J Inf Process Syst, Vol.17, No.4, pp.772~786, August 2021
https://doi.org/10.3745/JIPS.03.0162

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Model for Illegal File Access Tracking Using Windows Logs and Elastic Stack

Jisun Kim*, Eulhan Jo**, Sungwon Lee***, and Taenam Cho****

Abstract

The process of tracking suspicious behavior manually on a system and gathering evidence are labor-intensive, variable, and experience-dependent. The system logs are the most important sources for evidences in this process. However, in the Microsoft Windows operating system, the action events are irregular and the log structure is difficult to audit. In this paper, we propose a model that overcomes these problems and efficiently analyzes Microsoft Windows logs. The proposed model extracts lists of both common and key events from the Microsoft Windows logs to determine detailed actions. In addition, we show an approach based on the proposed model applied to track illegal file access. The proposed approach employs three-step tracking templates using Elastic Stack as well as key-event, common-event lists and identify event lists, which enables visualization of the data for analysis. Using the three-step model, analysts can adjust the depth of their analysis.

Keywords

Active Directory, Digital Forensics, Elastic Stack, Microsoft Windows Log, Security, Shared Folder

1. Introduction

Digital forensics is used to identify a malicious crimes suspect or forbidden behaviors on systems or networks, and to track evidences of such actions. Malicious users can attack a target system in many ways. The technique may not necessarily be a technical one. In accordance with the network environment, forensic research that is tailored to various environments, such the Internet of Things (IoT) and cloud computing, is being actively conducted [1,2].

Digital evidences tend to be widely scattered, and because such evidences are intermingled with other traces, identifying the key evidences manually is labor-intensive and tedious. Therefore, the experience of experts is required. For conducting digital forensics, various tools are currently available, including EnCase [3], Forensic Toolkit (FTK) [4], and AXIOM [5]. These tools categorize data in various ways to support digital forensic analysis. However, it is not possible for these tools to support all the various devices and applications currently in use. Moreover, although these tools can provide imaging of the disk containing evidence traces, or gathering and categorizing data in storage, they cannot provide tracking scenarios of specific events. Furthermore, these tools do not provide visualization; rather, they offer only a text-based user interface in explorer form. It is consequently difficult to visualize at a glance the

Manuscript received March 23, 2020; first revision June 22, 2020; accepted June 25, 2020.
Corresponding Author: Taenam Cho (tncho@ws.ac.kr)
* Igloo Security Inc., Secoul, Korea (rlawltjs122@gmail.com)
** BidCoaching Research Institute, Daejeon, Korea (joeulhan@gmail.com)
*** Izerone Digital Forensics Company, Jeonju, Korea (jema10@hanmail.net)
**** Dept. of IT and Electronics Engineering, Woosuk University, Jincheon, Chungbuk, Korea (tncho@ws.ac.kr)

correlation of various pieces of evidences.

Microsoft Windows (hereinafter Windows) is the most widely used operating system [6], constituting 54.6% of the global operating system market [7]. It offers the capability of storing various system logs for auditing. We can save events by setting audit policy, and can check the events using Windows Event Viewer. However, many events occur in Windows for even simple tasks, such as opening a file, and it is therefore difficult to identify the user actions from the events. This is because the stored logs often do not have information about the action that caused the events. Moreover, the events that one action triggers differ from case to case. The filtering structure of Event Viewer is too simple to filter out events of interest when multiple users performed multiple tasks in various ways.

Security is not limited to server or network attacks by professional attackers. Auditing and tracking of computer files are necessary for protecting trade secrets, preventing unauthorized access to information or resources, and preventing data manipulation [8]. Studies to protect the integrity and ownership of files have been conducted [9]. However, this paper focuses on accessing files in shared folders on servers under active directory (AD) [10]. Kim et al. [11] proposed a technique for extracting from complex Windows logs the key events that can track specific actions. In the present study, we organized the log classification into three steps by segmenting and extending the classification. This structure enables the analyst to adjust the trace depth in three steps. We additionally suggest dashboard templates using Elastic Stack to visualize the tracking process.

