. They applied the natural language processing technique to process the file content and used the document object model to obtain the firmware fingerprint. Using this fingerprinting approach, they were able to recognize firmware on the Internet. However, their approach has to interact actively with the firmware, thus is easy to be detected. In this paper, we propose a passive method to detect a particular web application of interest.

Network traffic analysis has also been extended to intelligent software testing. In work [24], an automated penetration-testing framework is built to detect vulnerability through traffic analysis. Pyshark is used to capture the traffic in IoT devices' four different states (booting, mobile application interaction, firmware mode, and offline mode). Then, 'tshark' is used to read the .pcap files and check for vulnerabilities such as insecure firmware, lack of transport encryption, and insecure network services. Similar to [23], this approach also interacts actively with the firmware.

3. Preliminaries

3.1. Definitions

In this section, we first define several definitions that will be used in this work and then present the formal definition of our user action identification problem.

Definition 1. (*User action*). *A user's action, a_i, is one interaction that a user carries out on a web application. For example, a user types in his/her account and password to log in to the web application. Users' actions set $A = \{a_1, a_2, \ldots, a_n\}$ contains interactions that a user can perform on a web application.*

Definition 2. (*Traffic flow*). *A traffic flow F is referred to as network packets generated by a series of user actions during a certain time interval T. We present $T = [t_{start}, t_{end}]$ and $F = \{(p_k, t_k)\}_{k=1}^{m}$, in which p_k represents a packet, t_k is the corresponding timestamps of p_k and $t_{start} = t_1$, $t_{end} = t_m$.*

Definition 3. (*Session*). *A session $S = (\{p_i, t_i\})_{i=1}^{n}$ is a series of network packets generated by a certain user u_i in a time range $T = [t^s_{start}, t^s_{end}]$. $p_{i=} \pm l_i$ is a signed integer that represents the packet length and direction, and its corresponding timestamp is t_i.*

Definition 4. (*Burst*). *A burst b is a packet sequence generated by a user action; it is a section of S. In addition, $\forall(p_j, t_j)$ in b, we have $\Delta t = t_j - t_{j-1} < \alpha$, where α is a predefined time interval threshold. In agreement with the previous study [25], 95% of all packets generated by a user action arrive at most 4.5 s after their predecessors.*

Definition 5. (*User role*). *A user role U_i is one type of user of a web application. It is modeled as a directed and connected graph G = (V, E), where V is the set of nodes, which denotes user actions, and E is the set of edges, which represents transitions between actions.*

For example, a user performs action v_1 after v_0 according to some transition probability distribution. The transition probability distribution in one step can be represented as a $|V| \times |V|$ matrix $P = (P_{xy})$, $x, y \in V$, where P_{xy} denotes the probability of moving from x to y in one step.

Definition 6. (*Web application*). *A web application W predefines various permissions for different users to perform tasks. Thus, the web application model consists of two levels with respect to users and the action transitions they can carry out.*

Figure 2 is an example of a web application with two user roles, i.e., Registered Users and Guests. As depicted in this figure, v_i denotes a user action, and P_{ij} represents the transition probability between v_i and v_j.

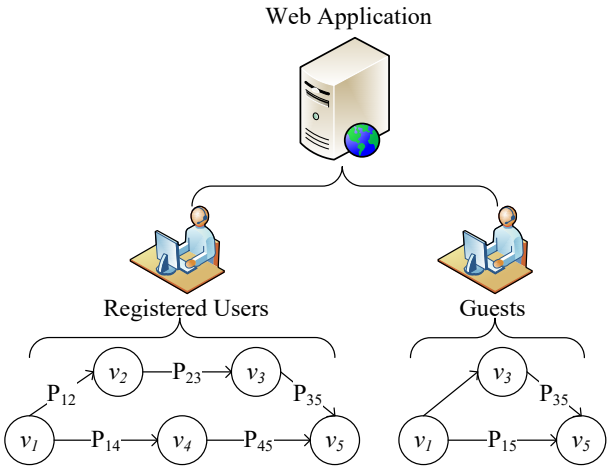

Figure 2. Web application model.

Definition 7. (*Clickstream*). *A clickstream* $C = \langle U, A, T \rangle$ *records users' interactions with a certain web application over a period of time* $[t_{start}, t_{end}]$. *U is the user role set.* $A = \{a_1, a_2, \ldots, a_n\}$ *is the set of actions that different users perform. T is the set of corresponding timestamps.*

Definition 8. (*Click*). *A click c records one request and is represented as a triple* $c = \langle u, a, t \rangle$, *where u is the user identity, a is the action u performs, and t is the corresponding timestamp.*

The most important quantities for the work are included in Table 1.

Table 1. The definition of important quantities.

Quantity	Description
T	a certain time interval
p_i	a packet in network traffic, including the packet's length and direction
l_i	the length of p_i
t_i	the timestamp of p_i
Δt	the time interval between two consecutive packets
α	a predefined time interval threshold to split a session into bursts
P_{xy}	the transition probability between user actions x and y

3.2. Identification Problems

We present the definition of our identification problem followed by an example. The problem we set out to solve is:

Problem 1. *Given a stream of burst b, detect whether it is generated by a certain user action, i.e., we can identify a user action from unknown network traffic.*

Example 1. *Consider users carrying out interactions with a social forum (e.g., registered users log in to the forum, browse several posts, and leave some comments). We collect network traffic generated by user actions performed on this forum and use PHMM to train different action patterns (e.g., browsing a post or leaving a comment). These patterns are then used to identify corresponding user actions from unknown network traffic.*

Problem 2. *Given a traffic flow F, we try to identify its corresponding user actions and the role of the user that performs these actions. Then, we can identify a monitored web application W or its logically identical web application W'.*

Example 2. *Consider a social forum with two user roles, i.e., Registered Users and Guests. As users of the same role often present a similar action transition pattern (e.g., registered users log in to the forum, browse several posts, and leave some comments), models u_r and u_g can be built to represent the action transitions of Registered Users and Guests, respectively. These role models can be built from the historical clickstream data (e.g., the forum's access log). We then collect network traffic generated by users and use PHMM patterns to identify corresponding user action sequences from traffic in a monitored network. The identified user action sequences are matched to u_r and u_g to determine their corresponding user types.*

4. User Action Identification

In this section, we describe our user action identification method in three parts. The first part collects traffic data for training the traffic classifier. The second part constructs user action patterns from the network traffic. The third part describes how these patterns are used to identify user actions from unknown network traffic. We detail the traffic collection in Section 4.1 and describe the methodology used to generate action patterns in Section 4.2, while in Section 4.3, we identify user actions from network traffic.

Figure 3 shows our user action identification process. It is composed of offline training and online classification. In the training phase, traffic sequences of different kinds of user actions are captured, classification features are extracted, and user action patterns are built. These patterns are then used to detect corresponding actions from traffic sequences in the online classification phase.

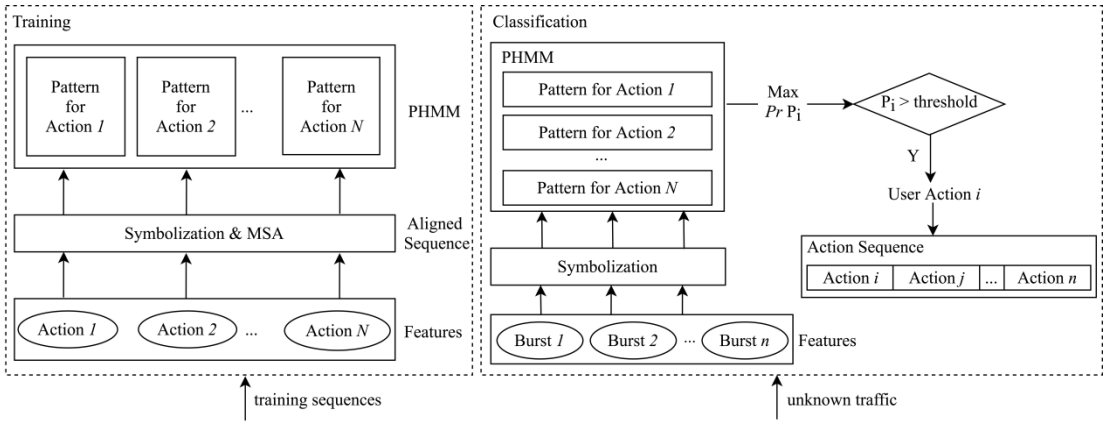

Figure 3. An overview of the user action identification process.

4.1. Traffic Collection

A desktop computer is configured as an Access Point (AP) in a controllable small-scale network to record the network traffic of users accessing a certain web application. The network traffic collection approach is depicted in Figure 4. We shield the traffic generated by other behaviors other than users' access to the target Web application and record the start and end time of users' interactions with the Web application so as to filter out abnormal traffic flow as much as possible.

