Available online at www.sciencedirect.com

ScienceDirect

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

A flow-based detection method for stealthy dictionary attacks against Secure Shell☆

*Akihiro Satoh**, *Yutaka Nakamura*, *Takeshi Ikenaga*

Kyushu Institute of Technology, 1-1 Sensui-cho, Tobata-ku, Kitakyushu-shi, Fukuoka, Japan

ARTICLE INFO

Article history:
Available online 26 September 2014

Keywords:
ssh dictionary attack
Flow analysis
Network operation
Machine learning algorithm

ABSTRACT

SANS has warned about the new variants of SSH dictionary attacks that are very stealthy in comparison with a simple attack. In this paper, we propose a new method to detect simple and stealthy attacks by combining two key innovations. First, on the basis of our assumptions, we employ two criteria: *"the existence of a connection protocol"* and *"the inter-arrival time of an auth-packet and the next"*. These criteria are not available, though, owing to the confidentiality and flexibility of the SSH protocol. Second, we resolve this problem by identifying *"the transition point of each sub-protocol"* through flow features and machine learning algorithms. We evaluate the effectiveness through experiments on real network traffic at the edges in campus networks. The experimental results show that our method provides high accuracy with acceptable computational complexity.

1. Introduction

Secure Shell (SSH) is run on many hosts with various scopes other than just operation, so a dictionary attack against SSH services is a common security threat. In addition to the attack, the SysAdmin, Audit, Network, Security (SANS) Institute (SANS Internet Storm Center) has warned about new variants, namely slow-motion SSH dictionary attack and distributed SSH dictionary attack. The variants are very stealthy in comparison with a simple one. Since even one success of these attacks causes serious problems, such as leaks of confidential information, transmission of spam email, and deployment of phishing sites, administrators should be prepared to cope with them.

SSH dictionary attacks have been detected in two basic ways that rely on either log files (Thames et al., 2008; Su et al., 2011) or network traffic (Sperotto et al., 2009; Takemori et al., 2009). The first approach parses the log file of all hosts in large networks, and thereby imposes heavy maintenance costs on administrators. The second approach limits the above costs because its requirement is to only capture traffic through a few observation points. However, this approach cannot distinguish between successful and unsuccessful attacks. Of more immediate concern, both approaches are ineffective in the case of stealthy attacks that have little impact on log files and network traffic. An ideal method should be able to detect stealthy attacks and to distinguish between their success and failure, based on network traffic of each connection.

In this paper, we focus on realizing such a method to enable secure networks. We have developed the method by combining two key innovations (Satoh et al., 2012). First, we employ two criteria in our assumptions that are derived by reference to SSH protocol specifications. The specifications show that an SSH handshake consists of three major sub-protocols — i.e., transport layer, user authentication, and connection protocols — and that an auth-packet contains a

☆ This article belongs to the special issue Security, Privacy and Trust in Future Networks and Mobile Computing.

* *Corresponding author.*
E-mail addresses: satoh@isc.kyutech.ac.jp (A. Satoh), yutaka-n@isc.kyutech.ac.jp (Y. Nakamura), ike@ecs.kyutech.ac.jp (T. Ikenaga).

http://dx.doi.org/10.1016/j.jisa.2014.08.003
2214-2126/

username and password pair for authentication. A summary explanation of the two criteria is as follows:

The existence of a connection protocol — a criterion to estimate whether a username and password pair is accepted in authentication, to distinguish between successful and unsuccessful dictionary attacks;
The inter-arrival time of an auth-packet and the next — a criterion to estimate whether a username and password pair is entered by user's keystrokes, to detect dictionary attacks.

These criteria are not available, however, owing to the confidentiality and flexibility of the SSH protocol. Second, we resolve this problem by identifying *the transition point of each sub-protocol* through flow features (Moore et al., 2005) and machine learning algorithms (Jain et al., 1999). A transition point is the point at which an SSH handshake shifts to the next sub-protocol in a flow. A flow is bi-directional packet exchanges between a client and a server with the same source address, source port number, destination address, destination port number, and protocol number, and its features are statistical patterns — in terms of, for example, packet size, packet inter-arrival time, and packet order — in externally observable packets taken from a flow. The reason for using flow features rests on two perspectives: (1) the transition point of each sub-protocol typically has the features distinct from those of non transition points; (2) the features are observable without direct packet inspection.

We evaluate the effectiveness of our method through experiments on real network traffic at the edges in campus networks. The experimental results show that our method provides high accuracy with acceptable computational complexity. The significant contribution is a means to alleviate the threat of simple and stealthy attacks that administrators will face in the future.

This paper is organized as follows. Section 2 summarizes related work and their limitations. Our findings from analysis of SSH dictionary attacks at flow level are given in Section 3. On the basis of these analytical results, our method is proposed in Section 4, and this method is evaluated in Section 5. We conclude and look at future work in Section 6.

2. Background

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

2.1. SSH dictionary attack

An SSH dictionary attack is defined as a login attempt to gain fraudulent access by guessing a username and password pair. The attack relies on the fact that many users tend to choose their password from a small domain. A malicious client tries all possible username and password pairs until the correct one is found. As a result, these attacks contaminate log files and flood network traffic.

New variants have emerged as an invisible security threat: slow-motion SSH dictionary attack and distributed SSH dictionary attack. The former type is made by a malicious client, and its target changes one after the other. Specifically, the destination address varies with the login attempts though the source address is constant. The latter type is made by a large coordinated group of malicious clients, such as botnets. Each of the clients perpetrates login attempts against their target at an interval. To be precise, the source address varies with the login attempts though the destination address is constant. In both cases, malicious login attempts leave little impact on log files and network traffic because their number never exceeds single digits over a long time period. Consequently, these attacks are very stealthy in comparison with a simple one.

2.2. SSH protocol specification

An SSH handshake consists of three major sub-protocols: transport layer, user authentication, and connection protocols (Ylonen and Lonvick, 2006a, 2006b, 2006c).

A transport layer protocol negotiates encryption, integrity, compression, and key exchange algorithms to establish secure connections between a client and a server. For example, the encryption algorithms are AES-CBC and 3DES-CBC; the integrity algorithms are HMAC-MD5 and HMAC-SHA1. Note that encryption, integrity, compression algorithms are immediately applied after finishing this sub-protocol. Then, the SSH handshake shifts to a user authentication protocol. The role of this sub-protocol is to determine whether the server allows the client to establish connections via SSH. The client notifies the server of an authentication method name with its attribute values by sending an auth-packet, and the server returns the result of authentication. For example, the method names are password and public-key; the attribute values are username and password. Finally, a connection protocol provides various functions such as remote login, file transfer, X11 forwarding, TCP/IP forwarding, and so on.

