IEICE TRANS. INF. & SYST., VOL.E98–D, NO.4 APRIL 2015

PAPER *Special Section on Information and Communication System Security—Against Cyberattacks—*

A New Approach to Identify User Authentication Methods toward SSH Dictionary Attack Detection

Akihiro SATOH[†a)], **Yutaka NAKAMURA**[†], ***and*** **Takeshi IKENAGA**[†], ***Members***

SUMMARY A dictionary attack against SSH is a common security threat. Many methods rely on network traffic to detect SSH dictionary attacks because the connections of remote login, file transfer, and TCP/IP forwarding are visibly distinct from those of attacks. However, these methods incorrectly judge the connections of automated operation tasks as those of attacks due to their mutual similarities. In this paper, we propose a new approach to identify user authentication methods on SSH connections and to remove connections that employ non-keystroke based authentication. This approach is based on two perspectives: (1) an SSH dictionary attack targets a host that provides keystroke based authentication; and (2) automated tasks through SSH need to support non-keystroke based authentication. Keystroke based authentication relies on a character string that is input by a human; in contrast, non-keystroke based authentication relies on information other than a character string. We evaluated the effectiveness of our approach through experiments on real network traffic at the edges in four campus networks, and the experimental results showed that our approach provides high identification accuracy with only a few errors.

key words: *SSH dictionary attack, user authentication method, flow analysis, network operation*

1. Introduction

Secure Shell (SSH) is one of the most important protocols in network operations. SSH provides administrators with various functions, such as remote login, file transfer, and TCP/IP forwarding [1]. In addition, the protocol assists in automating operation tasks. Examples of automated tasks are remote execution of interactive and batch jobs [2], [3], backup of valuable data from distributed hosts [4], [5], and collection of logging messages via the Internet [6], [7].

The SysAdmin, Audit, Network, Security (SANS) Institute [8], established in 1989 as a cooperative research and education organization, has warned about dictionary attacks against SSH. A dictionary attack is a login attempt to gain fraudulent access by guessing a username and password pair. Since even one successful attack causes serious problems, administrators must be prepared to cope with all attacks.

Many published methods [9]–[12] rely on network traffic to detect SSH dictionary attacks. This is because the connections of remote login, file transfer, and TCP/IP forwarding are visibly distinct from those of attacks; however, these methods do not deal with automated tasks through SSH. Unfortunately, these methods incorrectly judge the connections of automated tasks as those of attacks due to their mutual similarities.

In this paper, we propose a new approach to identify user authentication methods on SSH connections and to remove connections that employ non-keystroke based authentication. This approach is based on two perspectives: (1) an SSH dictionary attack targets a host that provides keystroke based authentication; and (2) automated tasks through SSH need to support non-keystroke based authentication. Keystroke based authentication relies on a character string that is input by a human; in contrast, non-keystroke based authentication relies on information other than a character string. For example, keystroke based authentication includes password and challenge-response functions, and non-keystroke based authentication includes public-key and host-based functions. However, the confidentiality and flexibility of the SSH protocol interfere with identification. We resolve these problems by two key innovations: (1) use of flow behaviors for identification; and (2) consideration of reference points for flow behaviors.

We evaluated the effectiveness of our approach through experiments on real network traffic at the edges in four campus networks, and the experimental results showed that our approach provides high identification accuracy with only a few errors. Our significant contribution is improved ways for detecting SSH dictionary attacks.

This paper is organized as follows. Section 2 describes the SSH protocol and summarizes the limitations of related work on SSH dictionary attacks. Our findings from the analyses of SSH connections at the flow level are given in Sect. 3. On the basis of these analytical results, a new approach is proposed in Sect. 4, and our proposal is evaluated in Sect. 5. We conclude our paper and state future work in Sect. 6.

2. Background

In this section, we describe the details of SSH protocol specifications and user authentication methods. We then discuss related work and their limitations regarding SSH dictionary attacks.

2.1 SSH Protocol Specification

An SSH handshake consists of three major sub-protocols: transport layer, user authentication, and connection protocols [13]–[15].

Manuscript received May 21, 2014.
Manuscript revised October 1, 2014.
Manuscript publicized December 4, 2014.
[†]The authors are with Kyushu Institute of Technology, Kitakyushu-shi, 804–8550 Japan.
a) E-mail: satoh@isc.kyutech.ac.jp
DOI: 10.1587/transinf.2014ICP0005

Table 1 Types of user authentication methods, cipher algorithms, MAC algorithms, and compression algorithms.

User Auth. Method	password, challenge-response, public-key (rsa, dsa, ecdsa), host-based (rsa, dsa, ecdsa)
Cipher Algorithm	aes128-ctr, aes192-ctr, aes256-ctr, arcfour256, arcfour128, aes128-cbc, 3des-cbc, blowfish-cbc, cast128-cbc, aes192-cbc, aes256-cbc, arcfour, rijndael-cbc@lysator.liu.se
MAC Algorithm	hmac-md5, hmac-sha1, hmac-md5-96, hmac-sha1-96, hmac-ripemd160, umac-64@openssh.com, hmac-ripemd160@openssh.com
Compression Algorithm	none, zlib

The transport layer protocol negotiates cipher, message authentication code (MAC), and compression algorithms to establish a secure connection between a client and a server. For example, as shown in Table 1, cipher algorithms include aes-cbc and 3des-cbc, and MAC algorithms include hmac-md5 and hmac-sha1. Note that the cipher, MAC, and compression algorithms are immediately applied after finishing the transport layer protocol. Then, the SSH handshake shifts to the user authentication protocol. The user authentication protocol assumes responsibility for authenticating the client to the server by the user authentication method described in Sect. 2.2. Finally, the connection protocol performs various functions such as remote login, file transfer, TCP/IP forwarding, and so on.