The captured data inevitably contains some irrelevant network traffic, such as the ACK, ARP, and DHCP packets that do not carry payload and packets triggered by advertisements on the webpage. These irrelevant packets are filtered at first. In addition, we assume a traffic sequence can be attributed to a specific user by examining the IP addresses or cookies. In the case of Network Address Translation (NAT), the proposed method in [26] can be used to identify users. Thus, IP addresses are utilized to filter network traffic generated by other irrelevant users that connect to the same Wi-Fi access point.

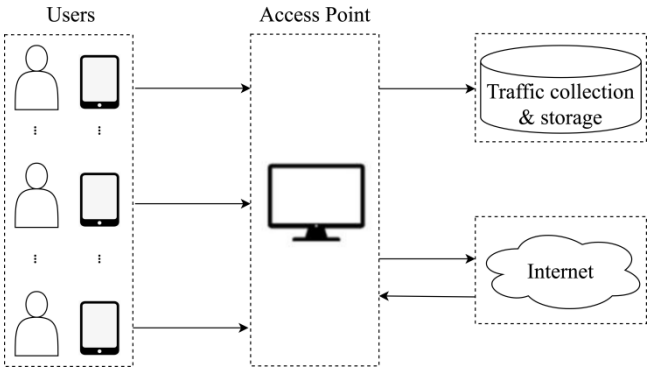

Figure 4. Network traffic collection.

Packet time information is easily affected by network conditions; thus, it is not as stable as packet length. In this paper, like previous works [6,12,13], we utilize the packet length and direction to represent a packet, i.e., a traffic sequence $S = (p_1, p_2, \ldots p_n)$, where $p_i = \pm l_i$, l_i is the length of the ith packet, and the sign indicates its direction. The traffic sequences are symbolized and aligned to construct the associated action patterns. We describe the traffic sequence symbolization and alignment in the following part.

4.2. Traffic Classifier Building

According to the previous study, the network traffic sequence of a given user action typically follows a pattern that depends on the nature of the user–service interaction of that action. These patterns can be used to recognize a specific user's actions from online network traffic. In this part, we use a machine learning method PHMM to model different users' actions.

4.2.1. Profile Hidden Markov Model

In computational biology, PHMM was first introduced in [27] and has been applied to the task of evaluating a new sequence for membership in a family of sequences as well as the construction of genetic linkage maps. Biological sequences of the same family often have differences, but they normally maintain similar functions and structures. It is a fact that certain positions in a family are more conserved than other positions and allow substitutions less readily. Like biological sequences, network traffic sequences generated by the same type of user actions (e.g., two users log in to a web application separately) may fluctuate due to the network environment, but the essential functionality part can still be the same. For example, each web page contains some special objects (e.g., CSS files, pictures), and clicking on a web page will initiate packet sequences that can uniquely characterize it.

PHMM consists of five basic types of states, i.e., a beginning state, several match states, insert states, delete states, and an end state. Match states are the core of the model. Every match state is represented by a set of emission probabilities for each output symbol. Insert states are portions of sequences that do not match anything in the match states. PHMM also provides delete states to deal with arbitrary long gaps in sequences. Since arbitrary gaps can introduce many transitions in the model, representing them with deleted states is one of PHMM's advantages. From this perspective, PHMM is suitable for modeling network traffic sequences because it can exploit positional information contained in the sequences and is robust to sequence fluctuations (e.g., insertions and deletions).

The structure of PHMM is shown in Figure 5; the circles are delete states, the diamonds are insert states, the rectangles are match states, and the solid arrows stand for state transitions. In our work, match states represent the "functionality" structure of network traffic. Insert states allow variations between the observed traffic sequences produced by the same user actions. The delete states allow packets at a given position in a sequence to be removed or replaced with ones that do not fit the model. In general, if an observed

sequence can be mapped to one model, the path will be Begin $\rightarrow M_1 \rightarrow \ldots \rightarrow M_n \rightarrow$ End, but sometimes it may pass through many delete or insert states before reaching the end. The PHMM patterns for different user actions are constructed in the training phase, which is described in detail later.

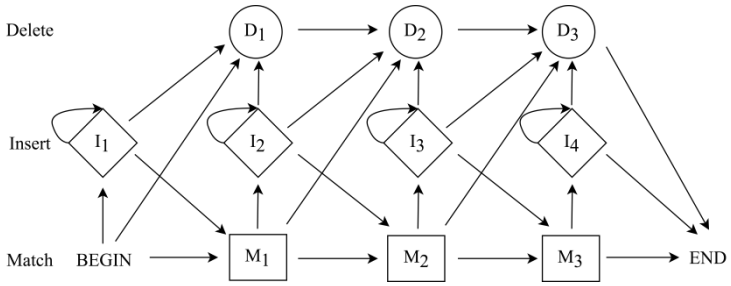

Figure 5. A PHMM example.

4.2.2. Traffic Pre-Process

In order to build the user action patterns, the collected network traffic needs to be pre-processed. The clustering and Multiple Sequence Alignment (MSA) algorithms [28] are utilized to accomplish this task. Since packet length and direction are used to represent traffic sequences, the packet value range is from -1500 to 1500. In order to make the PHMM patterns more robust, we utilize symbolization to reduce the number of observable states. In this part, the *K*-means algorithm is used to cluster all the packets in sequences. To determine a proper cluster number *K*, the intra-cluster distance $intra_K$ and inter-cluster distance $inter_K$ are calculated according to (1) and (2), respectively, in which C_i is the ith cluster, and $center_i$ is the cluster center of C_i. We choose *K* with the minimum $intra_K/inter_K$ as the final number of clusters and assign a distinct symbol (e.g., a letter) to each cluster. For each packet in a sequence, we find out the cluster it falls in and replace this packet with the corresponding symbol. The symbolization algorithm is given in Algorithm 1. After the symbolization, a packet sequence may look like 'QRJPQNNNCBBBB ... '.

$$intra_K = \frac{1}{K}\sum_{i=1}^{K}\sum_{x \in C_i} |x - center_i| \tag{1}$$

$$inter_K = min|center_i - center_j|i, j \in \{1, 2, \ldots, K\} \tag{2}$$

Algorithm 1 Traffic Sequence Symbolization

Input:
A traffic sequence *S*
Output:
Symbolized sequence *seq*
1: $l \leftarrow 0$, $sign \leftarrow 0$, $seq \leftarrow$ null;
2: **for** each packet *p* in *S* **do**
3: $l \leftarrow$ length of *p*;
4: **if** *p* is an outgoing packet **then**
5: $sign \leftarrow -1$;
6: **else**
7: $sign \leftarrow 1$;
8: **end if**
9: $l \leftarrow l \times sign$;
10: $seq \leftarrow seq$ + *Symbolize(l)*
/ convert l into its symbolization label and add it to the symbolized sequence */*
11: **end for**
12: return *seq*

To have an intuitive sense of how symbolized sequences extract differences among different interactions, we select four different actions that users carry out on a social forum (i.e., login, reading, posting, and log out) and visualize their symbolized sequences in Figure 6. We can see that the symbolized sequences show significant differences. As a result, these symbolized sequences can be used to distinguish different users' actions.

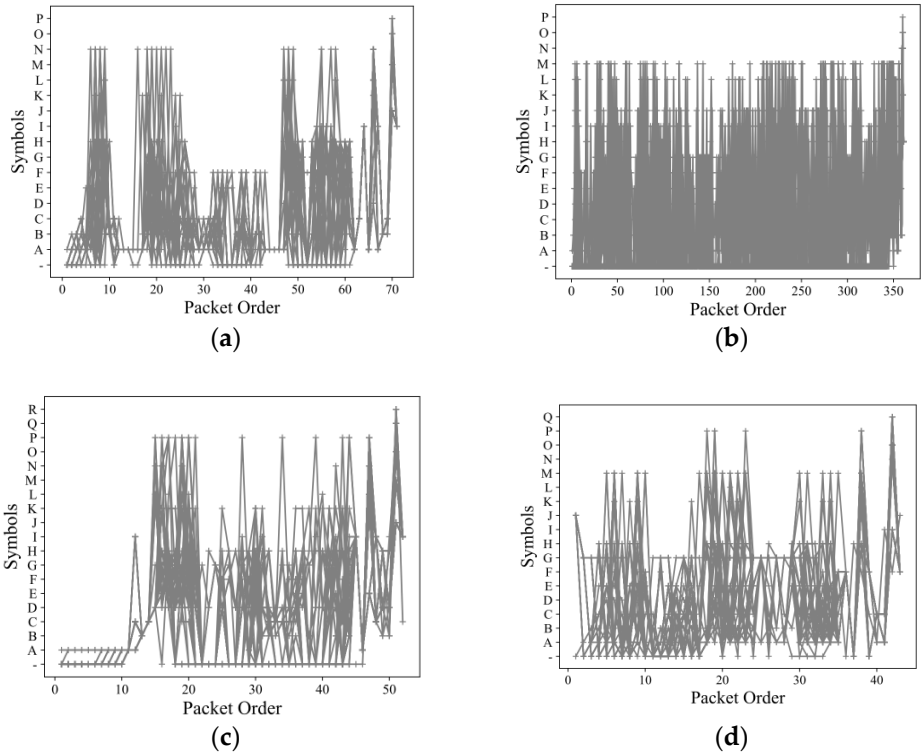

Figure 6. Symbolized sequences of different user actions. (**a**) Login; (**b**) reading; (**c**) posting; (**d**) log-out.

