WAF Signature Generation with Real-Time Information on the Web

1st Masahito Kumazaki
Graduate School of Informatics
Nagoya University
Nagoya, Japan
Email: kumazaki@net.itc.nagoya-u.ac.jp

2nd Yukiko Yamaguchi
Information Technology Center
Nagoya University
Nagoya, Japan
Email: yamaguchi@itc.nagoya-u.ac.jp

3rd Hajime Shimada
Information Technology Center
Nagoya University
Nagoya, Japan
Email: shimada@itc.nagoya-u.ac.jp

4th Hirokazu Hasegawa
Information Security Office
Nagoya University
Nagoya, Japan
Email: hasegawa@icts.nagoya-u.ac.jp

***Abstract*—Zero-day attacks and attacks based on publicly disclosed vulnerability information are one of the major threats in network security. To cope with such attacks, it is important to collect related information and deal with vulnerabilities as soon as possible. Therefore, we propose a system that collects vulnerability information related to Web applications from real-time information on the Web and generates Web Application Firewall (WAF) signatures. In this paper, at first, we collected vulnerability information containing the specified keyword from the National Vulnerability Database (NVD) data feed and generated WAF signatures automatically. Then, we confirmed the possibility of WAF signature generation from one tweet. Finally, we extracted tweets that may contain vulnerability information and labeled them according to the filtering algorithm. From these results, we could prove the efficiency of the proposed system.**

Keywords–Web Application Firewall(WAF); Zero-day Attack; Vulnerability Information; Real-time Information.

I. INTRODUCTION

Web applications are recognized as an important part of the social infrastructure and we use various Web applications every day. On the other hand, cyberattacks are increasing year by year and are widely recognized as an obstacle to social infrastructure. There are many cases of serious damage such as classified information leak by unauthorized access and attacks using vulnerabilities [1] [2].

Among cyberattacks, attacks that used published vulnerabilities are especially increasing [3] [4]. As an actual case, the number of detected attacks for Apache Struts 2 has increased immediately after the announcement of its vulnerability [5], and some of them resulted in personal information leakage due to lack of necessary countermeasures such as timely system update [6] [7]. Another major information security issue, zero-day attacks that occur before both a release of vulnerability information and a provision of patches [18]. For example, Google Chrome suffered such a zero-day attack in 2019 [9].

Generally recommended countermeasure against such cyberattacks is to apply fixed patches distributed by the vendor on time. However, there is an unprotected period against cyberattacks until the patch has released if the attack is a zero-day attack.

In this paper, we propose a Web Application Firewall (WAF) signature generation system using real-time information on the Internet to mitigate zero-day attack problems. Real-time information sources like twitter are used for a variety of purposes [10] [11]. They may contain the latest vulnerabilities and emergency plans, which are useful to mitigate the problem before formal vulnerability information and patches are released. Therefore, in this study, we propose a system that automatically collects vulnerability information to construct new WAF signatures to mitigate the problems until the release of formal countermeasures. Although there are previous studies on the automatic generation of signatures for Intrusion Detection System (IDS) [12] [13] [14], we use WAF in this study because it targets web applications. To acquire the latest vulnerability information from the Web, the system collects vulnerability information from real-time data feeds such as a Social Networking Service and websites for discussions on security technologies. After performing data cleansing on the collected data, the system checks for associated vulnerable Web applications and generates WAF signatures for them as a virtual patch.

In this paper, we extracted the vulnerability information including the specified keyword from the vulnerability information provided by the National Vulnerability Database (NVD) [15], and automatically generated the signature of WAF, as a proof of the concept. Then, we collected vulnerability information from Twitter, which is one of the well-known real-time information sources, and attempted to generate WAF signatures from tweets. Finally, we extracted and filtered tweets using a manual approach. From the results, we could prove the efficiency of the proposed system.

In the following, in section II, we describe the background of this study, such as exisiting countermeasures. In section III, we describe our proposed system and architectures which implement this system. In section IV, we describe the experiments using the inplemented system. In section VI, we discuss additional real-time information sources. Finally, we summarize this paper in section VI.