The SSH protocol has two notable properties: confidentiality and flexibility. Confidentiality means establishing a secure connection by cipher, MAC, and compression algorithms. Flexibility means varying these algorithms in accordance with circumstances. For example, these algorithms are independently negotiated for each host, so each host chooses its own algorithms.

2.2 User Authentication Method

The user authentication method determines whether a user should be allowed to establish a connection via SSH. OpenSSH [16] and Tectia [17] are general implementations of SSH. The implementations employ four user authentication methods: (1) password, (2) challenge-response, (3) public-key, and (4) host-based. Password authentication is a traditional method. During this authentication, a client transmits a password to a server over an encrypted connection, and then the server checks that the given password is acceptable for the target user. In addition to establishing a secure connection, challenge-response authentication encrypts the password itself to prevent man-in-the-middle attacks [18]. Public-key authentication verifies the user's identity by holding the private counterpart of an authorized public key. Host-based authentication allows a connection between hosts without validation if both hosts have a trust relationship with each other.

User authentication methods are categorized as two types: keystroke and non-keystroke. Keystroke based authentication relies on a character string that is input by a human. This authentication is employed for remote login, file transfer, and TCP/IP forwarding. In contrast, non-keystroke based authentication relies on information other than a character string to avoid prompting by a human. This authentication enables administrators to automate operation tasks through SSH. Specifically, keystroke based authentication includes password and challenge-response functions; non-keystroke based includes public-key and host-based functions.

2.3 Related Work and Limitations

The activities of SSH dictionary attacks were explored in [19], [20]. On the basis of these reports, numerous studies have been published, and their methods have been categorized into two types.

The first type focused on the importance of log files that record login attempts. Many tools [21]–[25] could read through log files and keep track of unsuccessful login attempts against SSH servers. Further connections from a client were denied by dynamically adding a rule, if the number of unsuccessful login attempts exceeded a pre-defined threshold. Thames et al. [26] outlined a new architecture for preventing SSH dictionary attacks. In this architecture, trustworthy servers gathered, analyzed, and distributed information about malicious clients through collaboration. However, these methods do not scale well in large networks. Since these methods must be applied on all hosts in each large network, they impose heavy maintenance costs on the administrators.

The requirement of the second type is to capture network traffic at only a few observation points and thereby limits the above costs. Vykopal et al. [9] deemed a rapid increase in connections to an SSH server to be dictionary attacks, and blocked the connections. Hellemons et al. [10] showed that such attacks typically consist of three phases, and the authors represented their behaviors at the flow level by using a hidden Markov model. Takemori et al. [11] discovered a significant upsurge in the number of pointer (PTR) resource records in DNS traffic while attacks were underway. To resolve the problems of these studies, we proposed a new approach [12] that detects dictionary attacks and their success or failure. This approach relied on two perspectives: (1) for an SSH connection, a username and password pair is manually entered through the user's keystrokes, while for an SSH dictionary attack, the pair is automatically entered by one's dictionary; and (2) for a successful attack, the connection protocol appears in the flow, while for an unsuccessful attack, this protocol does not appear.

These studies demonstrated that the connections of remote login, file transfer, and TCP/IP forwarding are visibly distinct from those of dictionary attacks; however, they did not account for automated tasks through SSH. The connections of automated tasks have the following similarities with those of attacks: (1) massive and brief connections are observed during a short period; and (2) an unattended authentication process is automatically run on each connection. The similarities give rise to the detection errors of dictionary attacks. Specifically, our previous work [27] reported this problem through some experiments; the technique of Javed et al. [28] ignored the connections of automated tasks based on the logging messages of SSH daemon. It should be pointed out that for traffic-based methods no studies regarding this problem have appeared in the literature.

3. Analysis

The challenge in this paper is to exclude the connections of automated tasks, which are non-keystroke based and therefore erroneously detected as dictionary attacks, from all connections. Figure 1 shows the overview of our proposal with dictionary attack detection. Our approach first identifies the user authentication methods on the connections, and then removes the traffic of connections that employ non-keystroke based authentication. Dictionary attacks are finally detected from only the connections of keystroke based authentication. Note that one of the advantages is the ability to replace the detection function with a new one.

However, the confidentiality and flexibility of the SSH protocol interfere with identification. Confidentiality is traditionally employed to establish a secure connection by cipher, MAC, and compression algorithms and to avoid direct packet inspection. Our first key innovation is to use flow behaviors [29] for identifying user authentication methods. A flow consists of bi-directional packet exchanges between a client and a server with the same five-tuple, where the five-tuple is the source IP address, destination IP address, source port number, destination port number, and protocol number. Its behaviors are statistical patterns in terms of, for example, packet size, packet direction, and packet order, in externally observable packets taken from a flow. The reasons for using flow behaviors are as follows: (1) flow behaviors are observable without direct packet inspection; and (2) flow behaviors differ depending on the type of user authentication methods. Flexibility is employed by varying the cipher, MAC, and compression algorithms in accordance with the circumstances. These algorithms affect flow behaviors. Our second key innovation is to consider reference points, which can mitigate the impact of these algorithms on flow behaviors.

In this paper, we extend our previous work [30] in several ways. First, we provide a framework for our proposal by formalizing it as pattern-recognition problem. Second, we expand the tests to verify its effectiveness in detecting dictionary attacks, and we also discuss contributions other than detection.

The first part of this section deals with the datasets used in our analyses. In the second part we analyze flow behaviors and verify their effectiveness for identifying user authentication methods. On the basis of the analytical results, our approach is then proposed in Sect. 4.

3.1 Analysis Datasets

Table 2 summarizes the two analysis datasets. The datasets were captured at a gigabit ethernet link between two hosts running OpenSSH. We manually generated both $D1$ and $D2$ by combinations of all user authentication methods, cipher algorithms, MAC algorithms, and compression algorithms shown in Table 1. The difference between $D1$ and $D2$ was their success or failure in authentication.