The remainder of this paper is as follows. Section 2 discusses the Windows audit system. Section 3 introduces Elastic Stack. Section 4 presents and discusses the proposed event trace model. Section 5 demonstrates event tracking using the proposed model. Section 6 provides conclusions and future work.

2. Windows Audit System

The security audit policy settings under Security Settings\Local Policies\Audit Policy provide broad security audit capabilities for client devices and servers that cannot use advanced security audit policy settings. The basic audit policy settings are audit account logon events, audit account management, audit directory service access, audit logon events, audit object access, audit policy change, audit privilege use, audit process tracking, and audit system events [12].

2.1 Windows Audit Policy

Windows records and manages event logs in six categories: Account Logon, Account Management, Detailed Tracking, DS Access, Logon/Logoff, and Object Access. Our research is interested in Account Logon, Logon/Logoff, Object Access categories and their subcategories. Table 1 describes categories and subcategories [13].

Table 1. Categories and subcategories of interest

Category	Subcategory
Account Logon	Credential Validation, Kerberos Authentication Service, Kerberos Service Ticket Operations, Other Logon/Logoff Events
Logon/Logoff	Account Lockout, User/Device Claims, IPsec Extended Mode, Group Membership, IPsec Main Mode, IPsec Quick Mode, Logoff, Logon, Network Policy Server, Other Logon/Logoff Events, Special Logon
Object Access	Certification Services, Detailed File Share, File Share, File System, Filtering Platform Connection, Filtering Platform Packet Drop, Handle Manipulation, Kernel Object, Other Object Access Events, Removable Storage, Central Access Policy Staging

As shown in Fig. 1, we can set policies to log the events that are required for auditing [14]. The figure depicts an example of setting “Audit object access” to “Success, Failure” to record the events for the shared file that is the target of our study.

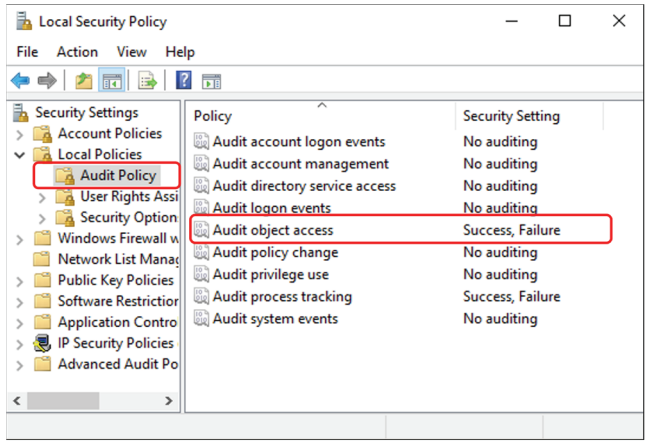

Fig. 1. Windows audit policy setting.