Affected by the network environment, traffic sequences generated by the same user actions may vary in length. We can build profiles from these unaligned sequences using the Baum–Welch procedure [29], but it is easy to get stuck around a local optima [30] and much more time-consuming. Thus, in the training phase, packet sequences of the same actions (e.g., leaving a comment under an online post) are aligned to the same length. Here, the Clustal Omega (the latest addition to the Clustal family, allowing thousands of sequences to be aligned in only a few hours [31]) is used to accomplish multiple sequence alignments. Figure 7 shows an example of multiple sequence alignment. Five sequences are aligned to the same length (115 characters), then they are used to construct a PHMM pattern.

```
> 1
D L O A S I R R I B B B B B B N B B E P N B B B B B Q L B I R R I R N N Q O R R H A I A R R R H A A I I I E T J C G G K H M
I B B B B B B B B B B B B B B B B B B R P
> 2
I I I I MB B B B B B B B B E P B B B B B B D L E O A A I A Q L H B B B B B B B B Q Q B Q I I C G G K G C H MB B B B B B B B B
B B B B B B B B B R Q B B T P
> 3
I MB B B B B B B B E P B B B B B B Q N I I A A C G G K H MB B B B B B B B B B B B B B B B B B R I P P A N I
> 4
I MB B B B B B B B E P B B B B B B I I I P P D L O A C G G K H M I B B B B B B B B B B B B B B B B B B R I I I I
> 5
MB B B B B B B B B B B B B B E P A N P P I B B MD L O A B B B B B A B B B B B B B B N N Q Q I MB B B B B B B B B B B B B
B B B B B B P MB B B B B B B B B B T MB B B B B B B B B B B B B B B B B B P I
```

(**a**)

Figure 7. *Cont.*

```
> 1
D L O A S I R R I B B B B B B N B B E P N B B B B B Q L B I R R I R N N Q O R R H A I A R R R H A A I I I E T J C - - -
- - - - - - - - - - G G K H M I B B B B B B B - - - - - - - B B B B B - - - - - B B B B B B R P - - - - -
> 2
- - - - - I I I I M B B B B B B B B B E P B B B B B B B D L E O A A I A Q L H B - - - - - - B B B B B B Q Q B Q I I C - - -
- - - - - - - - - - G G K G C H M B B B B B B - - - - - - - B B B B B B B B B B B B R Q B B T P - - - - -
> 3
- - - - - - - - I M B B B B B B B B E P B B - - - - - - - - - - - - - - - - - - - - - - - - B B B B Q N I I A A C - - -
- - - - - - - - - - G G K H M B B B B B B B B - - - - - - - B B B B B - - - - - B B - B B B R I P P A N I
> 4
- - - - - - - - I M B B B B B B B B E P B B - - - - - - - - - - - - - - - - - - - - - B B B B I I I P P D L O A C - - -
- - - - - - - - - - G G K H M I B B B B B B B - - - - - - - B B B B B - - - - - B B B B B B B R I I I I - -
> 5
- M B - - B B B B B B B B B B B B B B E P A N P P I B B M D L O A B B B B B A B B B B B B - - B B N N Q Q I M B B B B B B
B B B B B B B B B B B B B P M B B B B B B B B B B B B T M B B B B B B B B B B B B B B B B B B B B B B P I - - - -
```

(**b**)

Figure 7. An example of the multiple-sequences alignment. (**a**) The symbolized packet sequences (unaligned); (**b**) the aligned packet sequences.

4.2.3. Training the Classifier

After pre-processing the network traffic, different action patterns are constructed to identify user behaviors from unknown network traffic. In this work, we use a tool named HMMER to build Profile Hidden Markov Models [32]. HMMER uses a more complex model called Plan 7. As shown in Figure 8, the Plan 7 architecture is substantially different from the original model in Figure 5. The main model of Plan 7 consists of *B*, *M*, *D*, *I*, and *E* states, and its probability parameters are generally learned from data in a multiple sequences alignment. The other states {*S*, *N*, *C*, *T*, and *J*} are called special states; with the help of these states, the Plan 7 architecture can model a complete sequence, regardless of how much it matches the main model. Readers can refer to [33] for more information about Plan 7.

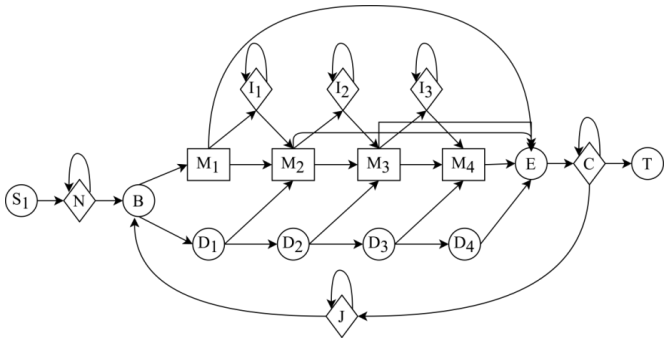

Figure 8. Plan 7 architecture.

4.3. Traffic Identification

In the classification phase, the established patterns are used to identify the targeted traffic. We extract features (i.e., packet length and direction) from the traffic to get packet sequences and symbolize them in the same way as the training phase. Then, we compute the likelihood that the target traffic sequence is generated by a specific user action. For example, if we model N different user actions in the training phase, N PHMM patterns are built. During the classification phase, each packet sequence will gain N probabilities. We choose the maximum probability P_i and label this sequence as generated by the ith action. The detailed classification process is described as follows.

We split a traffic trace into bursts and try to identify their corresponding user actions. As depicted in Figure 3, suppose N PHMM patterns $\{M_1, M_2, \ldots, M_N\}$ are built; we symbolize each burst and compute the probabilities that it is produced by each pattern. The maximum probability P_i indicates that the burst is generated by the ith user action. The user action identification algorithm is given in Algorithm 2.

Algorithm 2 User Action Identification

Input:
A network traffic trace T
PHMM patterns $M_1, M_2, \ldots, M_N$
Output:
User action sequence S
$i \leftarrow 1$, $x \leftarrow 0$, $start \leftarrow 1$, $burst \leftarrow null$, $S \leftarrow null$;
$\alpha \leftarrow$ the predefined threshold
for ith packet p_i in T do
    if $start == 1$ then
        //start == 1 indicates p_i is the first packet in a burst
        $burst \leftarrow burst + p_i$;
        $start \leftarrow 0$;
    else if $t_i - t_{i\_1} < \alpha$ **then**
        $burst \leftarrow burst + p_i$;
    else
        Symbolize *burst* with Algorithm 1;
        $x \leftarrow argmax(Pr(burst \mid M_x), x = 1, 2, \ldots, N)$;
        //the burst is generated by the xth type of user action
        $S \leftarrow S + action_x$;
        $burst \leftarrow null$;
        $start \leftarrow 1$;
    end if
end for
return S

5. Adaptive Symbolization

In Section 4.2, the *K*-means clustering algorithm is used to symbolize traffic sequences. Symbolization is an important series analysis method. It converts a sequence of real numbers into a sequence of symbols according to its numerical characteristics [34]. The goal of symbolization is to retain as much information as possible with the smallest symbol set. Choosing an appropriate symbolization strategy is a difficult problem.

The most common approach for coarse symbol definition involves partitioning the range of sequences into a finite number of regions. Each region is associated with a specific symbolic value. The original measurement is uniquely mapped to a particular symbol depending on the region in which the measurement falls. For example, Zhuo [12] divided the network traffic sequence into equal intervals and assigned different symbols to each interval to form a symbolized sequence. As shown in (3), the size of the symbol set N is adjustable, and the symbol s_i of measurement x_i in the sequence is determined by the interval in which x_i falls. It is easy to see that different symbol sets can be obtained by adjusting the interval division.

$$s_i = \begin{cases} N-1 & x_i \geq X_{N-1} \\ \ldots & \ldots \\ 2 & X_3 > x_i > X_2 \\ 1 & X_2 > x_i > X_1 \\ 0 & x_i < X \end{cases} \tag{3}$$

Both the equal interval and the *K*-means clustering method used in Section 4.2 have obvious defects. In the former method, the division of the interval is randomly specified. Even though few or no packets fall in certain intervals, the algorithm still assigns symbols to these sparse intervals. However, for intervals that contain lots of packets, the algorithm cannot assign more symbols to reflect the statistical distribution characteristics. Thus, although the equal interval symbolization method is very straightforward, it cannot describe the traffic sequence well. Similarly, the *K*-means algorithm clusters packet in

traffic sequences and assign symbols to each cluster. However, it is hard to make sure the generated symbolized sequences describe the original sequences well.