For pre-processing in the analyses, we first extracted SSH flows with the same five-tuple from each dataset. Second, the flows were given corresponding labels in accordance with their user authentication methods. Finally, we removed TCP control packets (e.g., SYN, FIN, or ACK flags with no payload) from the flows because such packet exchanges are application-independent.

3.2 Analysis on Packet Exchanges in Each Sub-Protocol

As a first step towards identifying user authentication methods, we analyze packet exchanges in each sub-protocol.

First, we randomly chose either $D1$ or $D2$ as the type of flow. Then we investigated all packets in each flow and clarified the relations between packet size, packet direction, and sub-protocol with respect to sequence number. To visualize these flows, we adopted the technique of Wright et al. [31].

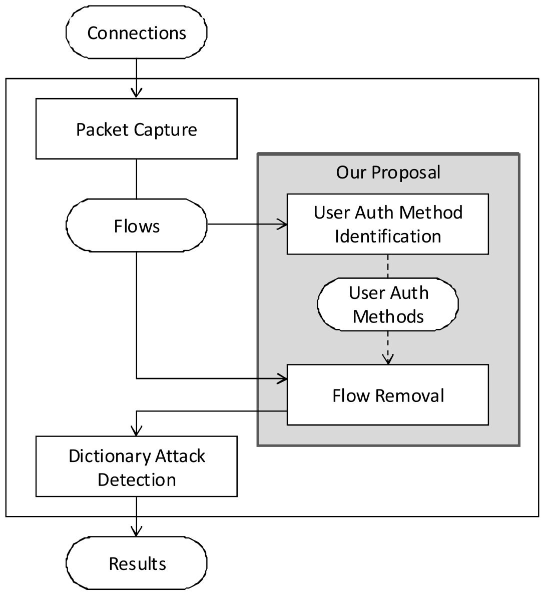
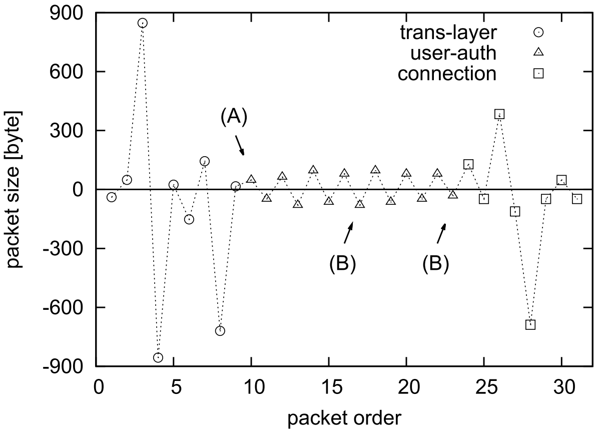


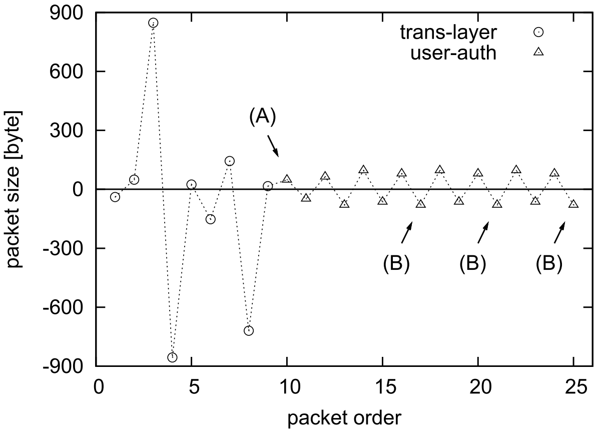
Fig. 1 Overview of our proposal with SSH dictionary attack detection.

Table 2 Number of flows in analysis datasets.

	password	chal-resp	public-key	host-based
$D1$	182	182	546	546
$D2$	182	182	546	546



(a) Success of challenge-response authentication



(b) Failure of challenge-response authentication

Fig. 2 Packet exchanges in each sub-protocol.

The results are shown in Fig. 2. Each result is from a typical instance of the flows. A packet is represented as a symbol, and the symbol type indicates its sub-protocol. The X-coordinate corresponds to the sequence number in the individual flow, and the Y-coordinate corresponds to the packet size and the packet direction. Positive and negative Y values are the "incoming" and "outgoing" directions, respectively, where "incoming" means transmission from a client to a server and "outgoing" means transmission from a server to a client. In either case, the magnitude of the Y-coordinate gives the packet size in bytes. For example, a symbol at $(10, -500)$ means that an outgoing packet that is 500 bytes in length was the 10-th to arrive after the start of a flow. Furthermore, the arrows and their labels denote the following: (A) the initial applied packet of cipher, MAC, and compression algorithms, and (B) the notified packet of an authentication result. Note that the multiple notified packets denote authentication failure.

Figure 2 (a) shows that the notified packet of authentication success appears before shifting to the next sub-protocol, and Fig. 2 (b) shows that the last notified packet of authentication failure appears before finishing the flow. In addition, these results indicate that the cipher, MAC, and compression algorithms are immediately applied after finishing the transport layer protocol and that the user authentication protocol repeats the request and response packets.

3.3 Analysis on Flow Behaviors of Each User Authentication Method

As the next step, we clarify the difference in the flow behaviors of each user authentication method.

