Vol.2019-IOT-46 No.1
2019/6/14

A Design of a Logging System for a Computer Security Incident Response

Motoyuki OHMORI[1,a)]

Abstract: Computer security has been getting more attentions because a computer security incident may cause great damage on an organization. In order to quickly and properly detect and/or handle a computer security incident, ones may need store logging messages of network equipment and servers as many as possible. These logging messages then require a large size of storage while these logging messages are not so useful during daily normal operations. It is then better to minimize sizes of logging files as much as possible. On the other hand, ones may need to retrieve a specific logging message on a computer security incident, and its retrieval time should be shorter as much as possible. It is then necessary to solve this dilemma between a smaller storage size and shorter retrieval time.
To this end, this paper proposes a new logging system dedicated for a computer security incident response. The proposed logging system is designed based upon natures of logging messages which are required for a computer security incident response. This paper firstly tries to find features of logging messages for a computer security incident response. For example, newer logging messages should be retrieved shorter than older logging messages, and all logging messages are in time series. This paper also tries to design a new logging system architecture dedicated for a computer security incident response while logging messages in todays' logging systems are stored in generic database like RDBMS or in text files consisting of raw syslog messages.

1. Introduction

Computer security has been getting more attentions because a computer security incident may cause great damage on an organization. In order to quickly and properly detect and/or handle a computer security incident, ones may need store logging messages of network equipment and servers as many as possible. These logging messages then require a large size of storage while these logging messages are not so useful during daily normal operations. It is then better to minimize sizes of logging files as much as possible. On the other hand, ones may need to retrieve a specific logging message on a computer security incident, and its retrieval time should be shorter as much as possible. It is then necessary to solve this dilemma between a smaller storage size and shorter retrieval time.

To this end, this paper proposes a new logging system dedicated for a computer security incident response. The proposed logging system is designed based upon natures of logging messages which are required for a computer security incident response. This paper firstly tries to define requirements for logging messages for a computer security incident response. For example, newer logging messages should be retrieved shorter than older logging messages, and all logging messages are in time series. This paper also tries to design a new logging system architecture dedicated for a computer security incident response while logging messages in todays' logging systems are stored in generic database like Relational Database Management System (RDBMS) or in text files consisting of raw syslog messages.

The rest of this paper is organized as follows. **Sec. 2** presents our motivations to store and compress logging messages. **Sec. 3** presents required logging messages for detecting and/or handling a computer security incident. **Sec. 4** introduces features of logging messages required for a computer security incident response. **Sec. 5** proposes the logging system dedicated for a computer security incident response. **Sec. 6** refers to related work. **Sec. 7** finally concludes this paper.

2. Background

This section introduces our motivations to design a new logging system dedicated for a computer security incident response.

We, Tottori University, currently employ Network Address Port Translation (NAPT) for almost all edge users. When an external organization alerts a suspicious communication to us, the external organization can provide us with the global IP addresses and port numbers of the suspicious communication. In this situation, we must identify an associated private IP address of the suspicious communication. In order to identify the private IP address, we must store all traffic logging messages of NAPT equipment. Our NAPT equipment is a next-generation firewall, Paloalto PA-5220, that has four 10GbE links for incoming and outgoing traffic. We have one WAN 1GbE link for SINET. In our environment, our PA-5220 requires approximately 20GB/day for all logging messages including threat, URL filtering and data filtering. We may then need about 4TB storage for a year in order to just store traffic logging messages. This size of 4TB is not small for us, and it is better to make its size smaller.

[1] Tottori University, Koyama-minami, Tottori Japan, 680–8550 Japan
[a)] ohmori@tottori-u.ac.jp

In order to make file size of logging messages, we employ an object storage called Swift that can automatically compress a logging file using zip-like algorithm. Swift can reduce physical size of a logging file. We, however, require a long delay to retrieve a logging message of a specific traffic flow. For example, we need 20 min. when we retrieve a specific traffic from from a traffic logging file of one day. This 20 min. may be enough long to intrude and compromise information, and this delay should be minimized.