The SSH protocol has two notable properties: confidentiality and flexibility. Confidentiality means encrypting connections, checking integrity, and authenticating each other. Flexibility means choosing suitable algorithms according to circumstances. In the transport layer protocol, for example, encryption, integrity, compression, and key exchange algorithms are negotiated independently for each host, so each host chooses its own algorithms from a set it supports.

2.3. Related work

Numerous studies are related to SSH dictionary attack. Traffic causality graphs (Asai et al., 2011) were proposed for visualizing and analyzing the temporal and spatial causality of flows to profile network applications without direct packet inspection. The results helped administrators identify the root that cause various attacks, including SSH dictionary attacks. Another line of work (Goyal et al., 2006; Alsaleh et al., 2012) was to design a secure protocol for preventing dictionary attacks. For example, Goyal et al. (Goyal et al., 2006) improved an authentication protocol by adding fast one-way hash functions and challenge-response exchanges, and the protocol

was easy to implement without any infrastructural changes. Several sources (Guha et al., 2011; Song et al., 2001) estimated character strings entered by user's keystrokes. Here keystrokes were a part of typing in an SSH connection except its handshake; hence the algorithms were not applicable for detecting SSH dictionary attacks.

The behavior of SSH dictionary attacks was explored in Owens and Matthews (2008); Ramsbrock et al. (2007). On the basis of these reports, the attacks have been detected in two types of ways.

The first type goes through log files and keeps track of unsuccessful login attempts against SSH services. Further connections from a client are denied by dynamically adding a new rule, if the number of unsuccessful login attempts exceeded a pre-defined threshold. Developers released many tools to the public (SSHGuard; DenyHOSTS; SSHBLACK; BlockHosts; BruteForceBlocker). Thames et al. (2008) and Su et al. (2011) outlined similar architecture for preventing SSH dictionary attacks. In the architecture, trustworthy servers gathered, analyzed, and distributed information about malicious clients through collaboration. This approach must be applied on all hosts in large networks, and thereby imposes heavy maintenance costs on administrators.

The second type senses a significant deviation from a predefined threshold, modeling network traffic statistically. The causes of the deviation are deemed to be SSH dictionary attacks. Sperotto et al. (2009) showed that such attacks typically consist of three phases, and the authors represented their behaviors at flow level by Hidden Markov Model. Takemori et al. (2009) discovered a significant upsurge in the number of PTR resource records in DNS traffic while the attacks were underway. This approach limits the above costs because its requirement is to only capture traffic at a few observation points. However, the approach cannot distinguish between successful and unsuccessful attacks.

Unfortunately, both approaches are ineffective in the case of stealthy attacks, which have little impact on log files and network traffic. We could consider using a more aggressive threshold, but the benefit is minimal due to the following reasons: (1) individual malicious hosts usually establish very few connections during stealthy attacks; (2) an aggressive threshold would penalize all users who incorrectly input their credentials. Thus, an ideal method should be able to detect stealthy attacks and to distinguish between their success and failure, based on network traffic of each connection.

In this paper, we extend our previous work Satoh et al. (2012) in several ways. First, we provide a framework to our proposal, by formalizing it as pattern-recognition problem. Second, we expand the tests to verify the effectiveness in detecting stealthy dictionary attacks. In addition to the tests, we indicate a qualitative comparison of our work and previous work, and discuss the weaknesses in the actual operation.

3. Analysis of SSH dictionary attacks at flow level

We realize the ideal method by combining two key innovations. The first innovation is to employ two criteria in our assumptions:

the existence of a connection protocol — a criterion to estimate whether a username and password pair is accepted in authentication, to distinguish between successful and unsuccessful dictionary attacks;
the inter-arrival time of an auth-packet and the next — a criterion to estimate whether a username and password pair is entered by user's keystrokes, to detect dictionary attacks.

However, these criteria are not available owing to the confidentiality and flexibility of the SSH protocol. The confidentiality is to encrypt connections, to check integrity, and to authenticate each other, thereby constraining direct packet inspection. The flexibility is to choose suitable algorithms according to circumstances, thereby varying the number of packets transmitted in each sub-protocol. For this reason, it is difficult to estimate the two criteria.

We resolve this problem by the second innovation that is to identify *the transition point of each sub-protocol* through flow features and machine learning algorithms. The reason for using flow features rests on two perspectives: (1) the transition point of each sub-protocol typically has distinct the features from non transition points; (2) the features are observable without direct packet inspection.

Datasets used in our analysis are described in Section 3.1. We verify our assumptions in Section 3.2 and 3.3, and then show the effectiveness of flow features for identifying the transition point of each sub-protocol in Section 3.4.

3.1. *Analysis datasets*

Table 1 summarizes the two analysis datasets. We installed monitoring tools (Tshark) to capture traffic through two observation points. *D1* was traffic traces observed at an SSH server in the Internet, and *D2* was collected by low-level honeypots (Kojoney).

To establish a reference point in the analysis, we first extracted SSH flows with the same source address, source port number, destination address, destination port number, and protocol number from each dataset. Second, the flows were given corresponding labels — i.e., Successful SSH Dictionary Attack, Unsuccessful SSH Dictionary Attack, and SSH Connection — in accordance with investigating the log files of each host. Here, each label means the following: a successful SSH dictionary attack is a connection that succeeded in cracking a password by a malicious user; an unsuccessful SSH dictionary attack is a connection that failed in cracking a password by a malicious user; an SSH connection is login attempt by a user, and the category includes both success and failure of login. Finally, we removed TCP control packets — i.e., SYN, FIN, or ACK flags with no payload — from all flows because exchange of these packets is application-independent.

Table 1 – Number of flows in analysis datasets.

	SSH dictionary attack		SSH connection	Total
	Successful	Unsuccessful		
D1	0	41,279	384	41,663
D2	70	22,372	0	22,442

3.2. Analysis of using the existence of a connection protocol

We assume the existence of a connection protocol is effective for distinguishing between successful and unsuccessful attacks. The rationale is as follows: an SSH server accepts a username and password pair when a dictionary attack is successful; then its handshake shifts to the connection protocol from the user authentication protocol; thus, the connection protocol of a successful attack has several packets in contrast to that of an unsuccessful attack.

To verify this assumption, we confirmed the number of packets transmitted in each sub-protocol. We used all flows of $D2$ for the analysis. The results are shown in Fig. 1. An open square corresponds to the mean number of packets, and a horizontal bar corresponds to the standard deviation. A label in the front of parentheses denotes the type of flow, and a label in parentheses denotes the type of sub-protocol. For example, $S(T)$ is the transport layer protocol of successful attacks, $S(A)$ is the user authentication protocol of successful attacks, and $U(C)$ is the connection protocol of unsuccessful attacks. We found a considerable disparity between S and U. For $S(C)$, the mean number of packets was 5.45, and the standard deviation was 0.85. In contrast, for $U(C)$ the mean number of packets was 0, and the standard deviation was 0. Thus, in successful attacks, the connection protocol had a few packets, whereas in unsuccessful attacks, the sub-protocol had no packets. These results indicate that the existence of a connection protocol is effective for distinguishing between successful and unsuccessful attacks.