II. BACKGROUND

Existing countermeasures against zero-day attacks include defense-in-depth solutions that combine multiple security appliances such as firewalls, IDS/Intrusion Prevention System (IPS) based allow/deny list, and so on. However, these countermeasures may result in an unprotected period against attacks if the countermeasure is based on static rules. To mitigate the damage caused by the zero-day attacks, we propose a WAF signature generation system using real-time information on the Web. The proposed system generates a WAF signature for blocking access to the vulnerable Web application when the system finds vulnerability information of the Web application from real-time information on the Internet. As a result, the unprotected time against attacks is shortened and the damage may be mitigated.

 ISBN: 978-1-61208-821-1

A. Web Application Firewall (WAF)

Web applications are becoming more complicated year by year so that it is getting harder to detect vulnerabilities from Web application implementations. Therefore, WAF is used as a security measure to protect Web applications from ingress traffic and mitigate attacks that exploit vulnerabilities [16]. The WAF could be installed at multiple locations, such as a host type installed on a Web server, a network type installed on a communication path to the Web server, and a cloud type using WAF services on a cloud provided by the cloud service provider. By setting rules to prevent attacks aiming at typical Web application vulnerabilities, we can protect the Web server from attacks that used the typical vulnerability. Some WAFs allow you to set your own rules for specific attacks so that it is possible to prevent the zero-day attacks by applying custom signatures. The WAF uses the following basic functions to prevent external attacks and notify the administrator.

Analyzing
: Analyze Hypertext Transfer Protocol (HTTP) communication based on the detection pattern defined as to allow/deny list.

Processing
: Performs pass-through processing, error processing, replacement processing, blocking processing, and so on. Judgement is based on the result of the analysis function.

Logging
: Record WAF activity. An audit log records the unauthorized HTTP communication detected and its processing method. An operation log records WAF operation information and error information.

B. ModSecurity

ModSecurity is an open-source host type WAF software provided by Trustwave. ModSecurity has the following functions.

- Recording and auditing whole HTTP traffic
- Real-time monitoring of HTTP traffic
- Flexible enough rule engine to act as an external patch for Web applications
- Can be embedded as a module of Web server software Apache, IIS, and Nginx

Also, the Open Web Application Security Project (OWASP) [17] provides the Core Rule Set (CRS) that includes signatures for typical cyberattacks for ModSecurity. In this study, we use ModSecurity as WAF for our proposed system in later experiments, considering the flexibility of the rule engine and the versatility of the module.

III. Proposed system

We assume that The organization utilizing proposed system is running WAF and the administrator can customize the rule of this WAF.

The proposed system consists of four modules:

(1) Collection module
(2) Cleansing module
(3) Signature generation module
(4) Notification generation module.

TABLE I. REAL-TIME INFORMATION SOURCES

	API	Identification	Timestamp
Twitter [18]	✓	✓	✓
Stack Overflow [19]	✓	✓	✓
Reddit [20]	✓	✓	✓
teratail [21]	×	✓	✓
Security StackExchange [22]	✓	✓	✓

Figure 1 shows the architecture of the proposed system and its data flow.

First, the collection module collects vulnerability information from real-time information sources such as social networks (Fig. 1 (1)). After that, the proposed system performs data cleansing on the collected data (Fig. 1 (2)). Finally, the proposed system generates WAF signatures and set them (Fig. 1 (3)). At the same time, the proposed system generates a notification file for the administrator (Fig. 1 (4)).

In this system, it is assumed that the administrator registers the names of the Web applications and its version information into the system beforehand. Based on the registered information, the system extracts vulnerability information from the Internet.

A. Collection module

The proposed system generates WAF signatures from real-time information shown in Table I. Subsequent processes will need IDs and timestamps to identify the articles and the date of publication. As shown in Table 1, these sources are equipped with IDs and timestamps. The collection module collects vulnerability information from these sources and saves their ID, timestamp, and body.

B. Cleansing module

The proposed system performs data cleansing on collected data to remove duplicate information and get the necessary information. The cleansing module extracts the following attributes from text and Web page. The system uses the following attributes for generating WAF signatures.