In order to tackle these problems, this paper presents an adaptive symbolization method under the condition of limiting the size of the symbol set. Our goal is to find the lowest information loss division method under a fixed symbol set size K. As depicted in Figure 9, the packet length x_i varies from -1500 to 1500. Then, the problem is to divide the packet length range into K intervals and assign symbols for each of them, i.e., we need to find out the appropriate interval length $\tau_1, \tau_2, \ldots, \tau_K$.

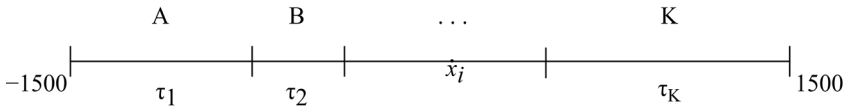

Figure 9. An example of a packet sequence interval division where the uppercase letters are the symbols and τ_i represents the interval length.

Suppose p_j is the probability of element x_j in an original sequence X. The distance D between the original and symbolized sequences can be calculated by (4), which $E(I_{\tau_i})$ is the expected value of an original sequence in the ith interval. We need to find the symbolization method that yields the minimal D.

$$\begin{aligned} D &= \sum_{i=1}^{K} \left(E(I_{\tau_i}) \cdot \sum_{j \in \tau_i} p_j - E(X) \right)^2 \\ &= \sum_{i=1}^{K} \left(E(I_{\tau_i}) \cdot \sum_{j \in \tau_i} p_j - \sum_{j \in \Omega} x_j p_j \right)^2 \end{aligned} \tag{4}$$

The value of $E(I_{\tau_i})$ depends on interval length τ_i. Suppose there are packets that fall in the ith interval, $E(I_{\tau_i})$ can be calculated with (5).

$$E(I_{\tau_i}) = \sum_{j=1}^{N_{\tau_i}} x_j p_j \tag{5}$$

It is obvious that $E(I_{\tau_i})$ is related to $\sum_{j \in \tau_i} p_j$, i.e., when $\sum_{j \in \tau_i} p_j$ changes, the $E(I_{\tau_i})$ changes too. We set $\sum_{j \in \tau_i} p_j = a_i$, then the corresponding expected value $E(I_{\tau_i})$ can be represented as a random variable X_i. Thus the distance D can be represented by (6).

$$D = \sum_{i=1}^{K} (a_i \cdot X_i - E(X))^2 \tag{6}$$

Then, finding the minimal D turns into an efficient estimation problem, i.e., estimating X with $\sum_{i=1}^{K} a_i \cdot X_i$, where $\sum_{i=1}^{K} a_i = 1$.

Suppose $D(X) = \delta^2$, in order to meet (7), we define conditional extremum function (8) based on the Lagrange multiplier.

$$\begin{aligned} D\left(\sum_{i=1}^{K} a_i \cdot X_i \right) &= \sum_{i=1}^{K} a_i^2 \cdot D(X_i) \\ &= \sum_{i=1}^{K} a_i^2 \cdot D(X) \\ &= \sum_{i=1}^{K} a_i^2 \cdot \delta^2 \end{aligned} \tag{7}$$

Let us set

$$f\left(a_1, a_2, \ldots, a_K, \delta^2, \lambda\right) = \sum_{i=1}^{K} a_i^2 \cdot \delta^2 + \lambda \cdot \left(\sum_{i=1}^{K} a_i - 1\right) \tag{8}$$

Take the partial derivative of (8) to produce

$$\begin{cases} \frac{\partial f}{\partial a_1} = 2a_1 + \lambda = 0 \\ \frac{\partial f}{\partial a_2} = 2a_2 + \lambda = 0 \\ \cdots \\ \frac{\partial f}{\partial a_K} = 2a_K + \lambda = 0 \\ \frac{\partial f}{\partial \lambda} = \sum\limits_{i=1}^{K} a_i - 1 = 0 \end{cases} \tag{9}$$

Solving Equation (9), one can obtain

$$\begin{cases} a_i = -\frac{\lambda}{2} & P_1 \\ \sum\limits_{i=1}^{K} a_i - 1 = 0 & P_2 \end{cases} \tag{10}$$

Put Equation P_1 into P_2 to get

$$-\frac{K}{2}\lambda - 1 = 0 \tag{11}$$

Namely,

$$\lambda = -\frac{2}{K} \tag{12}$$

Putting λ into (10), one can obtain

$$a_i = \frac{1}{K} \tag{13}$$

As mentioned earlier, $a_i = \sum\limits_{j \in \tau_i} p_j$ is set to represent the probability of measures that fall in the ith interval. Then, we have proven that the minimal D can be obtained when $a_i = \frac{1}{K}$. Thus, in order to retain information about the original sequence, we propose an equal probability symbolization method, which guarantees a basically equal number of packets falling in each interval. Namely, the traffic sequences are divided into K intervals, and the sum of the occurrence probabilities of packets in each interval is equal to $1/K$.

Suppose $S = \{S_{a_1}, S_{a_2}, \ldots, S_{a_N}\}$ is the collected network traffic generated by different users' actions, in which S_{a_i} represents traffic sequences of action a_i. All the packets in S are sorted in ascending order to get S'. The occurrence probability p_i of each packet x_i in is calculated. Then, S' is traversed sequentially. The probabilities of those traversed x_i are accumulated. When the summation is equal to $1/K$, we label the latest traversed x_i as a division point. Continue to traverse S'; one can locate the remaining $K - 2$ interval division points. The algorithm of the proposed symbolization method is shown in Algorithm 3.

Algorithm 3 The Proposed Symbolization Method

Input:
The collected traffic sequence set S
Symbol set size K
Output:
Interval Sets $\Psi = \{\tau_1, \tau_2, \ldots, \tau_K\}$
S'←all the packets in S are sorted in ascending order;
$N \leftarrow sizeof(S')$;
$i \leftarrow 1$, $j \leftarrow 1$, $prev \leftarrow 0$, $sum \leftarrow 0$, $p \leftarrow 1/N$, $\Psi \leftarrow$ null;
for x_i in S' **do**
    $prev \leftarrow sum$;
    $sum \leftarrow sum + p$;
    if $prev <= 1/K$ and $sum > 1/K$ **then//***totally K intervals*
        if $|prev\text{-}1/K| < |sum\text{-}1/K|$ **then**
            $x_{\tau_j} = x_{i-1}$
            $i \leftarrow i - 1$;
        else
            $x_{\tau_j} = x_i$
        end if
        $sum \leftarrow 0$, $prev \leftarrow 0$;
        $\tau_j \leftarrow$ all elements from $x_{\tau_{j-1}}$ to x_{τ_j}
        //τ_j *contains all the elements between two adjacent interval points*
        $j \leftarrow j + 1$;
    end if
end for
$\Psi \leftarrow \Psi + \tau_j$;
return Ψ;

6. Web Application Identification

We have proposed a PHMM-based user action (user–web application interaction) identification method. In this section, our proposed method is applied to a web application identification problem. We first introduce the necessary background of modeling a web application, then build user role models from clickstreams, and identify web applications from network traffic at last.

6.1. Web Application Model

A web application is often designed to provide different capabilities for different types of users. For example, most users log in, browse or leave comments on a social forum, but there are also guests who can only browse several blogs but cannot leave comments. Therefore, we can model a web application from the users' perspective.

In this paper, we assume the user types are known in advance and try to build the user role models. The user role set U represents different user types of a web application.

As described above, different types of users are allowed to perform different kinds of actions on a web application. User roles are proposed to represent them. It is observed that every user role $u \in U$ has its distinct action transition pattern; thus, the Markov chain models can be used to represent different roles. Markov chains are types of probabilistic finite-state machines; they describe the probability of reaching one state to another and are widely used in many study fields (e.g., speech recognition and bioinformatics). Applying Markov chains to describe user roles is natural since interacting with a web application creates a sequence of states in which each state corresponds to a kind of action. Each Markov chain can be mathematically represented by a transition matrix. A Markov chain and its corresponding transition matrix are presented in Figure 10 and Table 2 separately. The intersection of row S_i and column S_j denotes the transition probability from state S_i to S_j. For example, the transition probability from S_1 to S_2 is 0.8.

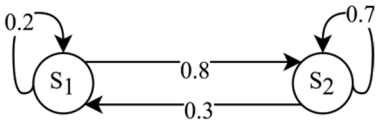

Figure 10. A Markov model.

Table 2. The corresponding transition matrix.