3.3. Analysis on using the inter-arrival time of an auth-packet and the next

Before explaining our assumption, we should provide some the details of two terms, "auth-packet" and "inter-arrival time". An auth-packet contains an authentication method name with attribute values. For example, the method names are password and public-key; the attribute values are username and password. A client notifies a server of an authentication method name with its attribute values by sending an auth-packet, and then the server returns the result of authentication. Inter-arrival time is represented as $t_i(x) - t_{i-1}(x)$, where t_i means the observed time of the i-th packet in flow x. We assume the inter-arrival time of an auth-packet and the next can be effective for detecting SSH dictionary attacks. The rationale is as follows: for an SSH connection, a username and password pair is manually entered by user's keystrokes, while for an SSH dictionary attack, the pair is automatically entered by malicious user's dictionary; thus, the time taken to enter the pair in an SSH connection is quite different from that in an SSH dictionary attack.

To verify this assumption, we conformed the inter-arrival time of an auth-packet and the next in both SSH connection and SSH dictionary attack. First, we classified all flows of $D1$ into two types: SSH connection and SSH dictionary attack. We measured the inter-arrival time for each type of flow, and then calculated cumulative probability distributions by kernel density estimation (Wegman, 1972). The results are shown in Fig. 2, where the X-coordinate corresponds to the inter-arrival time, and the Y-coordinate corresponds to the cumulative probability. Over 99% of SSH connections occurred in the range from 2.0 s to 7.0 s and over 99% of SSH dictionary attacks occurred in the range from 0 s to 0.4 s. There is a considerable disparity between the two types. Thus, these results indicate that the inter-arrival time of an auth-packet and the next is effective for detecting SSH dictionary attacks.

3.4. Analysis on using the transition point of each sub-protocol

The effectiveness of the two criteria is supported by the analytical results in Section 3.2 and 3.3. However, these criteria are not available owing to the confidentiality and flexibility of the SSH protocol. As the first step toward measuring the criteria, we analyze the details of packet exchanges in a flow.

First, we randomly chose three types of flow — i.e., SSH connection, successful SSH dictionary attack, and unsuccessful SSH dictionary attack — from $D1$ and $D2$. Then all packets of each flow were investigated to clarify the relations between packet size, packet direction, and sub-protocol with respect to sequence number. To visualize the relations, we adopted Wright et al.'s technique (Wright et al., 2006).

The results are shown in Fig. 3. Here, we point out that each result is from the typical instance of each flow. A symbol

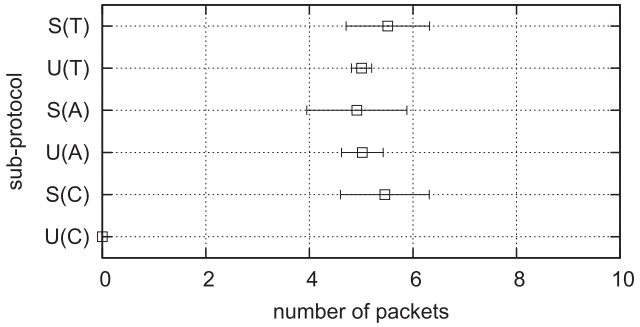


Fig. 1 – Number of packets in each sub-protocol.

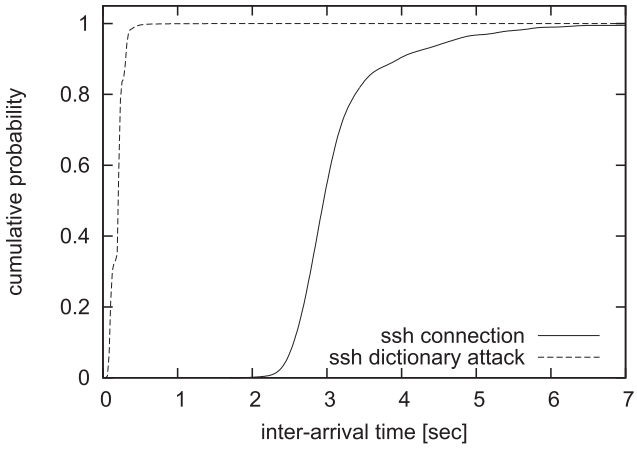
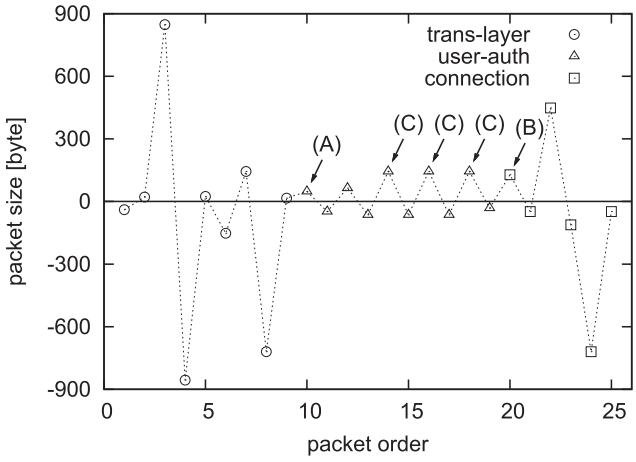
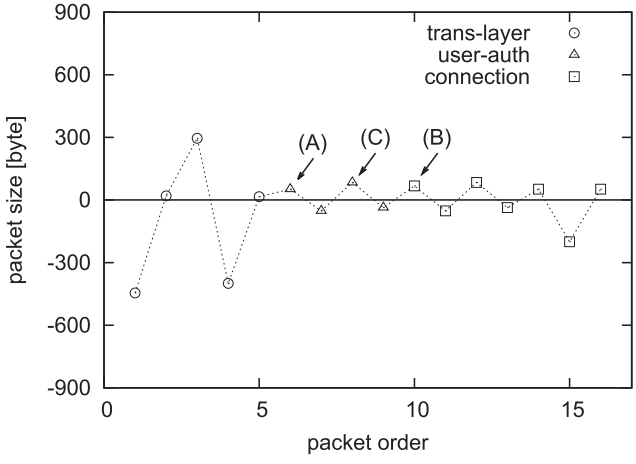


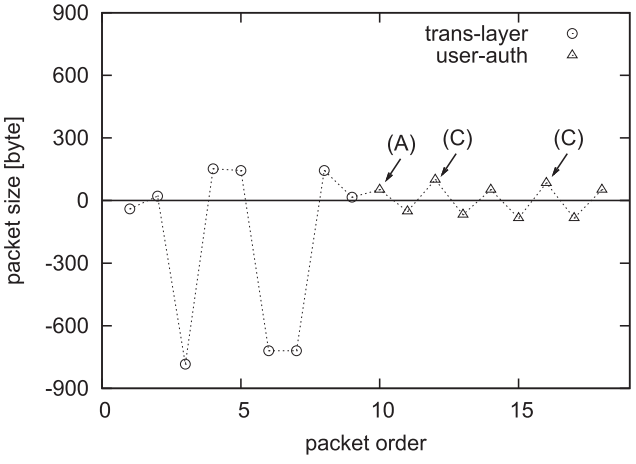
Fig. 2 – Cumulative probability distributions of inter-arrival time of an auth-packet and the next.