- Application name
- Vulnerability type
- Version information
- Vulnerability identification information such as a Common Vulnerabilities and Exposures (CVE)-ID

C. Signature generation module

If the Web application name which is used in operating Web application system is included in attributes from the cleansing module, the system generates a WAF signature to block HTTP requests to that Web application and apply it into the WAF. In addition, the signature generation module notifies the notification generation module regarding the signature generated.

D. Notification generation module

The proposed system generates a notification to the administrator. This notification includes information such as the name of the vulnerable web application and its version. We expect that when the vulnerability is resolved, for example by applying a patch, the administrator will use this notification to remove the signature.

 ISBN: 978-1-61208-821-1

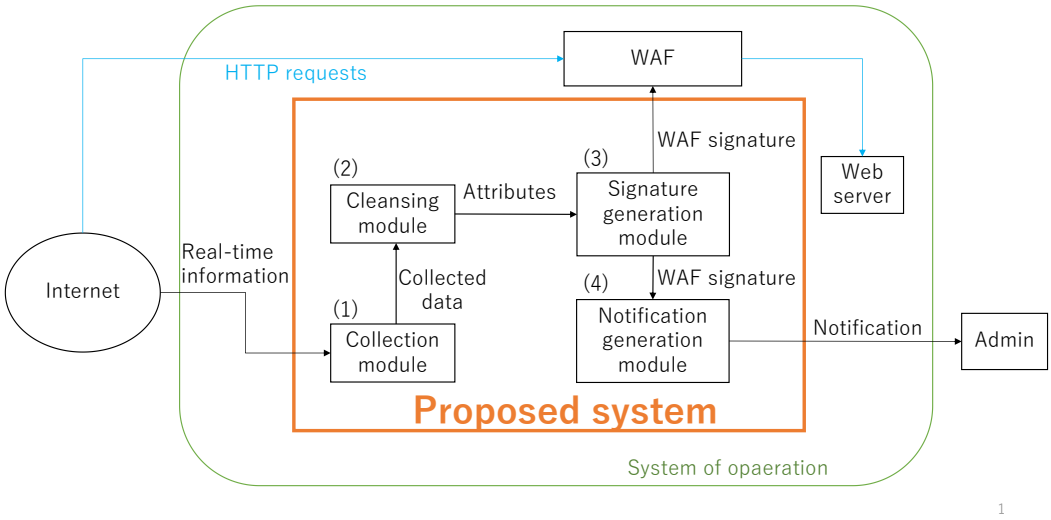

Figure 1. System architecture and data flow.

IV. Experiments

In this study, we conducted the following evaluation experiments. As a Web application to collect vulnerability information, we used WordPress as a keyword for extracting vulnerability information due to its known history of many vulnerabilities.

A. Experiment 1: WAF signature generation using CVE information

As a proof of concept, we implemented the proposed system shown in Figure 1 with Python 3.6.8 and AWK scripts and examined the automatic generation of WAF signatures using NVD data feeds instead of real-time information.

1) Processing Method: In the collection module (Fig.1(1)), we obtained NVD data feed on a daily basis. This data feed contains CVEs. In the cleansing module (Fig.1 (2)), we checked whether the keyword was included in the Common Platform Enimeration (CPE) name for each vulnerability information of the data feed. After that, following data has been extracted.

1) CVE-ID
2) CPE name
   CPE name is a name that identifies the platforms [23].
   cpe:2.3:[Part]:[Vendor]:[Product]:[Version]
   :[Update]:[Edition]:[SW_Edition]:[Target_SW]
   :[Target_HW]:[Language]:[Other]
   In this experiment, we used [Part] and [Product] from the CPE name. [Part] represents a product type with one character, where 'a' is an application, 'o' is an operating system, and 'h' is hardware. [Product] is the product name.
3) Version information

We extracted these data and stored them into JavaScript Object Notation (JSON) format.

In signature generation module (Fig.1 (3)), we generated a signature for ModSecurity from extracted information to prevent access to the Web application. With reference to `ModSecurity_41_xss_attacks.conf` and `ModSecurity_41_sqlinjection_attacks.conf` from ModSecurity's CRS, we added following signatures to ModSecurity.