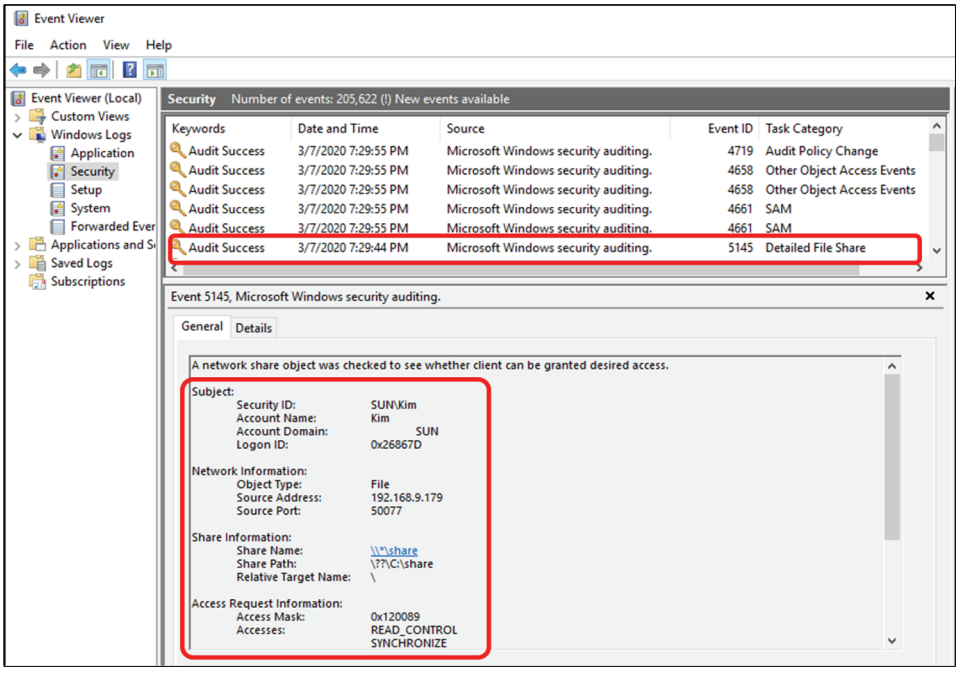

Fig. 2. Windows Event Viewer.

2.2 Windows Event Viewer

Windows provides an Event Viewer, which enables the viewing of event logs stored according to audit policy settings. Event Viewer can list the logs by category, such as Application or Security. Fig. 2 shows an example of selecting the Security category, including the keywords, date and time, event ID, and task

for the logs in this category. To view detailed information in the lower window, we can select an item from the list in the upper window. The details vary depending on the event. As we filter only the file access-related event (event ID: 5145) of interest, it shows the subject, file share information, and access mask.

2.3 Analysis of Windows Events

In the Event Viewer shown in Fig. 2, we can filter the logs by event-ID. However, this feature only supports very simple filtering; it does not support filtering by different attributes or multiple filtering conditions.

When a user performs one action, the system generates many event logs. In many cases, it is difficult to identify the action that caused the events, because different events may occur for the same action. Table 2 outlines the events that occur when we "overwrote a file in a shared folder" twice. The user actions were the same; however, the first action produced 15 logs and the second action produced 26 logs. Table 3 shows the meaning of each bit of the access masks [11,15].

Table 2. Different events for the same behaviors

Sequence number	Case 1		Case 2	
	Relative target name	Access mask	Relative target name	Access mask
1	Destination file name	0x17019F	\	0x120089
2	Destination file name	0x2	Destination file name	0x17019F
3	Destination file name	0x170197	Destination file name	0x2
4	Destination file name	0x2	Destination file name	0x170197
5	Destination file name	0x170196	Destination file name	0x2
6	Destination file name	0x2	Destination file name	0x170196
7	Destination file name	0x80	Destination file name	0x2
8	Destination file name	0x17019F	Destination file name	0x80
9	Destination file name	0x2	\	0x100081
10	Destination file name	0x170197	\	0x100081
11	Destination file name	0x2	\	0x100081
12	Destination file name	0x170196	Destination file name	0x17019F
13	Destination file name	0x2	Destination file name	0x2
14	Destination file name	0x17019F	Destination file name	0x170197
15	Destination file name	0x2	Destination file name	0x2
16			Destination file name	0x170196
17			Destination file name	0x2
18			Destination file name	0x17019F
19			Destination file name	0x2
20			\	0x100081
21			\	0x100081
22			srvsvc	0x12019F
23			\	0x100080
24			\	0x100080
25			Destination file name	0x80
26			\	0x100081

"Destination file name" displayed in the Relative Target Name field is the name of the overwritten file.

Table 3. Meaning of each bit of the access masks

Access mask	Access (neaning)
0x1	ReadDATA (or ListDirectory)
0x2	WriteDATA (or AddFile)
0x4	AppendData (or AddSubdirectory or CreatePipeInstance)
0x8	ReadEA
0x10	WriteEA
0x80	ReadAttributes
0x100	WriteAttributes
0x10000	DELETE
0x20000	READ_CONTROL
0x40000	WRITE_DAC
0x80000	WRITE_OWNER
0x100000	SYNCHRONIZE

3. Elastic Stack

Elastic Stack is a tool recently introduced for analysis visualization in various areas [16,17], including the security domain. Elastic Stack consists of Elasticsearch, Logstash, Kibana, and Beats. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that simultaneously consumes data from multiple sources, transforms it, and then sends it to a "stash," such as Elasticsearch. Kibana enables data visualization with charts and graphs in Elasticsearch. Beats is a family of lightweight, single-purpose data shippers in the Elastic Stack equation [18].