(a) SSH Connection



(b) Successful SSH Dictionary Attack



(c) Unsuccessful SSH Dictionary Attack

Fig. 3 – Relations between packet size, packet direction, auth-packet, and sub-protocol with respect to sequence number.

represents a packet in an individual flow, and the type of symbol represents the sub-protocol of a packet. The X-coordinate corresponds to the sequence number, and the Y-coordinate corresponds to the packet size and the packet direction. Positive and negative Y values are respectively in the "incoming" and "outgoing" directions, where "incoming" means transmission to a server from a client and "outgoing" means the reverse transmission. 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, labels with arrows denote the following: (A) the transition point to a user authentication protocol from a transport layer protocol, (B) the transition point to a connection protocol from a user authentication protocol, and (C) the transmission point of an auth-packet. The multiple transmission points of an auth-packet in a flow are due to authentication failure. In Fig. 3(a) and (b), the auth-packet appeared before the transition point from the user authentication protocol to the connection protocol, and in Fig. 3(c), the auth-packet appeared before the end of the flow. In Fig. 3(a), (b), and (c), the difference of the packets in the transport layer protocol was caused by negotiation of encryption, integrity, compression, and key exchange algorithms. The results tell us that the two criteria are measurable by identifying the transition point of each sub-protocol. Furthermore, the results show that the packet at the transition point of each sub-protocol and the next packet are indicated by one of many packet pairs in a flow. A packet pair means two consecutive request and response packets in a flow, and their directions are respectively incoming and outgoing. Let us assume, for example, that twelve packet pairs can be found in Fig. 3(a). The pair of the 6-th and 7-th packets includes the transition point to the user authentication protocol from the transport layer protocol; the pair of the 10-th and 11-th packets includes the transition point to the connection protocol from the user authentication protocol. Consequently, these results indicate the following: (1) the two criteria can be measured by identifying the transition point of each sub-protocol; (2) the packet at the transition point of each sub-protocol and the next packet are indicated by one of many packet pairs in a flow.

As the next step, to select the packet pair including the transition point from the many in a flow, we compare flow features at the transition points with those at non transition points. We should provide two definitions in advance of the analysis.

First, a sub-flow consists of some consecutive packets that are determined on the basis of a packet pair. More specifically, when the pair has the i-th and $(i+1)$-th packets in flow x, the sub-flow x_i is given by $(u+v+2)$-tuples:

$$x_i = \left[p_{i-u}(x), \cdots, p_i(x), p_{i+1}(x), \cdots, p_{i+1+v}(x)\right]. \quad (1)$$

Here $p_i(x)$ means the i-th packet in flow x; u and v mean the parameters relevant to length of a sub-flow. Second, flow features are statistical patterns in externally observable packets, and the features of the sub-flow x_i are represented as a $(u+v+2)$-dimensional vector:

$$\mathbf{x}_i = (s_1(x_i), s_2(x_i), \ldots, s_n(x_i), \ldots, s_{u+v+2}(x_i)). \quad (2)$$

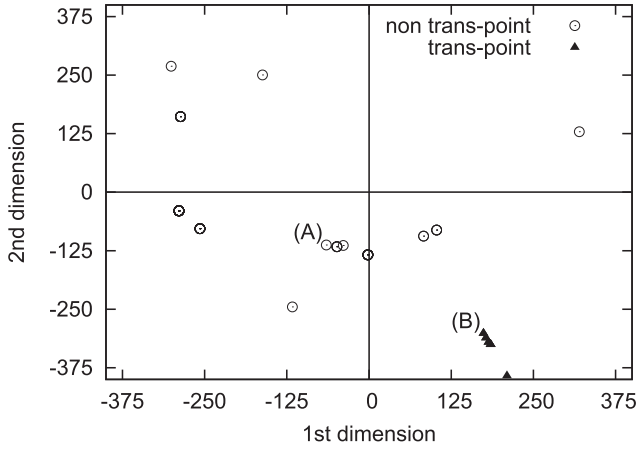
Here, we use the three types of flow features — i.e., packet size, packet direction, and packet order — for the vectorial representation. The absolute value of $s_n(x_i)$ denotes the size of the n-th packet in sub-flow x_i. The sign of $s_n(x_i)$ denotes the

direction of the n-th packet. For example, $s_n(x_i) = -500$ means the packet that is the n-th in sub-flow x_i, 500 bytes in length, and the outgoing direction. From Euclidean Distance it follows that the similarity between two sub-flows is given by Equation (3), where $||\boldsymbol{x}||_2$ means the 2-norm of the vector x.

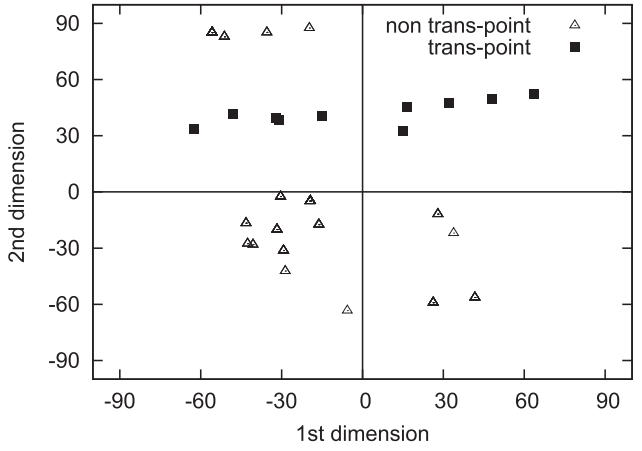
$$dist(x_i, x_j) = ||\boldsymbol{x}_i - \boldsymbol{x}_j||_2. \quad (3)$$

The analytical procedure was as follows. First, all sub-flows in $D1$ were first represented as $(u + v + 2)$-dimensional vectors, where the two parameters were tentatively set as $u = 1$ and $v = 1$ in this analysis. We will discuss the optimization of the parameters in Section 5.2. Then, the sub-flows were next categorized into two sets: (a) transport layer protocol, and (b) user authentication protocol. Both sets naturally included sub-flows taken from the transition points of the next sub-protocol. Finally, the sub-flows in each set were assigned to two-dimensional space by Multi-Dimensional Scaling (Torgerson, 1952) that is a typical means for visual comparison. Multi-Dimensional Scaling took a set of multi-dimensional vectors and returns a set of the best-fitting points on a scatter chart.