	S_1	S_2
S_1	0.2	0.8
S_2	0.3	0.7

6.2. User Role Construction

In this step, we try to model user action transitions from clickstream data. Clickstreams are traces of click-through events generated by online users during each web browsing session [35] and have been used to model web traffic and user browsing patterns [36–39]. In this work, a sequence of HTTP requests made by a user on a web application is a clickstream. As we all know, most requests correspond to the user explicitly fetching a page by clicking a hyperlink, although some requests may be programmatically generated [40]. As mentioned in Section 4.1, we also assume a clickstream can be attributed to a specific user by examining the IP addresses or cookies.

In this work, a session is used to represent a user's click sequence during a single visit to the web application. The clickstream is divided into sessions. As a matter of fact, not all users explicitly end their sessions by logging out of the web application. According to this, we consider two situations to indicate a session is over. A session is regarded as over if the user logs out actively; otherwise, we assume a session is over if the user does not make any requests in a predefined time interval threshold. According to the previous work, the threshold is set to 20 min [41].

Since different user roles have different capabilities, their click transition probabilities are different. In this paper, we build a Markov Chain model for each user role to characterize the transitions between clicks during sessions. In the role model, each state is a kind of user action, and edges are transitions between two consecutive actions. We use ENTER and EXIT states to mark the beginning and end of a session. In the case of a traditional web application, the LOGIN and LOG-OUT actions correspond to these two states, respectively.

We focus on calculating the corresponding transition probability matrix T of the Markov Chain. Every entry $T[i][j]$ represents the transition probability from state i to state j. In order to simplify the logic, we first store the occurrence count of transitions from state i to j in $T[i][j]$ instead of the transition probability. After obtaining the transition occurrence matrix, we divide each $T[i][j]$ by the total number of row i (the total transition occurrence from state i to other states) and convert it to the transition probability matrix. Since the transition probabilities are changing continuously, this conversion can make our computation much easier. The implementation of building a user role model is given in Algorithm 4.

6.3. Web Application Discriminator

As described in 4.3, user action patterns are built based on PHMM. A traffic trace is divided into bursts, and each burst can be associated with a specific user's action. Then, the traffic trace can be represented as a user's action sequence. According to the action transition matrices of user role models, we can calculate the likelihood that the identified action sequence is generated by each type of user. The maximum probability determines its corresponding user role. Then, we identify the targeted web application based on user roles. The proposed web application identification framework is depicted in Figure 11.

Algorithm 4 Transition Probability Matrix Calculation

Input:
a set of action sequences S
a transition occurrence matrix $T' \leftarrow null$
Output:
a transition occurrence matrix T
1: **for** each sequence s in S **do**
2: $n \leftarrow$ length of s;
3: $T'[ENTER][s\,[0].action]$++;
4: **for** $i \leftarrow 0$ to $n-1$ **do**
5: $T'[s[i].action][s[i+1].action]$++;
/* *calculate the occurrence of transitions between different user actions* */
6: **end for**
7: $T'[s[n].action][EXIT]$++;
8: **end for**
9: $i \leftarrow 0, j \leftarrow 0, sum \leftarrow 0$;
10: **for** $i \leftarrow 0$ to $T'.length$ **do**
11: **for** $j \leftarrow 0$ to $T'[i].length$ **do**
12: $sum \leftarrow sum + T'[i][j]$;
13: **end for**
14: **for** $j \leftarrow 0$ to $T'[i].length$ **do**
15: $T[i][j] \leftarrow T'[i][j]/sum$;
/* *convert transition occurrence matrix into transition probability matrix* */
16: **end for**
17: $sum \leftarrow 0$;
18: **end for**
19: **return** T;

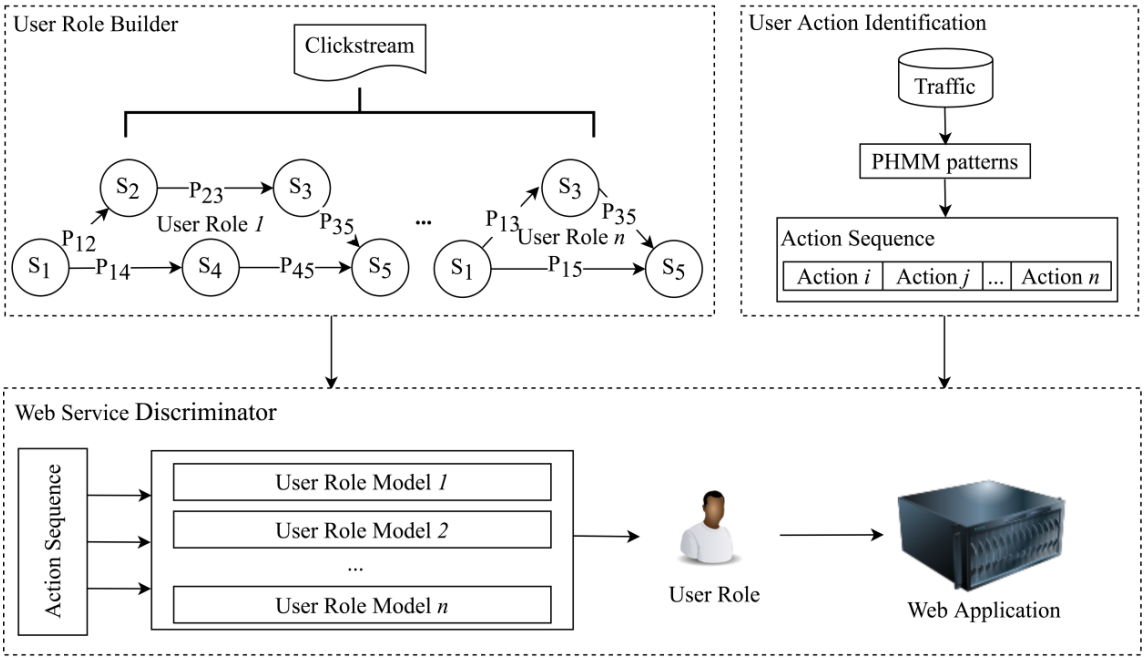

Figure 11. Web application identification framework.

7. Experiments

The first experiment compares our adaptive symbolization method with equal interval and *K*-means clustering methods. Then, our PHMM-based user action identification approach is evaluated. Finally, we build user role models from clickstreams and use the PHMM models to complete the web application identification task.

7.1. Traffic Symbolization

In this part, we conduct three symbolization methods, i.e., equal interval [12], *K*-means clustering [13], and our proposed adaptive method on a public network traffic dataset (https://github.com/Thijsvanede/FlowPrint/tree/master/datasets (accessed on 11 January 2023)) [4]. We run these methods on an Ali cloud ECS server2, then evaluate the distances between the symbolized and the original sequences (*D*) and time consumption separately.

Our proposed symbolization approach outperforms the *K*-means method from both *D* and time consumption. The performance of the adaptive algorithm in reducing the distance between the symbolized sequences and the original sequences is significantly better than that of the equal interval and *K*-means clustering algorithms so that it can retain the original information of the traffic sequences to the maximum. In terms of efficiency, the adaptive algorithm is better than *K*-means, and the time consumption is similar to that of the equal interval algorithm.

According to Figure 12a, with the increase of *K*, the distance between the symbolized sequence and the original sequence gradually converges. In order to determine the optimal value of *K*, the heuristic method is used to calculate the internal and external distance of the symbolized sequence according to Formulas (1) and (2). The interval center of our adaptive symbolization algorithm is the expected value of each interval, and C_i represents the *i*th interval. According to Formula (14), *K* with the smallest value is selected as the interval number after symbolization. We use our adaptive method to symbolize network traffic in the following experiments.

$$K = argmin(intra_K / inter_K) \tag{14}$$

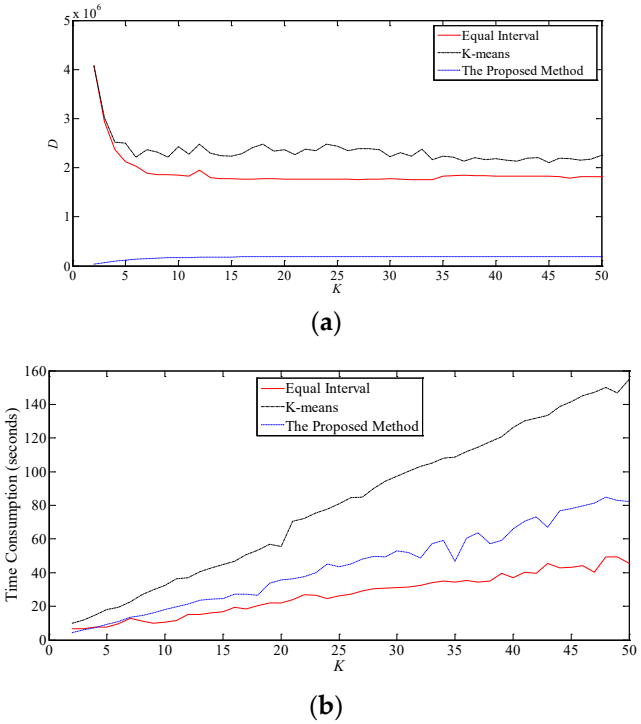

Figure 12. The comparison of equal-interval, *K*-means, and our adaptive symbolization algorithms. (**a**) Comparison of the distance between the symbolized and the original sequences. (**b**) Comparison of the time consumption.