Kibana supports a variety of charts, thereby enabling analysts to choose the appropriate chart for the situation. Analysts can set various options on each chart, such as filter and annotation, to obtain a concise result from a considerable amount of information. In addition, dashboards can be organized by consolidating the various charts required for analysis. Hence, analysts can view the analysis process at a glance.

Recently, many studies on applying Elastic Stack to data visualization have been actively conducted. In the security area, an increasing number of studies have employed Elastic Stack for security threat detection and analysis. Park and Hyun [19] proposed a service that collects scattered web artifacts and provides visualization using Elastic Stack for digital forensics. Kim and Shon [20] used Elastic Stack to detect cyber threats in industrial control systems. Lee and Yang [21] proposed an Elastic Stack-based security log analysis system. We performed an analysis on the windows logs in [22]. In this paper, we present a method to systematically classify logs based on the analysis results, and a method to support analysts by creating an analysis template using Elastic Stack.

4. Proposed Event Trace Model

4.1 Classification of Event Logs

As noted above, an action does not always generate the same events. It may generate different events, depending on the way of execution. We thus designed three event databases, as shown in Table 4, following the same procedure depicted in Fig. 3. First, we recorded the events that occurred by executing

one action more than ten times in the same way. *FullLog* is a list of events that commonly occurred in one action. *ComLog* is a list of common events for each similar action group created by selecting common events from *FullLog*. Finally, *IdLog* is a list of events extracted from *FullLog* and is the list of minimum events that can distinguish each action from other actions.

Table 4. Classified log databases

Log database	Content
FullLog	List of events that occur in common according to the actions and methods performed by the user
ComLog	List of events that occur in common to similar action groups
IdLog	List of events that can identify lists of FullLog

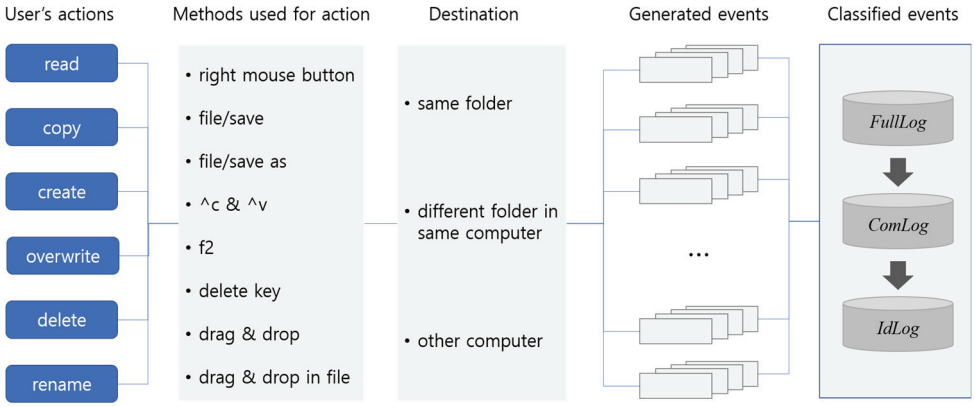

Fig. 3. Classification of logs based on the user's action and method.

Table 5. ComLog and IdLog

ComLog			*IdLog*		
Action group	**Relative target name**	**Access mask**	**Operation**	**Relative target name**	**Access mask**
Read	File name	0x120089	Open Copy to other computer File/Save As to other computer	File name	0x120089
Write	File name	0x12019F or 0x17019F	Save	File name File name	0x12019F 0x120089
			Write (Window function)	File name	0x17019F
			Write with File/Save As	File name	0x120196
			Overwrite	File name	0x170197 or 0x170196
			Overwrite with File/Save As	File name	0x12019F
Delete	Src. file name or Dest. file name	0x110080	Change file name	Src. file name Dest. file name	0x110080 0x110080 or 0x100081
			Delete	File name	0x10080

Table 5 outlines *ComLog* and *IDLog*. We divide the actions that cause the Open, Write, and Delete events into groups and record the common events that occur in each group in *ComLog*. To make *IDLog*,

we list the different actions that trigger the events of each group. Subsequently, we extract the events that distinguish each action. One event consists of a relative target name and an access mask. *FullLog* contains event logs for various actions. Each action consists of multiple logs. Because of space constraints, they are not included herein. *Flist* in Fig. 11 in Section 5.2 shows an example used in our experiment.