The results are shown in Fig. 4. In Fig. 4(a), a solid triangle symbol labeled "trans-point" refers to a sub-flow that includes a transition point from a transport layer protocol to a user authentication protocol, and an open circle symbol labeled "non trans-point" refers to a sub-flow that includes no transition points in a transport layer protocol. In Fig. 4(b), a solid square symbol labeled "trans-point" refers to a sub-flow that includes a transition point from a user authentication protocol to a connection protocol, and an open triangle symbol labeled "non trans-point" refers to a sub-flow that includes no transition points in a user authentication protocol. Fig. 4(a) has 120 thousand of these symbols; Fig. 4(b) has 95 thousand of these symbols. In both cases, the many symbols of the same type form the high-density cluster, and the cluster is far from those including the other symbols. For example, the clusters labeled (A) and (B) consist of more than 15 thousand of sub-flows, and no clusters of the other type lie close to them. These results indicate that transition points have features distinct from those of non transition points. Thus, flow features are effective for identifying the transition point of each sub-protocol.



(a) Transport Layer Protocol



(b) User Authentication Protocol

Fig. 4 – Comparison of flow features.

4. Proposal

We propose an SSH dictionary attack detection method by combining the two innovations. Fig. 5 gives an overview of our method that consists of four functions: (1) flow feature calculation, (2) sub-protocol training, (3) sub-protocol identification, and (4) SSH dictionary attack detection. The first three functions identify *the transition point of each sub-protocol* through flow features and machine learning algorithms. The last function detects individual dictionary attacks by *the inter-arrival time of an auth-packet and the next*, and it distinguishes between their success and failure by *the existence of a connection protocol*.

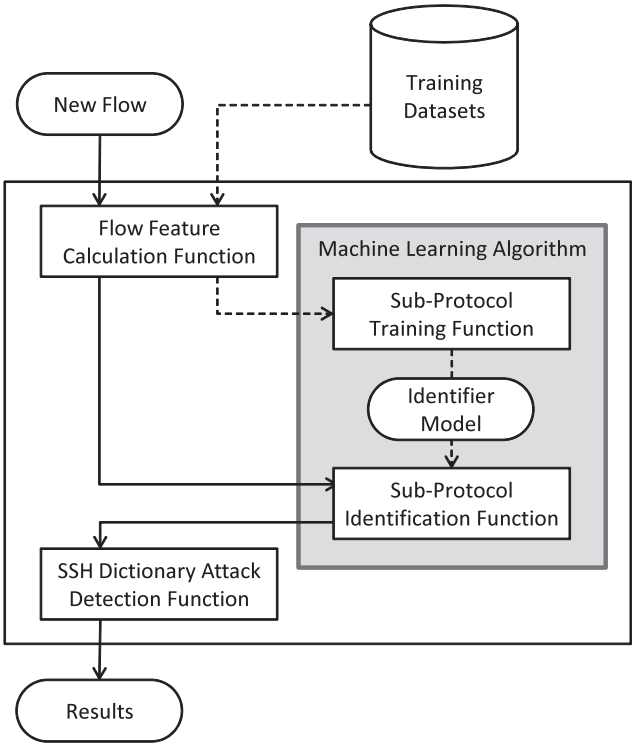


Fig. 5 – Overview of the SSH dictionary attack detection method based on flow analysis.

4.1. Flow feature calculation function

The flow feature calculation function selects sub-flows with respect to the packet pairs, and calculates the features of each sub-flow for numerical representation.

First, the function investigates all packets in flow x to find packet pairs. A packet pair consists of request and response packets. The request is transmitted to a server from a client, and the response is transmitted in the reverse direction. Thus, the function determines that there is a packet pair when two packets meet the conditions $d_i(x) = incoming$ and $d_{i+1}(x) = outgoing$, where $d_i(x)$ means the direction of the i-th packet in flow x. Next, the function selects sub-flows with respect to packet pairs. The sub-flow x_i is consecutive packets from $p_{i-u}(x)$ to $p_{i+1+v}(x)$ in flow x when the pair is $p_i(x)$ and $p_{i+1}(x)$, where $p_i(x)$ means the i-th packet in flow x. Finally, the function calculates the features of each sub-flow for numerical representation. The features are packet order, packet size, and packet direction, and their numerical representation is described by Equation (2).

4.2. Sub-protocol training function

The sub-protocol training function derives identifier models through machine learning algorithms. We employ Ward Clustering (Ward, 1963) based on Euclidean Distance as the machine learning algorithm, and the algorithm constructs hierarchical clusters for aggregating similar sub-flows. The reason for employing the hierarchical clustering algorithm is the difficulty of knowing beforehand the number of clusters. Needless to say, the clustering algorithm can be easily replaced if the other algorithms are more appropriate for our method.

Input to the function is sub-flows taken from training datasets. First, sub-flows are categorized into two sets: (a) transport layer protocol and (b) user authentication protocol. Both sets naturally include sub-flows taken from the transition points of the next sub-protocol. $\mathbb{X}^\alpha$ indicates the former set, and $\mathbb{X}^\beta$ indicates the latter. Next, the clustering algorithm aggregates sub-flows in each set when the similarity between the sub-flows is less than a pre-defined threshold. $\mathbb{Y}^\alpha$ and $\mathbb{Y}^\beta$ represent the clustering results of $\mathbb{X}^\alpha$ and $\mathbb{X}^\beta$ when the thresholds are set as th^α_{class} and th^β_{class}, respectively. In consequence, an individual cluster is constructed from sub-flows that contain the same request and response packets. Then, we investigate individual clusters in $\mathbb{Y}^\alpha$ and $\mathbb{Y}^\beta$, and remove the clusters when their sub-flows are not taken from the transition points. $\mathbb{Z}^\alpha$ and $\mathbb{Z}^\beta$ are the sets of clusters yielded by the above mentioned processing. Finally, the function outputs the carefully selected sets as identifier models.

4.3. Sub-protocol identification function

The sub-protocol identification function outputs the range of each sub-protocol in a new flow by comparing the identifier models $\mathbb{Z}^\alpha$ and $\mathbb{Z}^\beta$.