- **VARIABLES:** REQUEST_COOKIES | !REQUEST_COOKIES:/__utm/ | REQUEST_COOKIES_NAMES | ARGS_NAMES | ARGS | XML:/
- **OPERATOR:** Regular expression using Web application names
- **ACTIONS:** phase:2,block,msg:'*application name* injection.',severity:'2',id:'15000+line number'

In addition, we generated text files to notify the signature generation in notification generation module (Fig.1 (4)).

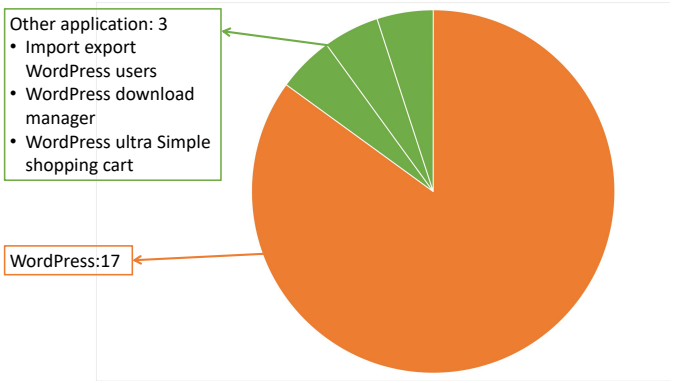

Figure 2. Extracted information in Experiment 1

2) Result: We performed this daily process for 10 days from 21st December, 2019 to 30th December, 2019.

Since the results for all days during the collection period was same, a single day result is depicted as a sample in Figure 2.

The result of automatically generated WAF signatures is shown in Figure 3. The signature for WordPress was generated from 17 cases of WordPress vulnerability information and other signatures were generated from 1 case corresponding to each application.

 ISBN: 978-1-61208-821-1

```
1 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    import_export_wordpress_users" "phase:2,block,msg:'
    WordPress injection.'severity:'2',id:'15001'"
2 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress" "phase:2,block,msg:'WordPress injection.'
    severity:'2',id:'15002'"
3 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress_download_manager" "phase:2,block,msg:'
    WordPress injection.'severity:'2',id:'15003'"
4 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress_ultra_simple_paypal_shopping_cart" "phase:2,
    block,msg:'WordPress injection.'severity:'2',id
    :'15004'"
```

Figure 3. Generated WAF signature in Experiment 1

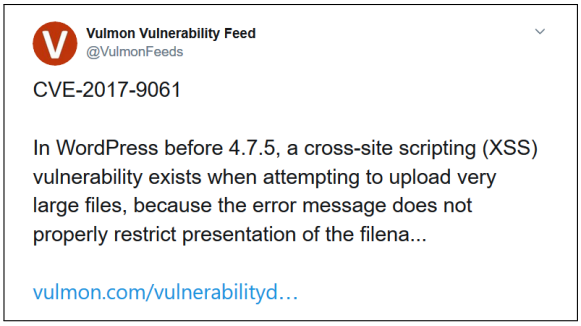

Figure 4. The tweet used in Experiment 2

TABLE II. EXTRACTED ATTRIBUTES OF THE TWEETS

key	Type	Description
id	Int64	tweet_id as an integer.
created_at	String	The time when the tweet was created.
username	String	User name who posted the tweet.
text	String	The tweet contents in UTF-8 format.
urls	List	Uniform Resource Locators (URLs) in the tweet[1].

3) Consideration: From the generated signatures it could be seen that in addition to the signature for actual WordPress application, other applications that include " wordpress " in their name have been blocked. These redundant rules give an additional burden to the WAF. To overcome this limitation, we need to improve the signature generation rule and tune it.

B. Experiment 2: Generation of WAF signatures using twitter feed

To confirm the possibility of WAF signature generation from real-time vulnerability information, we collected tweets from twitter feeds, chose one tweet, and generated a WAF signature from it. We collected tweets using the search Application Programming Interface (API) by setting query parameters to "wordpress" and chose the tweet shown in Figure 4. The proposed system uses information listed in Table II from tweets, so we extracted the following information.