4.2 Elastic Stack Template

Illegal file leakage and tampering of shared files are major audit targets. There are various means of copying and modifying files. For example, a copy operation may occur before a modify operation. We experimented using various modification methods. The logs generated by the modification methods thus differed. Fig. 4 shows three representative cases. In the simplest Case 1, we modified the file on the server. In Cases 2 and 3, we modified on the client's computer and overwrote to the server's shared folder rather than modifying in the server's shared folder. Note that the file modification time is not the written time on the server but the modified time on the client [22]. In Case 3, we opened two files on the server and client, respectively, then we copied the content from the server file and pasted into the client file.

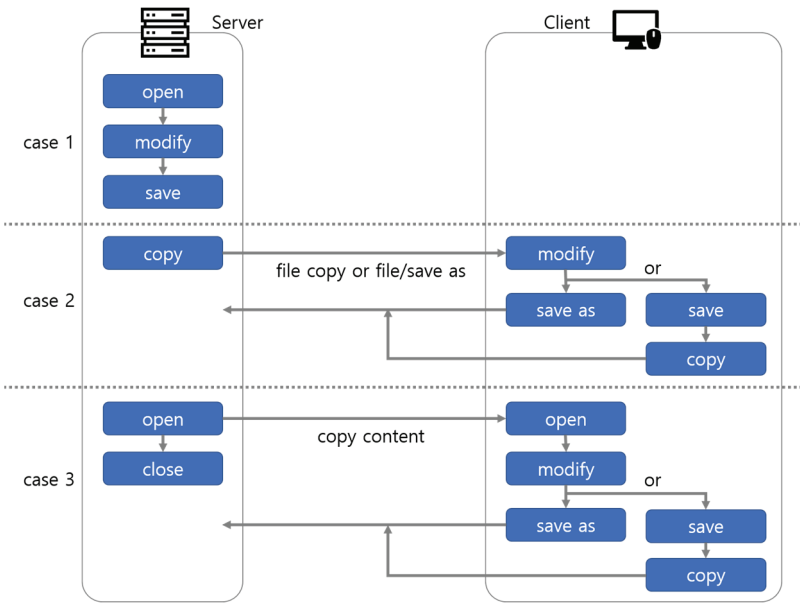

Fig. 4. Three methods to modify a shared file.

Event tracking occurs in several steps. As described in Section 2.2, the Windows Event Viewer provides only simple filtering. We thus used Kibana in Elastic Stack to configure the dashboard template to incrementally filter and analyze the logs, as shown in Fig. 5. The dashboard for the file tampering example consists of two charts (*Wchart* and *Rchart*) and two lists (*Flist* and *Rlist*), as shown in Fig. 5. To clarify the description, we define the symbols as follows:

- t_c: Creation time - when the file was created
- t_m: Modification time - when the file was most recently modified
- t_d: Detection time - when an auditor detects an illegal modification
- t_w: Writing time - when the attacker wrote the file to the server

- t_r: Reading/Copying time - when the attacker read or copied the file
- S: Suspected user - a user suspected of being the attacker

Fig. 5. Dashboard template for file modification.

Wchart is used to identify the time (t_w) of the most recent event in which the file was written/overwritten. In addition, it is possible to identify user *S* who caused the particular event. To more specifically track an attack, *Rchart* is used to identify the time (t_r) when *S* read or copied the file. Table 6 summarizes the options for constructing these charts. Annotations in the options are access masks to track. In *Wchart*, event IDs in *IdLog* are used as options for various ways of generating write/overwrite events. If one aims to ignore the write method used, only 0x12019F and 0x17019F in *ComLog* need be set for the write group. In *Rchart*, the start of the time range is set to t_c; however, it can be set to the last file backup time. We call the list of events selected from *FullList* for comparison *Flist*.