Input to the function is the sub-flow x_i taken from the new flow x. First, the function searches the identifier model $\mathbb{Z}^\alpha$ to find the best fitting cluster for the sub-flow x_i. Its mathematical representation is $\min_{z_j \in \mathbb{Z}^\alpha} dist(x_i, z_j) < th^\alpha_{class}$, in which z_j means the centroid of a cluster in the identifier model $\mathbb{Z}^\alpha$. Next, in the case of satisfying the condition, the function regards the i-th packet as the transition point to the user authentication protocol from the transport layer protocol. On the other hand, in the case of not satisfying the condition, the next of the sub-flow x_i in flow x is input into the function to identify whether the next sub-flow includes the transition point. Then, the identifier model $\mathbb{Z}^\alpha$ and the threshold th^α_{class} are changed to the model $\mathbb{Z}^\beta$ and to the threshold th^β_{class}, and the above processing is repeated to obtain the transition point to the connection protocol from the user authentication protocol. Last, the function outputs the range of each sub-protocol in flow x.

4.4. SSH dictionary attack detection function

The SSH dictionary attack detection function determines whether an individual flow is an SSH dictionary attack, and whether its authentication is successful. The function uses the two explicit criteria: "the existence of a connection protocol" and "the inter-arrival time of an auth-packet and the next". The range of each sub-protocol supports measurement of the two criteria, and the reasons for this are as follows: (1) as is evident from Fig. 3 an auth-packet and its result appear at the end of a user authentication protocol; (2) the existence of a connection protocol speaks for itself.

The function deems a flow to be an attack when the inter-arrival time of its auth-packet and the next is less than a threshold th_{time}. Here, th_{time} is defined as the minimum time spent to enter a username and password pair by a user. Then, the attack is then considered a success when its connection protocol exists. Consequently, the detection results are used to profile and to block the attack according to institutional policies.

5. Evaluation

In this section, we evaluate the effectiveness of our method through experiments on eight datasets. The evaluation points are mainly given for (1) accuracy of identifying the range of each sub-protocol, (2) accuracy of detecting and distinguishing dictionary attacks, (3) computational complexity, and (4) comparison with previously reported methods.

5.1. Testing and training datasets

The eight datasets in Table 2 were used as a "testing dataset" to assess validity of our method. *D5*, *D6*, *D7*, *D8*, *D9*, and *D10* were the sets of flows captured at each edge in six campus networks, and their observation periods were one day. The prefix length of the first two networks was 16, and the prefix length of the other networks was 20. *D11* and *D12* were collected by two honeypots for two months. The three datasets in Table 3 were used as a "training dataset" to create identifier models. *D3* and *D4* were the same observation points and the same observation periods as *D5* and *D7*, respectively. *D2* was described in Section 3.1.

All flows in these datasets were given corresponding labels — i.e., Successful SSH Dictionary Attack, Unsuccessful SSH

Table 2 – Number of flows in testing datasets.

	SSH dictionary attack		SSH connection	Total
	Successful	Unsuccessful		
D5	0	62,104	83	62,187
D6	0	15,321	68	15,389
D7	0	7576	72	7648
D8	0	23,917	123	24,040
D9	0	7717	136	7853
D10	0	3578	39	3617
D11	85	13,805	0	13,890
D12	104	45,403	0	45,507

Dictionary Attack, and SSH Connection — by the following: (1) investigation of log files at SSH servers, (2) search of a WHOIS database (Daigle, 2004) for the source and destination address, and (3) comparison with known flows. The meaning of each label has been described in Section 3.1. As expected, TCP control packets were removed from these datasets.

5.2. Accuracy of identifying range of sub-protocols

Accuracy of identification is a metric to measure the proportion of the flows in which the range of each sub-protocol is correctly identified. We define the metric as the following equation:

$$\mathscr{A}(D_L) = \sum_{x \in D_L} \frac{\phi(pt^{\alpha}(x), n^{\alpha})\phi(pt^{\beta}(x), n^{\beta})}{|D_L|}. \quad (4)$$

Here, D_L is the set of flows given the same labels in a dataset, and the absolute value means the number of elements in the set. Examples of the set are the 83 flows of SSH connections in D5 and the 13,805 flows of unsuccessful attacks in D11. $pt^{\alpha}(x)$ and $pt^{\beta}(x)$ are the identification results of each transition point when the actual transition points are n^{α} and n^{β}. ϕ is an indicator function: $\phi(a, b) = 1$ if $a = b$, and $\phi(a, b) = 0$ if $a \neq b$.

Table 4 shows the accuracy of identification varying with the size of a sub-flow in identifier models. The size of a sub-flow is written as (u, v), where u and v mean the number of packets before and after a packet pair. Label C, S, and U denote the flows of SSH connections, successful attacks, and unsuccessful attacks, respectively. The two thresholds relevant to clustering of sub-flows were $th^{\alpha}_{class} = 100$ and $th^{\beta}_{class} = 50$. For U in D5 and D11, all cases except (3,3) gave the accuracy in excess of 0.95. For C in D5 and S in D11, the accuracy of (1, 0), (2, 0), and (3, 0) was far superior to that of the other cases. The results indicate that the models of (1, 0), (2, 0), and (3, 0) are suitable for identifying the range of each sub-protocol.

Fig. 6 shows the number of clusters varying with the size of a sub-flow in identifier models. The X-coordinate corresponds to the number of packets in a sub-flow, and the Y-coordinate corresponds to the number of clusters held by identifier models. The bar types indicate the type of identifier model, that is, $\mathbb{Z}^{\alpha}$ or $\mathbb{Z}^{\beta}$ as described in Section 4.2. For (1, 0), the identifier models had far fewer clusters than the other models, and the number of clusters was 3 and 5. The results indicate that the models of (1, 0) best reduce the computational complexity of our method because the range of each sub-protocol is identifiable within a few comparisons.

Table 3 – Number of flows in training datasets.

	SSH dictionary attack		SSH connection	Total
	Successful	Unsuccessful		
D2	70	22,372	0	22,442
D3	0	28,665	157	28,822
D4	0	19,723	92	19,815

5.3. Accuracy of detecting and distinguishing SSH dictionary attacks

A common way to characterize the accuracy of detection and distinction is through two metrics known as true positive and false positive. 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:

$$\mathscr{T}(D_L) = \sum_{x \in D_L} \frac{\phi(f(x), L)}{|D_L|}, \mathscr{F}(D_L) = \sum_{x \in \overline{D}_L} \frac{\phi(f(x), L)}{|\overline{D}_L|}. \quad (5)$$

Here, D_L and $\overline{D}_L$ are respectively the set of flows given the same labels and the set of the other flows in a dataset, and the absolute value means the number of elements in the set. $f(x)$ is output of our method when the true label of the flow x is L. ϕ is an indicator function: $\phi(a, b) = 1$ if $a = b$, and $\phi(a, b) = 0$ if $a \neq b$.