- **id:** 1200259525707796482
- **created_at:** 2019-11-29 03:45:45
- **username:** Vulmon Vulnerability Feed
- **text:** CVE-2017-9061\n\nIn WordPress before 4.7.5, a cross-site scripting (XSS) vulnerability exists when attempting to upload very large files, because the error message does not properly restrict presentation of the filena...\n\nhttps://t.co/anaw5-QSAoa"
- **urls:** [http://vulmon.com/vulnerabilitydetails?qid=CVE-2017-9061]

```
1 SecRule REQUEST_COOKIES|!REQUEST:/__utm/|
    REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* "
    wordpress" "phase:2,brock,msg:'WordPress XSS.'severity
    :'2',id:'15001'"
```

Figure 5. Generated WAF signature in Experiment 2

1) Method and Result: As a result of a visual check of the Web application and vulnerability information contained in the tweet, we got the following attributes from the text.

- **Application name:** WordPress
- **Vulnerability type:** XSS
- **Version information:** 4.7.5 and earlier
- **CVE-ID:** CVE-2017-9061

We also checked the Web page indicated by the URL, but we couldn't get any more attributes. Similar to experiment 1, we generated the ModSecurity signature from these attributes. The generated signature is shown in Figure 5.

2) Consideration: From the experiment, it was confirmed that WAF signatures could be generated from collected tweets. However, a method to select a relevant tweet is needed. We expect to be able to extract information such as Web application name, its version, and type of vulnerability through the pattern matching approach.

C. Experiment 3: Filtering and extracting tweets through pattern matching approach

Based on the results of Experiment 2, we extracted and filtered vulnerability information from the tweets. Through twitter API We collected tweets every day for the following period. After eliminating the duplicates of the collected tweets, we further processed 1,116 tweets.

- **Collection period:** From 4th December, 2019 to 8th January, 2020
- **Search query:** "WordPress AND Vulnerability", "WordPress AND XSS", and "WordPress AND injection"

To check the results of the filter, we manually assigned the following labels to the tweets based on the relevance to the WordPress vulnerability.

0: WordPress vulnerability information
1: Other information

As a result of the manual labeling, 76 tweets regarding WordPress vulnerabilities have been identified. The remaining 1,040 tweets were mistakenly extracted since its URL referred to the Webpage created using WordPress, for example.

 ISBN: 978-1-61208-821-1

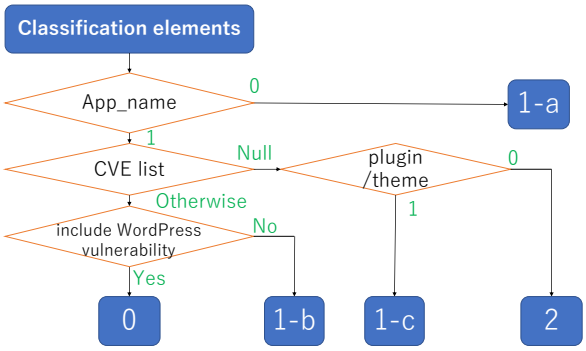

Figure 6. Flowchart of estimated label setting

1) Method: At first, we collected the attributes shown in Table II, from the tweet and then filtered them on its text context by pattern matching using regular expressions to the tweet's text and web page body which is indicated by URL. The following attributes are extracted from the tweet and stored in JSON format.

- **ID:** tweet-id
- **App_name:** If the string includes "wordpress" this element is 1, otherwise 0.
- **CVE:** Extract the string "CVE-\d{4}-\d+" and store it as a list.
- **plugin / theme:** If the string includes "plugin" or "theme" this parameter is 1, otherwise 0.

From these attributes and a list of CVEs corresponding to the target application, we automatically assigned an estimated label to tweets according to the flowchart shown in Figure 6. Each label indicates the following categorization result.