We define two terms: sub-flow and delta length. First, a sub-flow is some consecutive packets in a flow. The sub-flow $x_{i:j}$, taken from the i-th to j-th packets, is represented by the $(j - i + 1)$-tuple:

$$x_{i:j} = [\ p_i(x), p_{i+1}(x), \cdots, p_j(x)\ ].$$

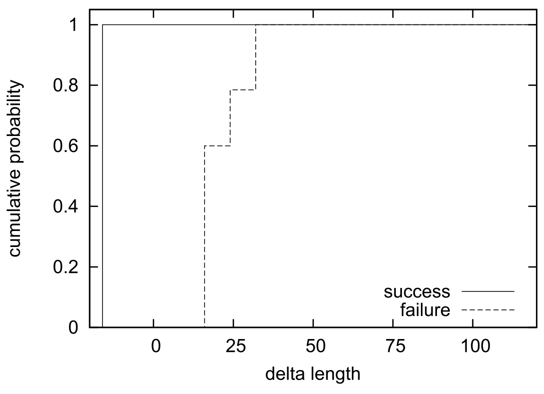
Here $p_i(x)$ and $p_j(x)$ denote the i-th and j-th packets in flow x. Second, delta length is one of the newly defined flow behaviors. The delta length $\delta_i(x)$ of the i-th packet in flow x is written in the following equation:

$$\delta_i(x) = \begin{cases} s_i(x) - s_\mu(x) & (d_i(x) = \text{incoming}), \\ s_i(x) - s_{\mu+1}(x) & (d_i(x) = \text{outgoing}). \end{cases}$$

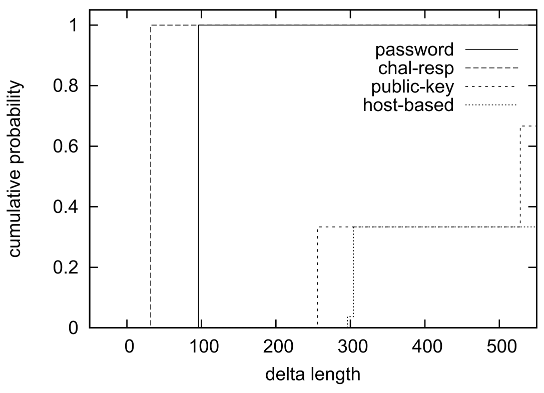
Here, $s_i(x)$ and $d_i(x)$ denote the i-th packet size and its direction, respectively. The μ-th and $(\mu + 1)$-th packets are the initial request and response in the user authentication protocol, respectively. We consider the μ-th and $(\mu + 1)$-th packets as reference points that can mitigate the impact of each algorithm on the flow behaviors. The rationale is as follows: both i-th and μ-th packets contain the same fields added by the cipher, MAC, and compression algorithms; the payload of the μ-th packet, expected to have a field added by these algorithms, has a fixed length in all flows because its remaining field is a message about the start of the user authentication protocol; thus, the difference between these two packets convey the message contained in the i-th packet.

The analytical procedure is as follows. First, all sub-flows $x_{K-2:K}$ located at the end of the user authentication protocol are selected in $D1$ and $D2$, where the constant K means the sequence number of the last notified packet in each flow. Let us assume, for example, that sub-flows, $x_{21:23}$ and $x_{23:25}$, can be found in Figs. 2 (a) and 2 (b), respectively. Then, the delta length of each packet in the sub-flows is visualized for comparison of the user authentication methods by cumulative probability distributions.

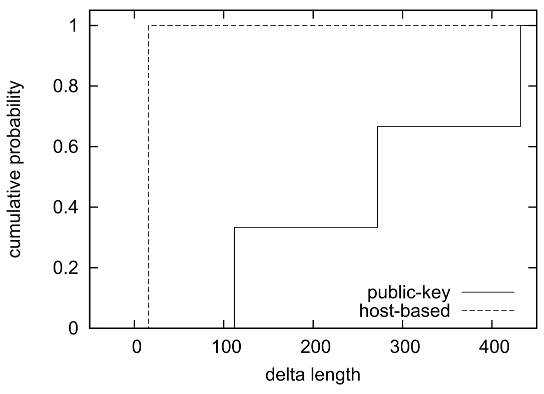
These distributions are shown in Fig. 3, where the X-coordinate corresponds to the delta length of each packet, and the Y-coordinate corresponds to the cumulative probability. In Fig. 3 (a), $\delta_K(x)$ has a value of -16 when the authentication is a success; $\delta_K(x)$ has a value between 16 and 32 when the authentication is a failure. It is evident from Fig. 3 (a) that the delta length of the last packet depends only on its authentication result. In Fig. 3 (b), $\delta_{K-1}(x)$ of password and challenge-response authentication methods is 96 and 32, respectively, whereas $\delta_{K-1}(x)$ of public-key and host-based overlaps in the range from 256 to 640. In Fig. 3 (c), $\delta_{K-2}(x)$ of public-key ranges from 112 to 432, while $\delta_{K-2}(x)$ of host-based has a value of 16. As shown



(a) Last packets in sub-flows



(b) Middle packets in sub-flows



(c) First packets in sub-flows

Fig. 3 Cumulative probability distributions of delta length of each packet.

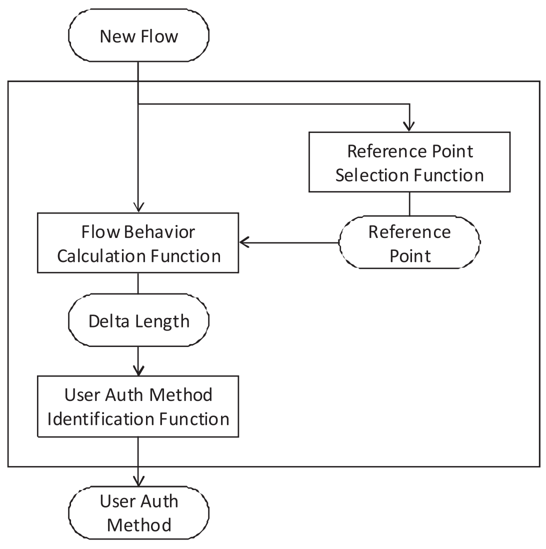

New Flow
Reference Point Selection Function
Flow Behavior Calculation Function
Reference Point
Delta Length
User Auth Method Identification Function
User Auth Method

Fig. 4 Overview of our approach to identify user authentication methods on SSH connections.

in Figs. 3 (b) and 3 (c), these results indicate that the delta length of the first and middle packets in the sub-flow, located at the end of the user authentication protocol, are effective for identifying the user authentication methods.