The accuracy calculated from the metrics are shown in Table 5. We set several parameters relevant to our method as $p = 1$, $q = 0$, $th^{\alpha}_{class} = 100$, $th^{\beta}_{class} = 50$, and $th_{time} = 1.5$. Our method kept the accuracy high in these worst cases: (1) the flows of SSH connections in D9 were given the correct labels with true positive of 0.911; (2) the flows of successful attacks in D12 were given the correct labels with true positive of 0.913; (3) the flows of unsuccessful attacks in D10 were given the correct labels with true positive of 0.988. Additionally, false positives of all were under 0.050 in the experiments. Thus, these results imply that our method could point toward a breakthrough in achieving practical attack detection and distinction, although there were a few errors.

These errors were mainly in four categories. The first category was where a client and a server in a busy state caused loss and delay of an auth-packet, and the inter-arrival time exceeded the pre-defined threshold th_{time}. Because of this, a total of 221 flows incorrectly identified.

The second category was where retransmission of packets caused changes of flow features, and the flow features reduced the accuracy of identifying the range of each sub-protocol. In this case, a total of 173 flows were incorrectly identified.

In the third category, errors were attributed to automatic SSH login without any keystrokes. For example, administrators employ public key authentication with no passphrase to automate management tasks through SSH; the inter-arrival time of the connection is small as well as that of dictionary attacks; therefore our method erroneously deems a normal connection to be an attack. Owing to this problem, a total of 12 flows were incorrectly identified.

In the fourth category, errors were attributed to the slight disparity in client software implementation. For example,

Table 4 – Accuracy of identifying the range of each sub-protocol.

		(1,0)	(1,1)	(2,0)	(2,1)	(2,2)	(3,0)	(3,1)	(3,2)	(3,3)
D5	C	0.987	0.192	0.951	0.084	0.084	1.000	0.048	0.048	0.000
	U	0.999	0.999	0.999	0.997	0.966	0.999	0.996	0.964	0.649
D11	S	0.934	0.400	0.957	0.442	0.242	0.985	0.442	0.242	0.157
	U	0.997	0.999	0.999	0.998	0.995	0.999	0.997	0.994	0.750

general SSH clients require input of a username and password pair on the eve of sending an auth-packet; in contrast, TeraTerm requires input when a user authentication protocol begins; therefore, the inter-arrival time does not differ between an attack and the other instances. As a result, a total of 19 flows were incorrectly identified.

5.4. *Discussion*

Table 6 provides a qualitative summary of our work and previous work for the categories of (1) data, (2) cost, (3) complexity, (4) simple attack, and (5) stealthy attack. The first category refers to the type of data each method requires, and the second one refers to the cost of applying the method. We have already discussed these in Section 2.3. The third category is the computational complexity evaluated relative to our method. The methods of Thames et al. (2008); Su et al. (2011) merely count the number of authentication failures through log files, their complexity being very low. In contrast, the method of Takemori et al. (2009) must analyze not only SSH traffic but also DNS traffic, so its processing leads to high complexity. Additionally, the complexity of the method of Sperotto et al. (2009) is not clearly stated in the paper. This is because that work focuses on modeling dictionary attacks and the results are utilized for various anomaly detection systems (Sperotto et al., 2010); thus, the complexity depends on the anomaly detection system employed rather than the model. The last two categories mean whether a method can detect and distinguish simple and stealthy attacks. The previous methods are able to detect simple attacks that use repeated connections between a client and a server. However they cannot detect stealthy attacks in which the source and destination addresses frequently change. Our method resolves this problem thereby determining normal connection, successful attack, or unsuccessful attack for an individual flow.

We are aware that there are three concerns about our method. The first is that a malicious user can emulate a normal connection by manipulating the sending rate of the packets; as a result, the inter-arrival time of an auth-packet and the next in the attack can be made consistent with that of a normal one, allowing the user to achieve his goal without being detected if a large-scale botnet is employed. Unfortunately, our method cannot distinguish between normal connections and emulated dictionary attacks, but it would capture the activity of a botnet by the rapid increase in the number of clients that fail authentication. It should be point out that the method does not have any worse than prior work on this problem, and it at least has potential to accomplish a certain result in detection of emulated dictionary attacks.

The second concern is that the SSH specification allows variable amounts of padding to be added to packets before encryption. A malicious user might evade detection of our method by an implementation adding a random amount of padding. However, our method could be alert to padded flows by a little improvement because a random amount of padding causes major changes of flow features. As a result, by giving notice of padded flows, the method assists in investigation by administrators.

The third concern is that when client software supports input of a username and password pair, the inter-arrival time of an auth-packet and the next in a normal connection indicates a value as small as that of an attack; thus, our method will erroneously deem a normal connection to be an attack. To alleviate the problem, we should take account of the name and version of a client's software. When an SSH handshake begins, both sides must first send an announcement string by an unencrypted packet. On the basis of the string including the name and version, our method will determine whether the supporting function has been implemented.

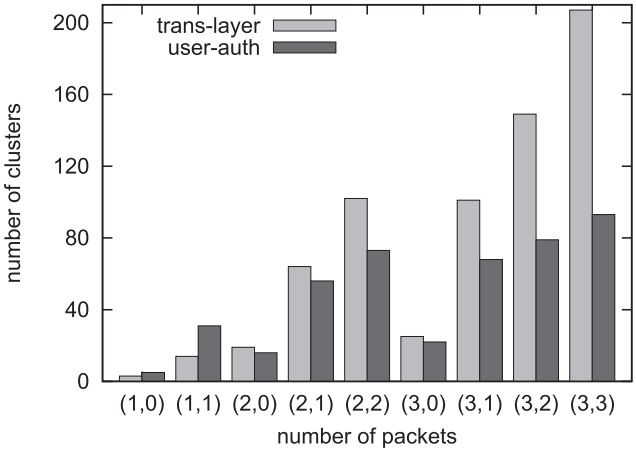


Fig. 6 – Number of clusters in identifier models.

Table 5 – Accuracy of detecting and distinguishing SSH dictionary attacks.

	SSH dictionary attack				SSH connection	
	Successful		Unsuccessful			
	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$	$\mathcal{T}$	$\mathcal{F}$
D5	–	–	0.999	0.000	0.963	0.001
D6	–	–	0.998	0.044	0.955	0.001
D7	–	–	0.999	0.041	0.958	0.000
D8	–	–	0.995	0.024	0.934	0.002
D9	–	–	0.990	0.022	0.911	0.007
D10	–	–	0.988	0.000	0.948	0.008
D11	0.928	0.001	0.997	0.028	–	–
D12	0.913	0.001	0.998	0.048	–	–

Table 6 – Qualitative summary of our work and previous work.