Table 6. Options and objectives of the dashboard charts

Chart	Time range	Filter	Annotation (access mask)	Group by (count)	Purpose (result)
Wchart	t_m to t_d	Event ID (5145), Target file name	0x12019F, 0x170196, 0x17019F, 0x120196	Subject user name	Determine suspect *S*, write time t_w, and write method
Rchart	t_c to t_m	Event ID (5145), Target file name, Subject name	0x120089	Access mask (0x100080)	Check read or copy action

In addition, *Rlist* lists events extracted from *RawLog* to identify the exact file copy method by checking detailed events on the suspect's operation. Unlike writing, it is difficult to distinguish between copying and reading; thus, it is necessary to compare them with *FullLog* on read/copy operations. *Flist* is composed of the *FullLog* to be compared. Table 7 summarizes the options for constructing these lists.

Table 7. Options and objectives of the dashboard lists

List	Selected fields	Time range	Filter	Purpose (result)
Rlist	Event ID, Relative Target Name, Subject User Name, Access Mask	t_c to t_m	Event ID: 5145, Target file name, Subject name Access mask: 0x100080,0x120089,0x100081	Confirm action using *FullLog*
Flist	-	-	-	Compare to *Rlist*

4.3 Event Tracing Process

Event tracing consists of preparation and analysis phases, as shown in Fig. 6. The preparation phase consists of P1, P2, and P3 steps as follows:

- P1: Create *RawLog* by setting the audit policy, as given in Section 2.1, to save the required logs.
- P2: Prepare *FullLog*, *ComLog*, and *IdLog* by analyzing and classifying *RawLog* according to the method given in Section 4.1.
- P3: Prepare dashboard templates for major attacks using *ComLog* and *IdLog*.

For example, if a file is suspected of being manipulated at time t_d, the file modification time is confirmed as t_m then the analysis begins. The analysis phase consists of steps A1, A2, and A3 using the dashboard described in Section 4.2.

- A1: Display *RawLog* in *Wchart* with t_d, t_m, the name of the file suspected of having been manipulated, event ID (5145), and the write-related access masks outlined in Table 5. Find suspect *S*, file modification time t_w, and the write method used in the last event in *Wchart*.
- A2: Add *S* to the *Rchart* option and change the option to the read-related access masks to determine the read or copy operation that was performed before the modification.
- A3: For the actions found in A1 and A2, the estimated suspect and actions are confirmed by comparing the corresponding *FullLog* and *RawLog* and by confirming the occurrences and orders of the detailed events.

If only a simple verification procedure is required, the analyst can verify the suspect's crimes with the results of A1. However, proceeding to A2 and A3, the analyst can obtain detailed evidence of the action. Therefore, the analyst may choose the depth of the evidence trace depending on the situation.

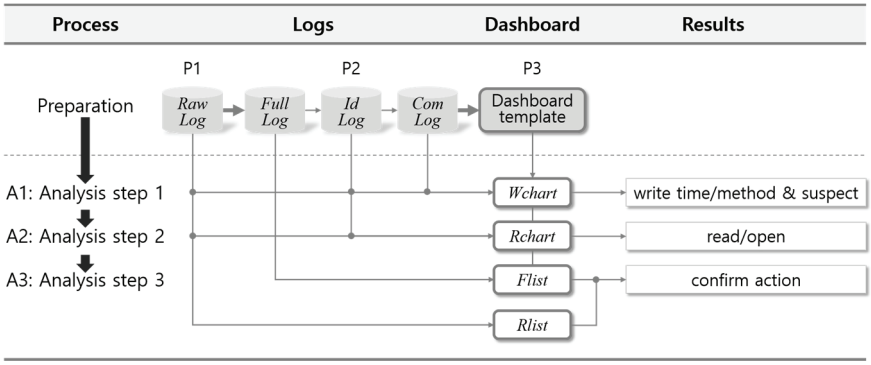

Fig. 6. Tracing process.