We have found that almost all cirtical incidents require recent, say a few months, logging messages because almost all critical incidents are detected by recent logging messages only. Few incidents are surely detected by very old logging messages, but these are very rare case. We then employ MongoDB [1] for recent logging messages. From the viewpoint of programmers, MongoDB is useful and flexible. MongoDB is, however, slower than traditional RDBMSes and requires more storage. We cannot have mongoDB store all logging messages because mongoDB requires more storage.

3. Logging Messages for a Computer Security Incident Handling

This section introduces required logging messages for detection and/or handling a computer security incident. This section also introduces actual logging messages in actual environment. Note that leading date, time and host strings of syslog, e.g., `Feb 13 00:45:44 localhost`, of actual logging messages here are omitted due to limited space.

3.1 Firewall

As described **Sec. 2**, traffic logging messages are required for a computer security incident. In our environment, the firewall, PA-5220, produces traffic logging messages. In addition, PA-5220 produces other logging messages. All logging messages are:

(1) traffic including not only denied traffic but also permitted traffic,
(2) detected threat communications,
(3) accessed URLs,
(4) uploaded or downloaded file names, and
(5) system logging messages.

Fig. 1 shows examples of logging messages of PA-5200, and its format is published in [2]. As shown in Fig. 1, Paloalto network equipment produces logging messages in Comma Separated Values (CSV). These logging messages, therefore, may be able to be easily converted into binary format.

Fig. 2 shows examples of logging messages of Cisco FWSM. As shown in Fig. 2, Cisco FWSM series produces logging messages in its own original format. These logging messages, however, are in pre-fixed format, and may be able to be converted into binary format.

Fig. 3 shows examples of logging messages of Fortigate. As shown in Fig. 3, Fortigate series produces also logging messages in its own original format. These logging messages, however, are also in pre-fixed format, and may be able to be converted into binary format.

Fig. 4 shows examples of logging messages of iNetSec by PFU. Note that iNetSec is not actually a firewall but Intrusion Detection System (IDS). As shown in Fig. 4, iNetSec series produces also logging messages in JSON format. These logging messages may be able to be converted into binary format even though it may require heavy processes.

3.2 ARP Table Entries of a Core Switch

For a computer security incident response, we must be able to resolve an associated MAC address from a given IP address. To this end, ones may usually poll ARP table entries from a core switch. **Fig. 5** shows logging messages of our own implementation of ARP table polling by SNMP. An IPv4 address and MAC address are in fixed length, and this pair can be easily converted into binary format.

3.3 ARP Snooping of a Core Switch

In addition to ARP table polling, we have implemented the scalable ARP snooping using policy-based mirroring of core switches, *AXARPS* [3] because ARP table polling has appeared to be inaccurate where the suspicious host moves so fast.

Fig. 5 shows logging messages of our own implementation of AXARPS. An IPv4 address and MAC address are in fixed length, and this pair can be easily converted into binary format.

3.4 DNS Queries

Loggine messages of DNS queries are useful for a computer security incident handling because a infected host usually resolves a FQDN of a malicious host. **Fig. 7** shows DNS query logging messages that *bind* produces. In case of DNS query logging messages, FQDN has a variable length, and it might be difficult to be converted into binary format.

3.5 Network Authentications

Network authentication is important in order to avoid a malicious access to the network, and identify a host and its user. We implement IEEE802.3x authentication for both of wired and wireless LAN. We then utilize freeradius, and its *linelog* module, and our IEEE802.1x authentication logging messages look like **Fig. 8**.

3.6 IdP Authentications

We implement Shibboleth IdP in order to achieve single signon. We then store IdP authentication logging messages that include user ID and IP address as shown in **Fig. 9**. IdP authentication logging messages are useful to identify possible users on a computer security incident when another NAPT equipment is introduced in a laboratory or a room.