The time complexity of *K*-means is O(*NKt*), where *N* represents the number of packets in the network traffic data set to be processed, *K* is the number of clusters, and *t* is the number of iterations of clusters. The equal interval and adaptive symbolization methods

have the same time complexity of O(N), where N is the number of packets in the network traffic data set.

However, according to Figure 12b, we can see that the equal interval method performs slightly better than the proposed method in terms of time-consuming. That is because the equal interval method does not involve statistics. For example, when we use equal interval symbolization, the interval length l is MTU/K, where MTU is the Maximum Transmission Unit. A packet with length *len* will be assigned to the ith interval, where $i = len/l + 1$. However, as described in Algorithm 3, the adaptive symbolization method needs to find the occurrence probability p_i of each packet x_i. This involves statistical work, which takes a little more time than the equal interval method.

7.2. Action Identification

In this section, we build a PHPWind-based (version 8.7, Hangzhou, China) web application on an Ali cloud ECS server. PHPWind is an open-source social forum template. Table 3 shows the actions performed by volunteers on this forum. Wireshark is used to capture network traffic data. Similar to [12], we perform each action in the forum 50 times and use the HMMER instruction *hmmbuild* to model these actions. Then, each action is executed 100 times again and used to match the constructed models with *hmmsearch* instruction. Here, *bit score* in the output of *hmmsearch* is used to help us classify actions. Because it does not depend on the size of the sequence database, it only relates to the PHMM and the target sequence and shows the significance of the match [32].

Table 3. Classic users' actions on the forum.

#	Action	Comment
1	LOGIN	A user logs into his/her account.
2	SCAN	A user browsers different parts of the forum.
3	READ	A user browsers blogs.
4	REPLY	A user leaves comments on blogs.
5	MODIFY	A user modifies his/her own blogs.
6	PREVIEW	A user previews his/her own blog before posting it.
7	POST	A user writes a blog and posts it.
8	LOG-OUT	A user logs out his/her account actively.

Three widely used machine-learning metrics, the *recall*, *precision*, and $F1$ score, are utilized to evaluate our user action identification algorithm.

$$recall = \frac{TP}{TP + FN} \tag{15}$$

$$precision = \frac{TP}{TP + FP} \tag{16}$$

where TP (True Positive) is the number that a user action is correctly identified as exactly the same monitored action, and FN (False Negative) is the number that a user action is incorrectly classified as a different action or a non-monitored action. FP is defined as the number of non-monitored user action that is incorrectly classified as being monitored.

The $F1$ score is defined as the harmonic mean of *recall* and *precision*, which is calculated as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{17}$$

Intuitively, the metric $F1$ with a score of 1 indicates a perfect classifier. The detailed identification results are listed in Table 4.

Table 4. Results of action identification.

#	Action	*Recall*	*Precision*	*F*1
1	Login	1	0.99	0.995
2	Scan	0.96	1	0.979
3	Read	0.95	0.98	0.964
4	Reply	0.96	0.58	0.722
5	Modify	0.98	0.95	0.965
6	Preview	0.97	0.98	0.975
7	Post	0.27	0.9	0.415
8	Log-out	1	1	1
Average		0.886	0.923	0.877

Figure 13 shows the confusion matrix of the identification results. In the confusion matrix, the horizontal axis represents the action types in the predicted label from the output of classifiers. In contrast, the vertical axis represents the one in a true label. Each cell along the main diagonal describes the correct rate of one single action identification, whereas the darker cell represents the higher correct identification rate. Accordingly, most single-action identification results are above 0.95. However, for action POST, the correct rate is very low (only 0.27). It is observed that 70% of the POST action is misidentified as the REPLY action.

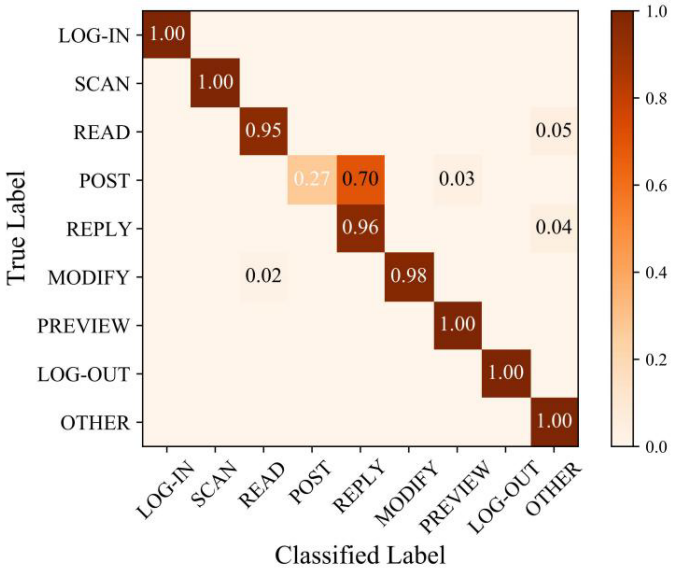

Figure 13. The confusion matrix of the user actions.

It is known that both POST and REPLY actions submit data to the web application. By checking the two actions manually, we find that both POST and REPLY actions lead to forms with the same URL path but different parameters. And these forms have similar structures. For example, posting a new blog may lead to a link *post.php?fid* = 2, while replying to this blog will jump to *post.php?action* = *reply&fid* = 2. Furthermore, POST and REPLY actions refresh the webpages. The new webpages have the same structure. Based on this observation, POST and REPLY actions are combined into a new type of POST action. Then, a new POST model is built based on the previously collected traffic sequences of POST and REPLY actions. This new PHMM model is used to identify POST/REPLY actions. The adjusted confusion matrix of the identified actions is shown in Figure 14. The average *recall*, *precision*, and *F*1 score are 0.973, 0.981, and 0.977, respectively.

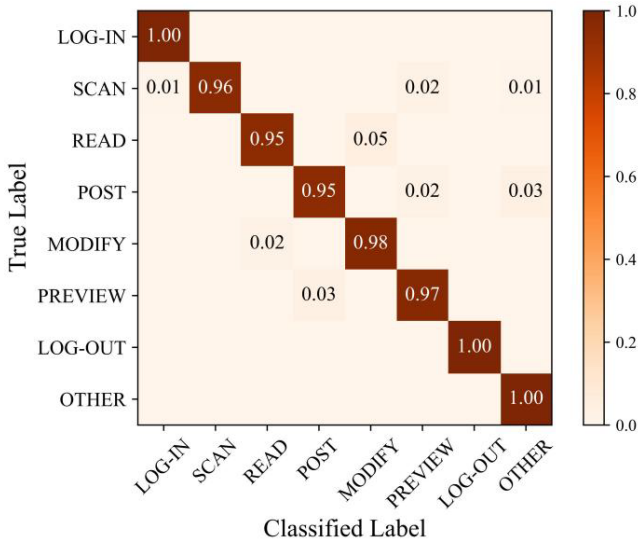

Figure 14. The adjusted confusion matrix.

In order to verify the efficiency of the proposed method, we conduct a comparative experiment against a Random Forest (RF) classifier. RF is a widely used ensemble learning algorithm that can achieve good performance in most classification cases.

Motivated by the literature [42], we take the packet length and the following statistical information for each sequence: maximum, minimum, mean, median absolute deviation, standard deviation, variance, skewness, and kurtosis to build feature vectors. In order to reflect the distribution information of the traffic sequences, the numbers of packets that fall in five intervals, i.e., 0–300, 301–600, 601–900, 901–1200, and 1201–1500 are also considered as features. Figure 15 shows the comparison results between our PHMM model and the Random Forest classifier used in [43].

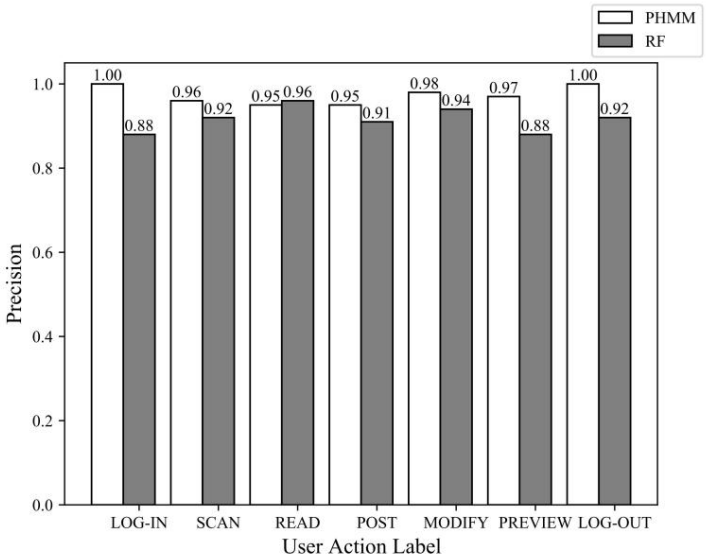

Figure 15. The comparison results of PHMM and Random Forest classifiers.