	Data	Cost	Complexity	Simple attack		Stealthy attack	
				Detection	Distinction	Detection	Distinction
Thames et al. (Thames et al., 2008)	log file	high	low	yes	yes	no	no
Su et al. (Su et al., 2011)	log file	high	low	yes	yes	no	no
Sperotto et al. (Sperotto et al., 2009)	traffic	low	not clear	yes	no	no	no
Takemori et al. (Takemori et al., 2009)	traffic	low	high	yes	no	no	no
Our work	traffic	low	moderate	yes	yes	yes	yes

The experimental results presented in Section 5 indicate that our method has the following abilities: (1) to detect stealthy attacks as well as simple attacks, along with their success or failure; (2) to provide high accuracy because true positive and false positive were respectively over 0.90 and under 0.05 in all cases; (3) to process enormous attacks because the complexity of our method was lower than that of the existing traffic based ways. The significant contribution of this work is that it will help alleviate the threat of stealthy attacks faced by administrators in the future.

6. Conclusion

In this paper, we have proposed a method for dealing with SSH dictionary attacks. This paper shows three important points through analysis and evaluation. First, simple and stealthy dictionary attacks are detected along with their success or failure by the two explicit criteria: "the existence of a connection protocol" and "the inter-arrival time of an auth-packet and the next". Furthermore, the range of each sub-protocol supports measurement of the two criteria, and the reasons for this are as follows: (1) an auth-packet and its result appear at the end of a user authentication protocol; (2) the existence of a connection protocol speaks for itself. Second, our method has the following abilities: (1) to provide high accuracy because true positive and false positive were respectively over 0.90 and under 0.05 in all cases; (2) to process enormous attacks because the complexity of our method was lower than that of the existing traffic based ways. Finally, our method has the several weaknesses in the actual operation. At the present time, however, this is the only means of alleviating the threat of stealthy attacks that administrators will face in the future.

In our future work, we will experiment further on the traffic of various dictionary attacks collected from enterprise and campus networks. The various dictionary attacks include not only ones against SSH, but also those against many different services. Hydra (THC-Hydra) is one example of a new tool to attack HTTP, SMTP, IMAP, SMB, MySQL, and many more. From the experimental results, we intend to continue improving our method with the ultimate goal of being able to detect the dictionary attacks against many services to enable secure networks.

Acknowledgment

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

Alsaleh M, Mannan M, Oorschot PCV. Revisiting defenses against large-scale online password guessing attacks. IEEE Trans on Dependable and Secure Computing 2012;9(1):128–41.
Asai H, Fukuda K, Esaki H. Traffic causality graphs: profiling network applications through temporal and spatial causality of flows. In: Proceedings of the 12th INFORMS Computing Society Conference; 2011. p. 95–102.
BlockHosts, http://www.aczoom.com/tools/blockhosts/.
BruteForceBlocker, http://danger.rulez.sk/projects/bruteforceblocker/.
Daigle L. WHOIS protocol specification. 2004. IETF Request for Comments 3912.
DenyHOSTS, http://denyhosts.sourceforge.net/.
Goyal V, Kumar V, Singh M, Abraham A, Sanyal S. A new protocol to counter online dictionary attacks. J Computers & Security 2006;25(2):114–20.
Guha S, Kidwell P, Ashrith Barthur WSC, Gerth J, Bullard C. A Streaming statistical algorithm for detection of SSH keystroke packets in TCP connections. In: Proceedings of the 12th INFORMS Computing Society Conference; 2011. p. 73–92.
Jain AK, Murty MN, Flynn PJ. Data clustering: a review. ACM Computing Surv 1999;31(3):264–323.
Kojoney, http://kojoney.sourceforge.net/.
Moore AW, Zuev D, Crogan ML. Discriminators for use in flow-based classification. Department of Computer Science, Queen Mary University of London; 2005. Tech. rep.
Owens J, Matthews J. A study of passwords and methods used in brute-force SSH attacks. Department of Computer Science, Clarkson University of New York; 2008. Tech. rep.
Ramsbrock D, Berthier R, Cukier M. Profiling Attacker behavior following SSH compromises. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks; 2007. p. 119–24.
SANS Internet Storm Center, https://isc.sans.edu/diary/Distributed+SSH+Brute+Force+Attempts+on+the+rise+again/9031.
Satoh A, Nakamura Y, Ikenaga T. SSH dictionary attack detection based on flow analysis. In: Proceedings of the 12th IEEE/IPSJ International Symposium on Applications and the Internet; 2012. p. 51–9.
Song DX, Wagner D, Tian X. Timing analysis of keystrokes and timing attacks on SSH. In: Proceedings of the 10th Conference on USENIX Security Symposium; 2001. p. 25.
Sperotto A, Sadre R, de Boer P-T, Pras A. Hidden markov model modeling of SSH Brute-Force attacks. Lecture Notes in Computer Science 2009;5841:164–76.
Sperotto A, Schaffrath G, Sadre R, Morariu C, Pras A, Stiller B. An overview of IP flow-based intrusion detection. IEEE Commun Surv and Tutor 2010;12(3):343–56.
SSHBLACK, http://sshblack.com.
SSHGuard, http://www.sshguard.net/.
Su Y-N, Chung G-H, Wu BJ. Developing the upgrade detection and defense system of SSH dictionary-attack for multi-platform environment. J iBusiness 2011;3(1):65–70.

Takemori K, Romana DAL, Kubota S, Sugitani K, Musashi Y. Detection of NS resource record DNS resolution traffic, host search, and SSH dictionary attack activities. International Journal of Intelligent Engineering and Systems 2009;2(4):35–42.
TeraTerm, http://sourceforge.jp/projects/ttssh2/.
Thames JL, Abler R, Keeling D. A distributed Active response architecture for preventing SSH dictionary attacks. In: Proceedings of the IEEE Southeast Conference; 2008. p. 84–9.
THC-Hydra, http://www.thc.org/thc-hydra/.
Torgerson WS. Multidimensional Scaling: I. Theory and method. J Psychometrika 1952;17(4):401–19.
Tshark, http://www.wireshark.org/.
Ward JH. Hierarchical grouping to optimize an objective function. J Amer Statist Assoc 1963;58(301):236–44.
Wegman EJ. Nonparametric probability density estimation. J Statist Comput Simulation 1972;1(3):225–45.
Wright CV, Monrose F, Masson GM. Using visual motifs to classify encrypted traffic. In: Proceedings of the 3rd International Workshop on Visualization for Computer Security; 2006. p. 41–50.
Ylonen T, Lonvick C. The Secure Shell (SSH) transport layer protocol. 2006. IETF Request for Comments 4253.
Ylonen T, Lonvick C. The Secure Shell (SSH) authentication protocol. 2006. IETF Request for Comments 4252.
Ylonen T, Lonvick C. The Secure Shell (SSH) connection protocol. 2006. IETF Request for Comments 4254.