0: WordPress vulnerability information
1: Other information
1-a: Not included the string "wordpress"
1-b: Expected as a WordPress plugin or theme
1-c: No WordPress vulnerability in CVE list
2: Unfiltered by this method

2) Result: We implemented the filter in Python as per the flowchart shown in Figure 6 and ran it on the collected 1,116 tweets. The result is shown in Table III. This method filtered 597 tweets and 98.5% of them were filtered correctly. On the other hand, 519 were unfiltered. By using the pattern matching approach, the number of objects of analysis could be reduced by half.

Out of 7 tweets that have been failed to be correctly filtered, 6 tweets were supposed to be labeled as 0 whereas it was mistakenly labeled as 1-b. they were weekly summaries of vulnerability information for WordPress and related modules. They contained information about WordPress and its plugins at the same time. Therefore, the filter misjudged them as information about WordPress plugins.

3) Consideration: The information required for the proposed system is vulnerability information that is up-to-date or has not been officially announced, which is included in the tweets labeled as 2 in this experiment. Therefore, there needs to be a way to extract the required information from these tweets.

TABLE III. FILTERING RESULTS

		estimated label					
		0	1-a	1-b	1-c	2	total
correct label	0	13	0	6	0	57	76
	1	1	94	292	191	462	1,040
	total	14	94	298	191	519	1,116

TABLE IV. NUMBER OF SEARCH HITS

	2018-20148	2019-17669	2019-20041
Stack Overflow	10(-)	6(-)	10(-)
Reddit	2(-)	15(-)	11(-)
teratail	6(-)	6(-)	3(-)
Security StackExchange	2(-)	0(-)	2(-)

(): Number of search hits related CVEs

V. CONSIDERATION OF ADDITIONAL SOURCES

Currently, the real-time information source is only Twitter and it could be prone to be disinformation. Therefore, we discussed the possibility of using other information sources which are shown in Table I.

To analyze whether those information sources are moderate sources or not, we explored discussions about following three WordPress vulnerabilities in those communities. These vulnerabilities are registered in the NVD and have a high Common Vulnerability Scoring System (CVSS) score.

- CVE-2018-20148 Published: 14th December, 2018
- CVE-2019-17669 Published: 17th October, 2019
- CVE-2019-20041 Published: 27th December, 2019

Since we cannot collect information from teratail via API, we searched the above vulnerabilities by Google search with queries "WordPress" and "vulnerability". The duration of the search was set to one month before and after the vulnerability announcement. The results are shown in Table IV. The numbers in parentheses are the number of search hits. The result shows that we could not obtain information about these vulnerabilities from these communities in a timely manner.

Since the results shown in Table IV are not suitable for a comparison of each knowledge community, we tried additional exploration. We compared the number of search hits per site using the Custom Search API provided by Google to see the number of discussions about vulnerabilities in each knowledge community. We set the query "vulnerability" for all sites, and the period is from 1st January, 2017 to 31st December, 2019. Since Reddit has a lot of topics that are not security-related, we only explored two subreddits, "security" and "cybersecurity".

The results are shown in Table V. Among the knowledge communities explored in this study, Stack Overflow and Security StackExchange are the most active in discussions about the

TABLE V. NUMBER OF SEARCH HITS IN EACH KNOWLEDGE COMMUNITY

knowledge community\Year	2017	2018	2019	total
Stack Overflow	2,166	2,072	1,837	6,075
Reddit[cybersecurity]	31	172	266	469
Reddit[security]	16	60	151	227
teratail	241	196	172	609
Security StackExchange	1,166	1,072	768	4,006

 ISBN: 978-1-61208-821-1

vulnerability so that they can be used as a good information source.

VI. CONCLUSION

In this study, we proposed the WAF signature generation system using real-time information on the Internet and conducted three types of experiments as initial studies. From a filtering experiment for 1,116 Tweet data, we were able to narrow down the required data to half of the total data. We also discovered the following challenges in those three experiments.

- How to clearly distinguish vulnerability information of other Web applications which may have a similar name as Web application name
- How to select the necessary information effectively from vulnerability information

In addition, the following challenges may occur when implementing the whole proposed system.

- How to determine the disinformation
- What to do with vulnerability information for which version information could not be extracted
- Block legitimate Web applications which include the name of the target application in their names
- React other than blocking based on the type of vulnerability
- Generalize of the system to other than WordPress

In order to solve these problems, we will consider and verify the following approaches.