3.7 Mail Authentications

We implement IMAP/POP authentication for a mail service using dovecot. We then store dovecot logging messages as shown in Fig. 9. These messages are also useful when another NAPT equipment is introduced.

3.8 Groupware Logging Messages

We implement a groupware in order to share information

```
1,2019/02/05 00:50:04,013201001747,TRAFFIC,end,1,2019/02/05 00:50:04,10.15.5.150,133.167.77.169,160.15.XXX.XXX,
  133.167.77.169,EXT-FW_DEFAULT PERMIT,,,ntp,vsys1,kenkyu,external,ae1.3999,ae1.3010,Log-profile,2019/02/05 00:50:04,
  2905472,1,123,123,31020,123,0x400053,udp,allow,188,94,94,2,2019/02/05 00:49:32,30,any,0,6598660527738326902,0x0,
  10.0.0.0-10.255.255.255,Japan,0,1,1,aged-out,0,0,0,0,,imc-me1f-fw01,from-policy,,,0,,0,,N/A
1,2019/02/05 00:55:57,013201001747,TRAFFIC,end,1,2019/02/05 00:55:57,10.15.5.150,129.250.35.250,160.15.XXX.XXX,
  129.250.35.250,EXT-FW_DEFAULT PERMIT,,,ntp,vsys1,kenkyu,external,ae1.3999,ae1.3010,Log-profile,2019/02/05 00:55:57,
  273130,1,123,123,21997,123,0x400053,udp,allow,188,94,94,2,2019/02/05 00:55:25,30,any,0,6598660527738464346,0x0,
  10.0.0.0-10.255.255.255,United States,0,1,1,aged-out,0,0,0,0,,imc-me1f-fw01,from-policy,,,0,,0,,N/A
```

Fig. 1 An example of firewall traffic logging messages (PA-5220).

```
%FWSM-5-106100: access-list internal_access_in permitted udp VRF-Kenkyu/10.15.5.150(123) ->
                external/106.185.48.114(123) hit-cnt 1 (first hit) [0x1d40ffe4, 0xf7f9a091]
%FWSM-6-305011: Built dynamic udp translation from VRF-Kenkyu:10.15.5.150/123 to external:160.15.XXX.XXX/29043
%FWSM-6-302015: Built outbound UDP connection 144554821147310617 for VRF-Kenkyu:10.15.5.150/123 (160.15.XXX.XXX/29043)
                to external:106.185.48.114/123 (106.185.48.114/123)
```

Fig. 2 An example of firewall t

among organization members. We then store the groupware logging messages that include user ID and IP address as shown in **Fig. 11**. We can obtain logging messages of not only authentications but also some behaviors such as an event creation in a calendar. Groupware logging messages are also useful to identify possible users on a computer security incident when another NAPT equipment is introduced in a laboratory or a room.

4. Features of Logging Messages Required for a Computer Security Incident Response

Logging messages required for a computer security incident response have been introduced in **Sec. 3**. Ones may be able to find features of these logging messages as follows:

(1) all logging messages are in time series,
(2) logging messages are not strictly in ascendant or descendant order,
(3) newer logging messages should be able to be retrieved faster than older logging messages,
(4) older logging messages can take longer time to be retrieved because a long time has already passed after an older security event if the older security events caused an incident,
(5) logging messages are mainly added and rarely removed or changed,
(6) logging messages are almost in fixed and predefined formats, and
(7) most of values such as IP address, MAC address and user name which are required for a computer security incident response are in fixed lengths.

5. Logging System for Incident Response

This section proposes a new logging system. This section firstly overview the proposed logging system. This section then presents each features of the proposed logging system.