3.4 Discussion

The analytical results indicated that the delta length has the ability to identify the user authentication methods employed in SSH connections. Two types of packets play an important role in identification. One is the initial applied packet of cipher, MAC, and compression algorithms. This packet is necessary for calculating the delta length. An effective means for finding this packet from a flow would be a direct payload inspection, because its previous packets are not encrypted. The other type is the last notified packet of an authentication result. This packet is necessary for determining the sub-flow located at the end of the user authentication protocol. Our analysis clarified the following consideration that assists in finding this packet: (1) the delta length derived from this packet depends only on its authentication result; (2) the exchanges after accepting this packet vary between authentication success and failure; (3) the two packets in front of this packet are the incoming and outgoing directions.

4. Proposal

We already illustrated the overview of our proposal in Fig. 1. The first part of our proposal identifies user authentication methods on SSH connections, and the second part removes connections if their user authentication methods are deemed as non-keystroke. In this section, we address realization of the first part based on the analytical results. The part consists of the three functions given in Fig. 4: (1) reference point selection; (2) flow behavior calculation; and (3) user authentication method identification.

4.1 Reference Point Selection Function

This function inspects the payload of each packet in new flow x to find the initial applied packet of cipher, MAC, and compression algorithms. Then, two packets next to the applied packet are selected as reference points. Specifically, the reference points are $p_{\mu+1}(x)$ and $p_\mu(x)$ when meeting both of the following conditions: (1) $c_{\mu+1}(x)$ and $c_\mu(x)$ are true; (2) $c_{\mu-1}(x), c_{\mu-2}(x), \cdots$, and $c_1(x)$ are false. Here, $p_\mu(x)$ is the μ-th packet in flow x, and $c_\mu(x)$ indicates whether the μ-th packet is encrypted by SSH.

4.2 Flow Behavior Calculation Function

This function has three steps: the first is to detect sub-flows based on packet direction in flow x; the second is to determine the sub-flow located at the end of the user authentication protocol; the third is to calculate the delta length of initial and middle packets of the sub-flow to identify the user authentication method employed in flow x.

In the first step, this function extracts a repetition of request and response by investigating the direction of each packet from the initial point of the user authentication protocol: $x_{i-2:i}$ is detected when the repetition is $d_i(x) = \mathit{outgoing}$, $d_{i-1}(x) = \mathit{incoming}$, and $d_{i-2}(x) = \mathit{outgoing}$. Here, $d_i(x)$ and $x_{i-2:i}$ denote the i-th packet direction and the sub-flow, respectively. The sub-flow $x_{i-2:i}$ consists of three consecutive packets from $(i-2)$-th to i-th.

Next, this function determines whether the sub-flow is located at the end of the user authentication protocol. This function proceeds to the last step when $\delta_i(x)$ meets either condition: (1) $\delta_i(x) \in \mathbb{Z}_s$; or (2) $\delta_i(x) \in \mathbb{Z}_f$ and $\forall j\ \neg x_{j-2:j}\ \ s.t.\ \ i < j$. Here, $\delta_i(x)$ is the delta length of the last packet in sub-flow $x_{i-2:i}$; $\mathbb{Z}_s$ and $\mathbb{Z}_f$ are the range of $\delta_i(x)$ in the case of an authentication success and failure, respectively. Simply put, the first condition is used for selecting the sub-flow that includes the notified packet of an authentication success; the second is used for selecting the sub-flow that includes the notified packet of an authentication failure and is the last in the flow.

Finally, this function calculates the delta length from the first and middle packets in sub-flow $x_{i-2:i}$. These values, $\delta_{i-1}(x)$ and $\delta_{i-2}(x)$, are input to the next function for identifying the user authentication method employed in flow x.

4.3 User Authentication Method Identification Function

This function identifies the user authentication method in flow x. We show the relationship between conditions and results in Table 3, where each set denotes the range of the delta length in each user authentication method. This function outputs the result corresponding to the condition in which $\delta_{i-1}(x)$ and $\delta_{i-2}(x)$ meet. For example, this function deems the method employed in flow x to be public-key if both $\delta_{i-1}(x) \in \mathbb{Z}_{hk}$ and $\delta_{i-2}(x) \in \mathbb{Z}_k$ are true. It should also be pointed out that an "unknown" result means a method that was not identifiable by our approach.

Table 3 Relationship between conditions and results.

Condition	Result
$\delta_{i-1}(x) \in \mathbb{Z}_p$	password
$\delta_{i-1}(x) \in \mathbb{Z}_c$	challenge-response
$\delta_{i-1}(x) \in \mathbb{Z}_{hk}$ and $\delta_{i-2}(x) \in \mathbb{Z}_k$	public-key
$\delta_{i-1}(x) \in \mathbb{Z}_{hk}$ and $\delta_{i-2}(x) \in \mathbb{Z}_h$	host-based
otherwise	unknown

5. Evaluation

In this section, we evaluate the effectiveness of our approach through experiments on six datasets. The evaluation points are mainly given for (1) the identification accuracy of user authentication methods, and (2) the detection accuracy of SSH dictionary attacks. The accuracy is calculated from two metrics: true positive and false positive.