- Defining reliability based on the account which posted information to the information sources
- Creating additional signatures to block the target only

REFERENCES

[1] Significant Cyber Incidents — Center for Strategic and International Studies https://www.csis.org/programs/technology-policy-program/significant-cyber-incidents [retrieved: July, 2020]

[2] The 15 biggest data breaches of the 21st century | CSO Online https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html [retrieved: August, 2020]

[3] 10 Major Security Threats 2019 — Information-Technology Promotion Agency (IPA), Japan https://www.ipa.go.jp/files/000076989.pdf [retrieved: July, 2020]

[4] Think Fast: Time Between Disclosure, Patch Release and Vulnerability Exploitation — Intelligence for Vulnerability Management, Part Two | FireEye Inc https://www.fireeye.com/blog/threat-research/2020/04/time-between-disclosure-patch-release-and-vulnerability-exploitation.html [retrieved: August, 2020]

[5] Attacks Heating Up Against Apache Struts 2 Vulnerability | Threatpost https://threatpost.com/attacks-heating-up-against-apache-struts-2-vulnerability/124183/ [retrieved: July, 2020]

[6] US House of Representatives Committee on Oversight and Government Reform, "The Equifax Data Breach", Majority Staff Report 115th Congress, 2018. https://republicans-oversight.house.gov/wp-content/uploads/2018/12/Equifax-Report.pdf [retrieved: July, 2020]

[7] GMO Payment Gateway, "Apology and Report for Leak of Personal InformationDueto Unauthorized Access", 2017. https://www.gmo-pg.com/en/corp/newsroom/pdf/170310_gmo_pg_en.pdf [retrieved: July, 2020]

[8] Committee on National Security Systems, "Committee on National Security Systems(CNSS) Glossary", CNSSI No. 4009, 2015

[9] Chrome Releases: Stable Channel Update for Desktop https://chromereleases.googleblog.com/2019/03/stable-channel-update-for-desktop.html [retrieved: July, 2020]

[10] T. Sakaki, O. Makoto, and M. Yutaka, "Earthquake shakes Twitter users: real-time event detection by social sensors.", In Proceedings of the 19th international conference on World wide web, p. 851-860, 2010

[11] O. Oh, M. Agrawal, and H. R. Rao, "Information control and terrorism:Tracking the mumbai terrorist attack through twitter.", Information-Systems Frontiers, vol. 13, no. 1, pp. 33–43, 2011

[12] S. More, M. Matthews, A. Joshi, and T. Finin, "A Knowledge-Based Approach To Intrusion Detection Modeling.", In Proceedings of 2012 IEEE Symposium on Security and Privacy Workshops, pp. 75-81, May 2012.

[13] C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots.", SIGCOMM Computer Communication Review., Vol. 34, No. 1, pp. 51-56, January 2004.

[14] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated Worm Fingerprinting.", In Proceedings of the 6th Symposium on Operating Systems Design and Implementation, pp.4-4, December 2004.

[15] NVD - Home https://nvd.nist.gov/ [retrived: October, 2020]

[16] K. Lawrence and L. Luo, "Interactive management of web application firewall rules.", U.S.Patent, No. 9,473,457, 2016

[17] OWASP Foundation | Open Source Foundation for Application Security https://owasp.org/ [retrived: October, 2020]

[18] Twitter https://twitter.com [retrived: October, 2020]

[19] Stack Overflow - Where Developers Learn, Share, & Build Careers https://stackoverflow.com [retrived: October, 2020]

[20] reddit: the front page of the internet https://www.teratail.com [retrived: October, 2020]

[21] teratail teratail.com [retrived: October, 2020]

[22] Information Security Stack Exchange https://security.stackexchange.com/ [retrived: October, 2020]

[23] B. Cheikes, D. Waltermire, and K. Scarfone, "Common platform enumeration: naming specification version 2.3", NIST Interagency Report 7695, pp.11-13, 2011.

 ISBN: 978-1-61208-821-1