5.1 Overview of the Proposed Logging System

Fig. 12 depicts the overview of the proposed logging system. The logging system consists of multiple database servers. All multiple database servers have the same IP address for IP anycasting. Network equipment or other servers (e.g., Web server, mail server and so on) send logging messages to the database using

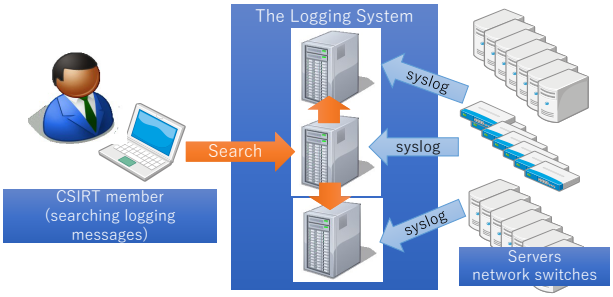

Fig. 12 Overview of the proposed logging system.

syslog protocol. The logging messages are then stored into one of database servers. When one, say a CSIRT member, searches for logging messages, one sends a search request to one of database servers. The database server receiving a request forward the request to the other database servers. All database servers then send responses back to one who sends a search request.

5.2 Binary Based Key Value Store

Existing logging systems are basically based upon text messages while logging messages of network or security equipment usually are in pre-defined text format. Existing logging systems, therefore, have overhead to handle text messages. The proposed logging system then stores binary values only in a record in a table, and text messages are indexed in another table. Because a logging message is usually output using printf functions, the logging system indexes a message format. A Logging message is then stored as a tuple of a message format index and variable values, i.e., variable arguments of printf. This section proposes how to convert logging messages in text format into ones in binary format. As described in **Sec. 3**, almost all logging messages are in fixed format. All logging messages then can be converted into binary format as follows:

(1) timestamp: can be represented as 128-bit number, which are usually stored in `struct timeval`.
(2) IPv4 address: can be represented as 32-bit number.
(3) IPv6 address: can be represented as 128-bit number.
(4) MAC address: can be represented as 48-bit number.
(5) port number: can be represented as 16-bit number.
(6) username: can be represented as 32-bit number by indices.
(7) string (fixed text): can be represented as 32-bit number by indices.

```
date=2019-02-05 time=00:48:04 devname="FG5H1E5818903568" devid="FG5H1E5818903568" logid="0004000021" type="traffic"
  subtype="sniffer" level="notice" vd="root" eventtime=1549295284 srcip=10.15.5.150 srcport=65213 srcintf="x1"
  srcintfrole="undefined" dstip=182.163.86.41 dstport=22 dstintf="x1" dstintfrole="undefined" sessionid=1024791349
  proto=6 action="accept" policyid=3 policytype="sniffer" service="SSH" dstcountry="Japan" srccountry="Reserved"
  trandisp="snat" transip=0.0.0.0 transport=0 duration=375584 sentbyte=33163251 rcvdbyte=2044 sentpkt=0 rcvdpkt=0
  appid=16060 app="SSH" appcat="Network.Service" apprisk="elevated" applist="sniffer-profile" sentdelta=2668 rcvddelta=0
date=2019-02-05 time=00:48:07 devname="FG5H1E5818903568" devid="FG5H1E5818903568" logid="0004000017" type="traffic"
  subtype="sniffer" level="notice" vd="root" eventtime=1549295287 srcip=10.15.5.150 srcport=123 srcintf="x1"
  srcintfrole="undefined" dstip=160.15.14.54 dstport=123 dstintf="x1" dstintfrole="undefined" sessionid=1048076579
  proto=17 action="accept" policyid=3 policytype="sniffer" service="NTP" dstcountry="Japan" srccountry="Reserved"
  trandisp="snat" transip=0.0.0.0 transport=0 duration=180 sentbyte=48 rcvdbyte=48 sentpkt=0 rcvdpkt=0 appid=16270
  app="NTP" appcat="Network.Service" apprisk="elevated" applist="sniffer-profile" utmaction="allow" countapp=1
  sentdelta=48 rcvddelta=48
```