7.3. User Role Models

In this step, we investigate the feasibility of using clickstreams to model user roles. The study is based on detailed clickstreams (historical access log from 8 July 2018 to 7 May 2019) of a PHPWind-based (version 8.5, Hangzhou, China) social forum. Table 5 describes the dataset summary. In total, our dataset includes 1,810,873 clicks from registered users. After discarding events with missing fields or HTTP status associated with error codes (e.g.,

301, 302), there are 1625 valid HTTP requests. We manually analyze the clickstreams and characterize each click with an IP address, a Unix epoch timestamp, and the corresponding user action. For example, the LOGIN action represents a user logging into his/her account, and the REPLY action indicates when a user leaves comments on blogs.

Table 5. Summary of the clickstream data.

Dataset	Users	Clicks	Sessions
access.log	44	1,810,873	1625

The possible action transitions of registered users are shown in Figure 16, which states that LOGIN and LOG-OUT mark the beginning and end of each session, respectively. To reduce the complexity of this figure, edges with a probability below 5% are pruned (except for transitions to the LOG-OUT state).

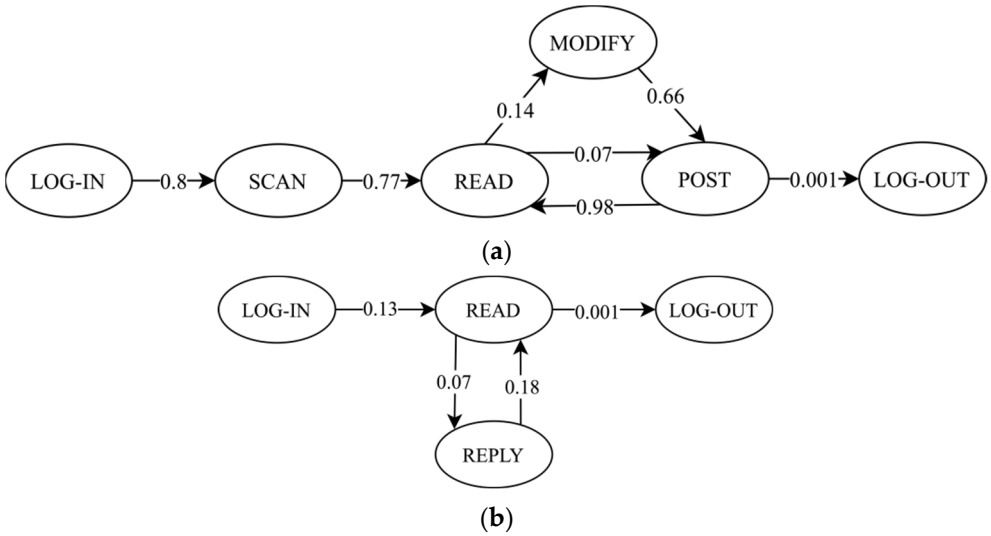

Figure 16. Examples of user action sequences. (**a**) The transitions between SCAN, READ, MODIFY and POST. (**b**) Another possible transition between READ and REPLY.

A possible user action sequence is shown in Figure 16a. A user logs in, browses, and posts a new blog or revises blogs he/she posted before. Figure 16b represents another possible user action sequence. A user reads several blogs and leaves comments under what he/she is interested in. Transition probabilities of stating LOG-OUT are very low (only 0.1%), which confirms that few users exit the service actively. Other than states shown in this figure, PREVIEW is a state whose related transition probabilities is very low (only about 0.2%).

We also construct user roles for guests. Due to the access control strategy of the forum, there are only SCAN and READ states in their role model.

7.4. Web Application Uncovering

In this part, we applied the proposed action identification methods to detect web applications. Volunteers are recruited to play different user roles, respectively, i.e., registered users, guests, and those who do not access the forum.

The registered users log in, read, and leave comments under posts in the forum. While the guests scan the forum and read several posts that they are interested in. They stay on the forum, capture the network traffic and take notes on what actions they have performed. The other volunteers visit the Alexa Top-50 sites' homepages, collect traffic data and label them with the corresponding web applications.

The traffic traces are mixed together and segmented with a time interval threshold. Inspired by [43], we utilized a smaller threshold (i.e., 2.5 s) to segment network traffic into a sequence of short time bursts. These bursts can be identified with the approach we have proposed in Section 4. In this step, traffic generated by visiting Alexa Top sites can be identified as unknown actions. Transitions between SCAN and READ are uncovered; however, both registered users and guests can perform this kind of transition. We can also identify transitions between READ and REPLY, which only exist in the constructed role model for registered users. This indicates the existence of a targeted web application.

8. Discussion and Limitations

8.1. Discussion

In this work, we address the users' action and web application identification problems based on PHMM since it exploits positional information contained in network traffic sequences and is sensitive to webpage transitional information. In order to be recognized by PHMM, the network traffic sequences are converted into symbolized sequences. We proposed an adaptive symbolization method, which takes packet distribution into consideration and outperforms *K*-means and equal interval methods in terms of the distance between the symbolized and original sequence. It is needed to be mentioned that all the results presented in the tables and graphs above are our top-ranked results.

8.2. Limitations

8.2.1. Packet Padding

Our method relies on packet information such as packet size, direction, and timestamps. Thus packet padding is the most direct technique to confuse our classifier. Opponents can add dummy packets to the normal network traffic to hide the user action patterns and make the classification more difficult. Luckily, due to the high bandwidth overhead, packet padding is not widely deployed yet.

8.2.2. Lack of Sufficient Prior Knowledge

In this work, we assume the user types are known in advance. User role models are built from the access log of a forum to help us uncover more information from network traffic. However, in the real world, it is not very easy to get various users' clickstream data (e.g., a website's access log) in advance and build the role models. User roles can serve as a complementary means for web application identification. We consider identifying different users' actions and try to build user role models from online network traffic in future work.

8.2.3. New Trends in Web Development

Due to the convenience of our work (the authors provide technical support for China's public security bureaus), we have learned that most fraud websites targeted by police officers are still developed traditionally. Thus we referred to a blog application as the practical solution of a Web application in this paper. Traditionally, a website is built as a collection of interlinked pages. When users follow hyperlinks to carry out their actions on the site, a browser loads the new page as a completely new entity, including the HTML and the resources that the HTML loads, such as CSS and JavaScript. This generates network traffic that contains transitional information. Since PHMM exploits positional information contained in network traffic sequences and is sensitive to webpage transitional information, we use it to build users' action patterns.

However, as the Internet has evolved, so have web development trends. Single-page applications (SPAs) are JavaScript-based web applications that load a single HTML page in users' browsers and dynamically update content as needed without refreshing the page. Progressive web applications (PWAs) help websites load in no time and work offline in apps to improve user experience. In the future, we will try to construct user–application interaction patterns from these types of Web applications.

9. Conclusions

In this paper, we propose an adaptive symbolization algorithm to represent network traffic and use the profile-hidden Markov model to construct user action patterns. User role models are built to describe different user types and utilized to identify web applications together with PHMM patterns. Compared with previous work, our equal probability symbolization method takes packet distribution information of original sequences into consideration. It is less time-consuming and reduces traffic sequence complexity while retaining information. The results demonstrate that our PHMM-based user action identification method has higher accuracy than the existing traditional classifiers. Different user types are first considered in web application identification, which helps us mine more detailed user information (e.g., users' actions and user types).

In the future, we will investigate the possibility of using online network traffic to build different user role models and uncover more information during web application identification. We will also carry out a more in-depth study of new types of web applications, such as SPAs and PWAs mentioned in Section 8.

Author Contributions: Methodology and writing—original draft preparation, X.Y. and J.X.; software and writing—review and editing, G.L., J.X. and X.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is mainly supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LGF21G030001, No. LGF21F020006, and No. LGF20G030001.

Data Availability Statement: The data and code are available from the corresponding authors upon reasonable request.