5.1 Testing and Training Datasets

The four datasets in Table 4 were used as a "testing dataset" to assess the validity of our approach. $D3$, $D4$, $D5$, and $D6$ were the sets of flows captured at each edge in four campus networks. The prefix length of the first two networks was 16, and the prefix length of the other networks was 20. Each network connected to the Internet by a 10 Gbps link. The two datasets, $D1$ and $D2$ in Table 2, were used as a "training dataset" to determine the parameters relevant to our approach, and their details were described in Sect. 3.1. As expected, TCP control packets were removed from the training and testing datasets.

As shown in Table 4, the flows in each dataset are categorized according to two labels. The first label is the type of user authentication method employed in the flow, and the second is whether the flow is a dictionary attack. The two labels, corresponding to the individual flows, are determined by means of the following: (1) log file investigation at SSH servers, (2) collation of source and destination addresses in white and black lists, (3) scan of allowed user authentication methods on SSH servers [32], [33], and (4) comparison with known flows.

5.2 Metrics

A common way to characterize the identification and detection accuracy is through two metrics known as true positive and false positive [34]. True positive is the proportion of flows given correct labels, and false positive is the proportion of flows given incorrect labels. These metrics are described as follows:

$$\mathcal{T}(D_L) = \sum_{x \in D_L} \frac{\phi(f(x), L)}{|D_L|}\ ,\ \mathcal{F}(D_L) = \sum_{x \in \bar{D}_L} \frac{\phi(f(x), L)}{|\bar{D}_L|}\ .$$

Here, D_L and $\bar{D}_L$ are the set of flows given the label L and the set of the other flows in the testing dataset D, respectively. The absolute values of D_L and $\bar{D}_L$ are the number of elements in each set. As an example, for a label L of password authentication, $|D3_L|$ in Table 4 is 34013 from the sum of flows of the normal connections and dictionary attacks; and $|\bar{D3}_L|$ is 18285 from the between of $|D3|$ and $|D3_L|$. In addition, $f(x)$ denotes the predicted label of the flow x by our approach, and ϕ is an indicator function: $\phi(a, b) = 1$ if $a = b$, and $\phi(a, b) = 0$ if $a \neq b$.

Table 4 Number of flows in testing datasets.

	Normal Connection				Dictionary Attack		Total
	password	chal-resp	public-key	host-based	password	chal-resp	
*D*3	192	110	848	0	33821	17327	52298
*D*4	286	164	119	0	28005	12508	41082
*D*5	135	97	72	0	7818	3447	11569
*D*6	149	52	516	125	3594	963	5399

Table 5 Identification accuracy of user authentication methods.

	password		chal-resp		public-key		host-based	
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
*D*3	0.991	0.000	0.993	0.000	0.984	0.000	—	0.000
*D*4	0.988	0.000	0.995	0.000	0.924	0.000	—	0.000
*D*5	0.993	0.000	0.979	0.000	1.000	0.000	—	0.000
*D*6	0.983	0.000	0.985	0.000	0.996	0.000	0.976	0.000

Table 6 Number of flows in testing datasets after removal by our approach.

	Normal Connection				Dictionary Attack		Total
	password	chal-resp	public-key	host-based	password	chal-resp	
$D3^*$	192	110	13	0	33821	17327	51463
$D4^*$	286	164	9	0	28005	12508	40972
$D5^*$	135	97	0	0	7818	3447	11497
$D6^*$	149	52	2	3	3594	963	4763

5.3 Identification Accuracy of User Authentication Methods

The purpose of the experiments is to verify the identification performance of our approach. The experimental procedures consist of training and testing phases. In the training phase, the ranges of delta length of sub-flows for each user authentication method were derived from the analysis of the training datasets, and the ranges were set as the parameters of our approach: $\mathbb{Z}_s = [-16]$, $\mathbb{Z}_f = [16, 32]$, $\mathbb{Z}_p = [96]$, $\mathbb{Z}_c = [32]$, $\mathbb{Z}_{kh} = [256, 640]$, $\mathbb{Z}_k = [112, 432]$, and $\mathbb{Z}_h = [16]$. The details are described in Sects. 3.3 and 4.3. In the testing phase, the user authentication method of each flow in the testing datasets was identified by our approach, and the output was compared to the label given to the flow. Then, we quantified the comparison as true positive or false positive. Note that our approach output "unknown" when the user authentication method was not identifiable for a flow.

Table 5 shows the identification accuracy of user authentication methods. Although a few errors occurred, our approach kept the accuracy high even in the worst case of each user authentication method: (1) the true positives of password, challenge-response, public-key, and host-based authentication took values of 0.983, 0.979, 0.924, and 0.976, respectively; and (2) the false positives of all cases were equal to zero in the experiments because our approach recorded an unknown response instead of an incorrect label for the user authentication methods. The experimental results imply that our approach correctly removed more than 98% of the total flows of non-keystroke based authentication that are not due to dictionary attacks, and thus our approach represents a breakthrough in improving practical detection.

The causes of the errors were mainly in four categories. In the first category, the errors were attributed to packet loss and packet retransmission. By massive connections traversing an observation point, packet loss occurred in bursts at the capture machine, and this led to changes of flow behaviors. For this reason, a total of 727 flows were incorrectly identified. In the second category, the errors were attributed to padding flows. The SSH specification allows variable amounts of padding to be added to packets before encryption, and thus the padding option complicated the traffic analysis. For this reason, a total of 191 flows were incorrectly identified. In the third category, the errors were attributed to dictionary attacks against the old version of the SSH protocol. The old version had no sub-protocols, that is, no transport layer, user authentication, and connection subprotocols, and accordingly, the flow behaviors of the old version were quite different from those of the current version. For this reason, a total of 69 flows were incorrectly identified. In the fourth category, the cause of a few errors could not be specified. For this reason, a total of 17 flows were incorrectly identified.