Fig. 3 An example of firewall traffic logging messages (Fortigate).

```
{"Release":"V20L10NF0301B15", "ID":"5C5862AF04188100", "Occurred":"2019-02-05T01:05:03.237+09:00",
  "Location":"1f-node-room", "MessageID":"00730003", "IP":"160.15.XXX.XXX", "MAC":"00:12:XX:XX:XX:00",
  "RiskLevel":"Low", "Suspiciousness":30, "RelatedNode":[{"Type":"C&C Activity", "SrcIP":"160.15.xxx.xxx",
  "DstIP":"XXX.XXX.XXX.XXX", "SrcPort":53, "DstPort":59225, "Protocol":"UKT",
  "Domain":null, "URL":null}], "Message":"Suspicious Traffic: C&C Activity"}
{"Release":"V20L10NF0301B15", "ID":"5C5862B104188110", "Occurred":"2019-02-05T01:05:05.216+09:00",
  "Location":"1f-node-room", "MessageID":"00730003", "IP":"10.xxx.xxx.xxx" , "MAC":"00:12:XX:XX:XX:00",
  "RiskLevel":"Low", "Suspiciousness":60, "RelatedNode":[{"Type":"C&C Activity", "SrcIP":"10.XXX.XXX.XXX",
  "DstIP":"XXX.XXX.XXX.XXX", "SrcPort":65297, "DstPort":80, "Protocol":"HTTP",
  "Domain":"www.msftncsi.com", "URL":"http://www.msftncsi.com/ncsi.txt"}],
  "Message":"Suspicious Traffic: C&C Activity"}
```

Fig. 4 An example of firewall traffic logging messages (iNetSec).

(8) URL (variable text): can be represented as 128-bit number by indices.

Ones may be worried that variable texts can not be converted into binary format if the number of unique variable texts are larger than 2^{128}. Almost all logging messages are, however, in fixed format as described in **Sec. 3**, and the number of unique variable texts except for URL or FQDN may be small. Conclusive representations of URL and FQDN are future work. Excepf ro URL and FQDN, almost all logging messages can be converted into binary format. These conversions may reduce sizes of logging files. For example, an IPv4 address is represented as `X.X.X.X`, which requires at least 7 bytes in text format. On the other hand, an IPv4 address represented as 32-bit number, and more than 40% of the size in text format is reduced. These conversion may also reduce the retrieving time because the size of a logging file is reduced.

5.3 Time Series Database (TSDB)

Since each logging message must have a timestamp, the proposed logging database always stores the timestamp as a primary key. All records are basically stored in ascending order of timestamps. Some records are, however, not strictly in ascending order for lock-free operation described later. That is, a little bit old logging messages can be recorded after a newer messages. On a computer security incident response, we usually retrieve logging messages by specifying the timestamp. The timestamp of logging messages, however, can be different from the actual time due to time synchronization deviations among servers. To be more specific, time of a suspicious traffic reported by an external organization may be slightly different from the time in the logging message. Ones may, thus, retrieve logging messages by allowing timestamps deviations on a computer security incident response.

5.4 Timestamp Index

Regarding searching a record, a timestamp is usually specified for a computer security incident. In order to improve searching speed, a location of a record at a timestamp is indexed. Since the number of logging messages may depend upon daytime or night, timestamp index improves searching a record of a specified timestamp.

5.5 Fixed Record Length

In order to improve search and insertion performance, the proposed logging system has the same record length for a table as same as recent RDBMSes. To this end, a logging message is classified by a kind of equipment, e.g., network switches, access points, servers and so on. Each table is then created for each kind of logging messages. This feature of fixed record length may enables a lock-free addition.

5.6 Lock-Free Insertion and Search

Let us assume that multiple threads are running on the proposed logging server and each thread is indexed. A thread is in charge of receiving syslog messages and inserting them to the database. When a thread inserts a new logging messages, the thread inserts the messages into the next space in a memory or a disk which is indexed by the index of the tread. That is, storage to store logging messages is separated by each index of each thread.