Acknowledgments: The authors would like to thank the anonymous reviewers for their helpful suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ionescu, P.; Keirstead, J.; Onut, I.; Wilson, D. Automatic Traffic Classification of Web Applications and Services Based on Dynamic analysis. U.S. Patent No. 10,542,025, 21 January 2020.
2. Tayor, V.F.; Conti, R.; Martinovic, I. Appscanner: Automatic fingerprinting of smartphone Apps from encrypted network traffic. In Proceedings of the 1st IEEE European Symposium on Security and Privacy, Saarbruecken, Germany, 21–24 March 2016; pp. 439–454.
3. Faik, A.H.; Jasleen, K. Can Android applications be identified using only TCP/IP headers of their launch time traffic. In Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks, Darmstadt, Germany, 18–20 July 2016; pp. 61–66.
4. van Ede, T.; Bortolameotti, R.; Continella, A.; Ren, J.; Dubois, D.J.; Lindorfer, M.; Choffnes, D.; van Steen, M.; Peter, A. FLOWPRINT: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In Proceedings of the 27th Network and Distributed Systems Security (NDSS) Symposium, San Diego, CA, USA, 23–26 February 2020; pp. 1–18.
5. Wang, T.; Cai, X.; Nithyanand, R.; Johnson, R.; Goldberg, I. Effective attacks and provable defenses for website fingerprinting. In Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, 20–22 August 2014; pp. 143–157.
6. Shen, M.; Liu, Y.; Zhu, L.; Du, X.; Hu, J. Fine-Grained Webpage Fingerprinting Using Only Packet Length Information of Encrypted Traffic. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 2046–2059. [CrossRef]
7. Sirinam, P.; Imani, M.; Juarez, M.; Wright, M. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In Proceedings of the ACM Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 1928–1943.
8. Wix. Available online: https://www.wix.com/ (accessed on 1 March 2021).
9. Sina News Report. Available online: https://tech.sina.com.cn/i/2018-10-16/doc-ihmhafir8971738.shtml (accessed on 1 March 2021).
10. Gorski, T. The k+1 symmetric test pattern for smart constracts. *Symmetry* **2022**, *14*, 1686. [CrossRef]
11. Eddy, S.R. Profile hidden Markov models. *Bioinformatics* **1998**, *14*, 755–763. [CrossRef] [PubMed]
12. Zhuo, Z.; Zhang, Y.; Zhang, Z.-L.; Zhang, X.; Zhang, J. Website Fingerprinting Attack on Anonymity Networks Based on Profile Hidden Markov Model. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 1081–1095. [CrossRef]
13. He, G.; Yang, M.; Luo, J.; Gu, X. A novel application classification attack against Tor. *Concurr. Comput. Pract. Exp.* **2015**, *27*, 5640–5661. [CrossRef]

14. Boukhlif, M.; Hanine, M.; Kharmoum, N. A Decade of Intelligent Software Testing Research: A Bibliometric Analysis. *Electronics* **2023**, *12*, 2109. [CrossRef]
15. Cheng, H.; Avnur, R. *Traffic Analysis of SSL Encrypted Web Browsing*; University of Berkeley: Berkeley, CA, USA, 1998; pp. 1–12.
16. Cai, X.; Zhang, X.; Joshi, B.; Johnson, R. Touching from a distance: Website fingerprinting attacks and defenses. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, Raleigh, NC, USA, 16–18 October 2012; pp. 605–616.
17. Hayes, J.; Danezis, G. K-fingerprinting: A robust scalable website fingerprinting technique. In Proceedings of the 25th USENIX Security Symposium, Austin, TX, USA, 10–12 August 2016; pp. 1187–1203.
18. Conti, M.; Li, Q.Q.; Maragno, A.; Spolaor, R. The Dark Side(-Channel) of Mobile Devices: A Survey on Network Traffic Analysis. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 2658–2713. [CrossRef]
19. Conti, M.; Mancini, L.V.; Spolaor, R.; Verde, N.V. Analyzing Android encrypted network traffic to identify user actions. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 1556–6021. [CrossRef]
20. Fu, Y.; Xiong, H.; Lu, X.; Yang, J.; Chen, C. Service Usage Classification with Encrypted Internet Traffic in Mobile Messaging Apps. *IEEE Trans. Mob. Comput.* **2016**, *15*, 2851–2864. [CrossRef]
21. Trimananda, R.; Varmaken, J.; Markopoulou, A. Packet-level signatures for smart home devices. In Proceedings of the 26th Network and Distributed System Security Symposium, San Diego, CA, USA, 24–27 February 2019; pp. 1–18.
22. Zhang, W.; Meng, Y.; Liu, Y. Homonit: Monitoring smart home apps from encrypted traffic. In Proceedings of the 25th ACM SIGSAC Conference on Computer and Communications, Toronto, ON, Canada, 15 October 2018; pp. 1074–1088.
23. Li, Q.; Feng, X.; Wang, R.; Li, Z.; Sun, L. Towards fine-grained fingerprinting of firmware in online embedded devices. In Proceedings of the IEEE conferences on computer communications, Honolulu, HI, USA, 16–19 April 2018; pp. 2537–2545.
24. Akhilesh, R.; Bills, O.; Chilamkurti, N.; Chowdhury, M.J.M. Automated Penetration Testing Framework for Smart-Home-Based IoT Devices. *Futur. Internet* **2022**, *14*, 276. [CrossRef]
25. Stober, T.; Frank, M.; Schmitt, J.; Martinovic, I. Who do you sync you are? smartphone fingerprinting via application behavior. In Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks, Budapest, Hungary, 17–19 April 2013; pp. 7–12.
26. Verde, N.V.; Ateniese, G.; Gabrielli, E.; Mancini, L.V.; Spognardi, A. No NAT'd User left behind: Fingerprinting users behind NAT from NetFlow records alone. In Proceedings of the IEEE 34th International Conference on Distributed Computing Systems, Madrid, Spain, 30 June–3 July 2014; pp. 218–227.
27. Krogh, A.; Brown, M.; Mian, I.S.; Sjolander, K.; Haussler, D. Hidden Markov Models in computational biology: Applications to protein modeling. *J. Mol. Biol.* **1994**, *235*, 1501–1531. [CrossRef] [PubMed]
28. Edgar, R.C.; Batzoglou, S. Multiple sequence alignment. *Curr. Opin. Struct. Biol.* **2006**, *16*, 368–373. [CrossRef] [PubMed]
29. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [CrossRef]
30. Bhargava, A.; Kondrak, G. Multiple word alignment with profile hidden Markov models. In Proceedings of the NAACL HLT Student Research Workshop and Doctoral Consortium, Boulder, CO, USA, 31 May–5 June 2009; pp. 43–48.
31. Clustal. Available online: http://www.clustal.org/omega/ (accessed on 1 June 2021).
32. Eddy, S.R.; The HMMER Development Team. *HMMER User's Guide: Biological Sequence Analysis Using Profile Hidden Markov Models*; Howard Hughes Medical Institute: Chevy Chase, MD, USA, 2018; pp. 1–227.
33. The Plan 7 Architecture. Available online: http://www.csb.yale.edu/userguides/seq/hmmer/docs/node11.html (accessed on 1 June 2021).
34. Daw, C.S.; Finney, C.E.A.; Tracy, E.R. A review of symbolic analysis of experimental data. *Rev. Sci. Instrum.* **2003**, *74*, 915–930. [CrossRef]
35. Shen, L.; Cheng, L.; Ford, J. Mining the most interesting web access associations. In Proceedings of the World Conference on the WWW and Internet, San Antonio, TX, USA, 30 October–4 November 2000; pp. 1–6.
36. Guzduz, S.; Ozsu, M.T. A web page prediction model based on clickstream tree representation of user behavior. In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 October 2003; pp. 535–540.
37. Lu, L.; Dunham, M.H.; Meng, Y. Mining significant usage patterns from clickstream data. In *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4198, pp. 1–17.
38. Obendorf, H.; Weinreich, H.; Herder, E.; Mayer, M. Web page revisitation revisited: Implications of a long-term clickstream study of browser usage. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, New York, NY, USA, 28 April–3 May 2007; pp. 597–606.
39. Sadagopan, N.; Li, J. Characterizing typical and atypical user sessions in clickstreams. In Proceedings of the 17th International Conference on World Wide Web, Beijing, China, 21–25 April 2008; pp. 885–894.
40. Wang, G.; Konolige, T.; Wilson, C.; Wang, X.; Zheng, H.; Zhao, B. You are how you click: Clickstream analysis for sybil detection. In Proceedings of the 22nd USENIX Security Symposium, Washington, DC, USA, 14–16 August 2013; pp. 241–255.
41. Benevenuto, F.; Rodrigues, T.; Cha, M.; Almeida, V. Characterizing user behavior in online social networks. In Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement, New York, NY, USA, 4−6 November 2009; pp. 49–62.

42. Taylor, V.F.; Spolaor, R.; Conti, M.; Martinovic, I. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 63–78. [CrossRef]
43. Yan, F.P. Technology of Identifying Wechat User Usage Based on Network Traffic. Master's Thesis, Hangzhou Dianzi University, Hangzhou, China, 2020.