5.4 Detection Accuracy of SSH Dictionary Attacks

The purpose of the experiments is to verify improvement of the detection performance by our approach. First, as described in Sect. 5.3, the flows of non-keystroke based authentication were removed from the testing datasets. Table 6 shows the testing datasets after removing the unnecessary flows. Then, dictionary attacks were detected by two implementations, based on the techniques of [9] and [12], in the testing datasets before and after the removal. The first implementation detected the aggregated flows, for which the number from one source exceeded a pre-defined threshold during a given time interval. Here, we set the threshold

Table 7 Detection accuracy of SSH dictionary attacks by the implementation of [9].

	Normal Connection		Dictionary Attack	
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
$D3$	0.331	0.024	0.975	0.668
$D4$	0.834	0.019	0.980	0.165
$D5$	1.000	0.009	0.990	0.000
$D6$	0.301	0.018	0.981	0.698

	Normal Connection		Dictionary Attack	
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
$D3^*$	1.000	0.024	0.975	0.000
$D4^*$	1.000	0.019	0.980	0.000
$D5^*$	1.000	0.009	0.990	0.000
$D6^*$	1.000	0.018	0.981	0.000

Table 8 Detection accuracy of SSH dictionary attacks by the implementation of [12].

	Normal Connection		Dictionary Attack	
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
$D3$	0.262	0.003	0.996	0.737
$D4$	0.790	0.005	0.994	0.209
$D5$	0.763	0.001	0.998	0.236
$D6$	0.238	0.000	1.000	0.761

	Normal Connection		Dictionary Attack	
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
$D3^*$	0.958	0.003	0.996	0.041
$D4^*$	0.980	0.005	0.994	0.019
$D5^*$	1.000	0.001	0.998	0.000
$D6^*$	0.975	0.000	1.000	0.024

and the time interval as 20 flows and 60 seconds, respectively. The second implementation detected the individual flows, for which the input time of a username and password pair exceeded a pre-defined threshold. Here, we set the input time as 1.5 seconds. The reason for selecting them is that major detection methods cover the two granularities of flows. Finally, the true positives and false positives were calculated from the detection results.

The detection accuracy by the first implementation is shown in Table 7. For the testing datasets before the removal of unnecessary flows, some detection errors occurred in the normal connections of $D3$, $D4$, and $D6$. The errors were caused by the flows of automated tasks, which employed non-keystroke based authentication; obviously, the reason for the true positive of 1.000 in $D5$ was that the dataset did not include the flows of automated tasks. Specifically, 1451 flows in total were incorrectly deemed to be dictionary attacks. The detection accuracy by the second implementation is shown in Table 8. For the testing datasets before the removal of unnecessary flows, the true positives of dictionary attacks in Table 8 were superior to those in Table 7 whereas the true positives of normal connections in Table 8 were inferior to those in Table 7. The decrease in the accuracy was attributed to not only the flows of automated tasks, but also all flows with non-keystroke based authentication. Specifically, 1680 flows in total were incorrectly deemed to be dictionary attacks. For the testing datasets after the removal in Tables 7 and 8, the true positives of normal connections exceeded 0.950 in all cases; the true positives of dictionary attacks remained the high values, which were equal to those before removing the unnecessary flows; thus our approach increased their detection accuracy without a drawback. Consequently, the experimental results show that our approach improves the performance of major detection methods against SSH dictionary attacks.

We then discuss the computational complexity. The complexity is negligible in theory because our approach outputs one identification result only by the following: (1) extraction of a few flow behaviors; and (2) comparison of several times. This supports the possibility of identifying user authentication methods on the fly. Thus our approach can be applied to real-time detection methods by assuming optimized implementation.

In addition to dictionary attack detection, our approach is useful in several situations. Dusi et al. [35] proposed a technique to detect SSH tunnels because they represented a significant security threat for any networks protected by firewalls. The authors lessened the problem of SSH tunnel detection by assuming that the user authentication method of an SSH connection was given. Thus, our approach will support practical SSH tunnel detection. Another example is controlling and blocking flows according to institutional policies. Our approach will contribute to realizing secure networks which only permit flows of public-key authentication.

6. Conclusion

In this paper, we proposed a new approach to identify user authentication methods by analyzing SSH connections at the flow level. This approach has two key innovations, the use of flow behaviors for identification and the consideration of reference points for flow behaviors, to resolve the problems caused by the confidentiality and flexibility of the SSH protocol. We evaluated the effectiveness of our approach through experiments on real network traffic at the edges in campus networks, and the experimental results showed that our approach provides high identification accuracy with only a few errors.

In our future work, we will experiment further on various SSH connections collected from enterprise or campus networks. From the results, we intend to continue improving our approach with the ultimate goal of being able to enable secure networks.

Acknowledgments

This work was supported in part by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C) (No.25330154).

References

[1] T. Ylonen, "SSH: Secure login connections over the Internet," Proc.

6th Conference on USENIX Security Symposium, Focusing on Applications of Cryptography, vol.6, 1996.

[2] Net-SSH-Expect, http://www.cpan.org/

[3] J. Ouellette, "Paranoid penguin: Managing SSH for scripts and cron jobs," Linux Journal, no.137, 2005.

[4] R. Strubinger, "Shell scripting: A homegrown backup solution utilizing RSA keys, SSH, and tar," J. Sys. Admin., vol.11, no.4, pp.37–38, 2002.

[5] C. Rapier and B. Bennett, "High speed bulk data transfer using the SSH protocol," Proc. 15th ACM Mardi Gras Conference, no.11, pp.1–7, 2008.

[6] F. Pellegrin and C. Pellegrin, "System administration: Secure logging over a network," Linux Journal, no.74, 2000.

[7] V. Marinov and J. Schonwalder, "Performance analysis of SNMP over SSH," Proc. 17th IFIP/IEEE International Conference on Distributed Systems: Operations and Management, vol.12, no.3, pp.25–36, 2006.