On the other hand, a query to search for a logging message is broadcast to all threads. All thread then searches a logging message and return results. In this manner, there is no need to obtain a lock on an insertion and a search.

```
2019/02/05 06:10:13 srv_ip 10.ZZZ.ZZZ.ZZZ vlan 2005 user_ip 10.XXX.XXX.XXX
                    user_mac 00:50:YY:YY:YY:YY time 2019-02-04 21:10:01.789416
2019/02/05 06:15:12 srv_ip 10.ZZZ.ZZZ.ZZZ vlan 2005 user_ip 10.XXX.XXX.XXX
                    user_mac 00:50:YY:YY:YY:YY time 2019-02-04 21:15:01.467962
```

Fig. 5 An example of ARP table polling logging messages.

```
Feb 12 00:17:46 localhost arp_sniffer.py: 00:aa:XX:XX:XX:XX 10.XXX.XXX.XXX
Feb 12 00:17:46 localhost arp_sniffer.py: 00:aa:XX:XX:XX:XX 10.XXX.XXX.XXX
```

Fig. 6 An example of ARP snooping logging messages.

5.7 Lock-Free Clustering Support

The proposed logging database does not strictly consider an order of a logging message. Timestamps in records are, therefore, not always in ascending order. This nature may delay finishing all search in order to make sure that the all log messages in specified timestamp in search are examined. This nature, however, makes insertion and lookup operations lock-free. Lock-free clustering can be then archived.

5.8 Recent in Memory and Old in Disk

When a computer security incident happens and quick response is necessary, recent logging messages are searched in most cases. Older messages are not required to be fast to be searched because its search itself is enough delayed already. The proposed logging system then always keeps recent logging messages in memory as much as possible, and write the messages to a disk if possible or all memory is consumed. Even after older messages are searched once, these older messages are not in memory in order to prioritize quick response for a recent computer security incident.

6. Related Work

Elasticsearch [4] is very famous frame work for storing and visualizing logging messages. Elasticsearch stores logging messages by employing Fluentd [5]. Elasticsearch, however, considers logging messages only in text format, and requires a large size of storage and memory.

MongoDB [1] is a document database, and all data is usually represented in a JSON format. From the viewpoint of the programming, a JSON format is flexible, but requires more memory and storage sizes. In addition, a retrieval is slower than RDBMS.

Splunk [6] is also very famous frame work for collecting logging messages. Splunk can provide strong expressions to search for a specific logging message. Splunk, however, considers logging messages only in text format, and also requires a large size of storage and memory.

Abe H. et al. proposes Hayabusa [7] that considers logging messages in time series and can scale out. Hayabusa utilizes SQLite. Hayabusa, however, considers logging messages only in text format, and does not consider binary format.

Gnocchi [8] is one of time series database. Gnocchi employs PostgreSQL for indices. Gnocchi considers logging messages only in text format, and then requires a large size of storage and memory.

7. Concluding Remarks

This paper has introduced logging messages that are required for detecting and/or handling a computer security incident. This paper has proposed a new logging system dedicated for a computer security incident response. This paper has also shown that these logging messages are in text format and can be converted into binary format. These simple conversion can reduce not only a size of a storage for logging messages but also reduce retrieving time from a large amount of logging messages. Implementation and evaluation are our future work.