[8] SANS Internet Storm Center, http://isc.sans.edu/

[9] J. Vykopal, T. Plesnik, and P. Minarik, "Network-based dictionary attack detection," Proc. International Conference on Future Networks, pp.23–27, 2009.

[10] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A flow-based SSH intrusion detection system," Lect. Notes Comput. Sci., vol.7279, pp.86–97, 2012.

[11] K. Takemori, D.A.L. Romana, S. Kubota, K. Sugitani, and Y. Musashi, "Detection of NS resource record DNS resolution traffic, host search, and SSH dictionary attack activities," Int. J. Intelligent Engineering and Systems, vol.2, no.4, pp.35–42, 2009.

[12] A. Satoh, Y. Nakamura, and T. Ikenaga, "SSH dictionary attack detection based on flow analysis," Proc. 12th IEEE/IPSJ International Symposium on Applications and the Internet, pp.51–59, 2012.

[13] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) transport layer protocol," RFC 4253, 2006.

[14] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) authentication protocol," RFC 4252, 2006.

[15] T. Ylonen and C. Lonvick, "The Secure Shell (SSH) connection protocol," RFC 4254, 2006.

[16] OpenSSH, http://www.openssh.com/

[17] Tectia, http://www.ssh.com/

[18] R.K. Guha, Z. Furqan, and S. Muhammad, "Discovering man-in-the-middle attacks in authentication protocols," Proc. 26th IEEE Conference on Military Communications, pp.1–7, 2007.

[19] J. Owens and J. Matthews, "A study of passwords and methods used in brute-force SSH attacks," tech. rep., Department of Computer Science, Clarkson University of New York, 2008.

[20] D. Ramsbrock, R. Berthier, and M. Cukier, "Profiling attacker behavior following SSH compromises," Proc. 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp.119–124, 2007.

[21] SSHGuard, http://www.sshguard.net/

[22] DenyHOSTS, http://denyhosts.sourceforge.net/

[23] SSHBLACK, http://sshblack.com

[24] BlockHosts, http://www.aczoom.com/tools/blockhosts/

[25] BruteForceBlocker, http://danger.rulez.sk/projects/bruteforceblocker/

[26] J.L. Thames, R. Abler, and D. Keeling, "A distributed active response architecture for preventing SSH dictionary attacks," Proc. IEEE Southeast Conference, pp.84–89, 2008.

[27] A. Satoh, Y. Nakamura, and T. Ikenaga, "A flow-based detection method for stealthy dictionary attacks against secure shell," J. Information Security and Applications, available online: http://www.sciencedirect.com/science/article/pii/S221421261400129X, 2014.

[28] M. Javed and V. Paxson, "Detecting stealthy, distributed SSH bruteforcing," Proc. 20th ACM SIGSAC Conference on Computer & Communications Security, pp.85–96, 2013.

[29] A.W. Moore, D. Zuev, and M.L. Crogan, "Discriminators for use in flow-based classification," tech. rep., Department of Computer Science, Queen Mary University of London, 2005.

[30] A. Satoh, Y. Nakamura, and T. Ikenaga, "Identifying user authentication methods on connections for SSH dictionary attack detection," Proc. IEEE 37th Annual Computer Software and Applications Conference, pp.593–598, 2013.

[31] C.V. Wright, F. Monrose, and G.M. Masson, "Using visual motifs to classify encrypted traffic," Proc. 3rd International Workshop on Visualization for Computer Security, pp.41–50, 2006.

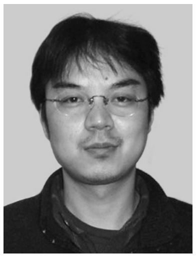
[32] N. Provos and P. Honeyman, "ScanSSH — Scanning the Internet for SSH servers," Proc. 15th USENIX Systems Administration Conference, pp.25–30, 2001.

[33] Net-Scan-SSH-Server-SupportedAuth, http://www.cpan.org/

[34] L. Bernaille, R. Teixeira, and K. Salamation, "Early application identification," Proc. ACM CoNext Conference, no.6, pp.1–12, 2006.

[35] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli, "Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting," Int. J. Computer and Telecommunications Networking, vol.53, no.1, pp.81–97, 2009.

Akihiro Satoh received his Ph.D. degree from Tohoku University in 2011. He is an assistant professor at Information Science Center in Kyushu Institute of Technology. His research interests include network operation and network security.



Yutaka Nakamura received the B.S. degree in polymer science from Kyoto Institute of Technology, Kyoto, Japan, in 1996 and M.E. and D.E. degrees in information systems from Nara Institute of Science and Technology in 1998 and 2001, respectively. From 2001 to 2002, he was a research associate in Engineering Science from Osaka University. From 2002 to 2004, he was an assistant professor at Information Technology Center in Nara Institute of Science and Technology. Currently, he is an associate professor at Information Science Center in Kyushu Institute of Technology. His research interests include technology for network monitoring, network operation and network security. He is a member of the IEICE and IPSJ.

Takeshi Ikenaga received B.E., M.E. and D.E. degrees in computer science from Kyushu Institute of Technology, Iizuka, Japan in 1992, 1994 and 2003, respectively. From 1994 to 1996, he worked at NEC Corporation. From 1996 to 1999, he was an assistant professor in the Information Science Center, Nagasaki University. From 1999 to 2004, he was an assistant professor in the Department of Computer Science and Electronics, Faculty of Computer Science and Systems Engineering, Kyushu Institute of Technology. From March 2004 to 2011, he was an associate professor in the Department of Electrical, Electronic and Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology. Since 2011, he has been a professor in the Department of Electrical, Electronic and Computer Engineering, Faculty of Engineering, Kyushu Institute of Technology. His research interests include performance evaluation of computer networks, wireless LANs and QoS routing. He is a member of the IEEE and IEICE.