References

[1] MongoDB, Inc.: mongoDB, https://www.mongodb.com/ (2018). Accessed: 2018/10/07.
[2] Paloalto Networks, Inc.: Traffic Log Fields, https://docs.paloaltonetworks.com/pan-os/8-0/pan-os-admin/monitoring/use-syslog-for-monitoring/syslog-field-descriptions/traffic-log-fields.html# (2019). Accessed on 2019/4/15.
[3] Ohmori, M., Miyatal, N. and Suzuta, I.: AXARPS: Scalable ARP Snooping Using Policy-Based Mirroring of Core Switches, *Proc. the 33rd International Conference on Advanced Information Networking and Applications (AINA-2019)*, pp. 667–676 (2019).
[4] Elasticsearch B.V.: Elasticsearch, https://www.elastic.co/jp/products/elasticsearch (2019). Accessd: 2019/4/16.
[5] Fluentd Project: fluentd, https://www.fluentd.org/ (2010). Accessed: 2018/10/07.
[6] Splunk Inc.: splunk, https://www.splunk.com/ (2019). Accessed on 2018/1/11.
[7] Abe, H., Shima, K., Miyamoto, D., Sekiya, Y., Ishihara, T., Okada, K., Nakamura, R., Matsuura, S. and Shinoda, Y.: Design and Evaluation of Scalable Syslog Search Engine Optimized for Time Dimensional Search Operation, *Journal of Information Processing*, Vol. 60, No. 3, pp. 728–737 (2016).
[8] the gnocchi developers: Gnocchi Metric as a Service, https://gnocchi.xyz/ (2019). Accessed on 2019/4/15.

```
client 10.XXX.XXX.XXX#44650 (buffalo.jp): query: buffalo.jp IN A + (160.15.XXX.XXX)
client 10.XXX.XXX.XXX#51449 (tumail.center.tottori-u.ac.jp): query: tumail.center.tottori-u.ac.jp IN A + (160.15.XXX.XXX)
```

Fig. 7 An example of DNS query logging messages

```
Accept: [XXXXX@tottori-u.ac.jp] MAC: a4:34:XX:XX:XX:XX (Win 10), SSID: eduroam, AP: 00:1a:XX:XX:XX:XX (gen-ee3f-ap03),
        NAS: imc-me1f-wc01, VLAN: none from 172.16.XX.XXX
Accept: [69854274] MAC: 5c:f9:XX:XX:XX:XX (OS X), SSID: imcwlan, AP: 00:1a:XX:XX:XX:XX (imc-me2f-ap01),
        NAS: imc-me1f-wc01, VLAN: none from 172.16.XX.XXX
```

Fig. 8 An example of IEEE802.1x authentication logging messages.

```
[Shibboleth-Audit.SSO: 241] 20190416T004831Z|urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect|
_ff9ed1b088d2490bb6865994cc0cd69d|https://moodle.center.tottori-u.ac.jp/shibboleth-sp|
http://shibboleth.net/ns/profiles/saml2/sso/browser|https://idp.tottori-u.ac.jp/idp/shibboleth|
urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST|_a73feaeb92602087da1818947f2a90da|USERNAME|
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport|uid,jasn,transientId,jaGivenName|
AAdzZWNyZXQxgw2rWdcD0EYwZ3OVve1pqiqC1j5PDpnFmYxduiuPWiAmVhRn3S7U7jvNJH0GtB0gBX2ZIZb7vH0JZYj+qoRQ4nG
EngnSHyZeksQfVDFE+pr//xCMcuqFtV8oNCbJwqDRR2oCvOJptBqsGZ9WJpcFeFEkwShFVltNJPY=|
_dcdbd24b69e6d852ed150f79077e4d2b|IPADDRESS|080142769A319D4AB74714CABBD6A70B|
```

Fig. 9 An example of Shibboleth IdP logging messages.

```
dovecot: imap-login: Login: user=<XXXXX>, method=PLAIN, rip=10.XXX.XXX.XXX, lip=10.XXX.XXX.XXX,
         mpid=23239, secured, session=<cpI61JqGNuAKDwgq>
dovecot: imap(XXXXX): Logged out in=436 out=9434
dovecot: imap-login: Login: user=<YYYYY>, method=PLAIN, rip=10.XXX.XXX.XXX, lip=10.XXX.XXX.XXX,
         mpid=23240, secured, session=<Na8+1JqG0sMKDwgp>
dovecot: imap(YYYYY): Logged out in=123 out=1091
```

Fig. 10 An example of mail authentication logging messages (dovecot).

```
10.XXX.XXX.XXX:Motoyuki OHMORI(username):grn.common:notice:[login] system (id:XXX, name:'Motoyuki OHMORI', account:username)
10.XXX.XXX.XXX:Motoyuki OHMORI(username):grn.schedule:notice:[create] event (eid:343XXX, event_title:'title')
```

Fig. 11 An example of groupware logging messages.