5. Experiment and Result

5.1 Experimental Environment

To evaluate the proposed model, we constructed the environment as shown in Fig. 7. Table 8 shows the operating system and software version used. We configured the audit policy in the AD server [23] to log events for "object access," and set up Winlogbeat on the AD server to send the event logs to Logstash

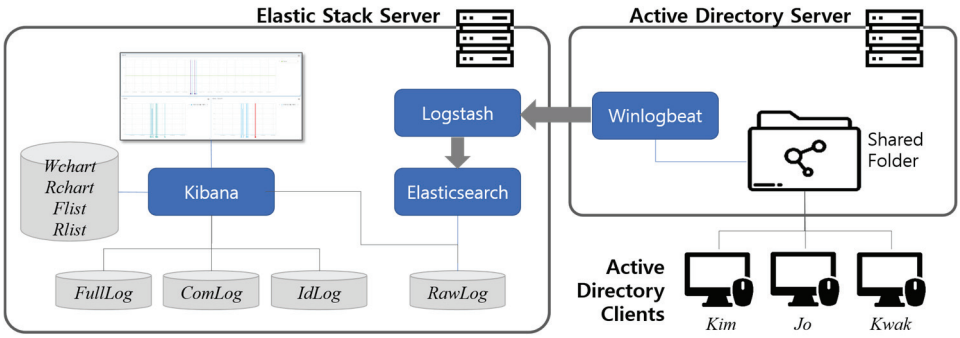

Fig. 7. Experimental environment.

on the Elastic Stack server in real time. On the Elastic Stack server, Logstash receives event logs from Winlogbeat and stores them as *RawLog*. *FullLog*, *IdLog*, and *ComLog* are constructed in table form by analyzing the file access logs in the shared folder. Users *Kim*, *Jo*, and *Kwak* are connected to the AD server as clients.

Table 8. Software versions

Computer	Software	Version
Active Directory server	OS	Windows Server 2016
	Winlogbeat	7.5.2-windo
Active Directory client	OS	Windows 10
Elastic Stack server	OS	CentOS Linux release 7.7.1908
	Logstash	6.8.6-1.noarch
	Elasticsearch	6.8.6-1.noarch
	Kibana	6.8.6-1.x86_64

5.2 Case Analysis

We generated a tampering action for a shared file in the AD server that produced complex logs as follows. The file *secret.doc* in the subfolder *Kim* of the shared folder is a file whose integrity must be guaranteed. *Kwak* copied *secret.doc* to his computer and modified it. He overwrote the modified file onto the server's original file. Fig. 8 shows the timeline of these actions. The actual unit of time stored in the system is milliseconds; however, for convenience, it is here expressed in minutes.

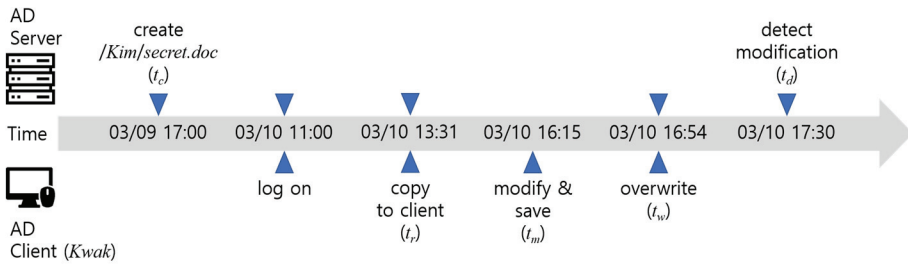

Fig. 8. Timeline of secret.doc modification and detection.

Suppose, March 10, 2020 at 17:30, it is determined that *secret.doc* has been manipulated. The last modification time of the file is March 10, 2020 at 16:15. As shown in Fig. 9, using *Wchart*, we can identify the user who tampered with the file as being *Kwak*. It is also possible to determine that *Kwak* overwrote the file by using "^ C & ^ V" or "drag & drop" on "Mar 10, 2020, 4:54 PM" based on the access mask (0x170196) and time in the annotation. Unfortunately, using the Windows event log it is impossible to determine which overwrite method was used.

Next, as shown in Fig. 10, we used *Rchart* to verify that *Kwak* had read the file several times before overwriting it. In *Rchart*, it was possible to view only the file that had been read for copying several times. To determine the specific copy method, *Flist* and *Rlist* were compared, as shown in Fig. 11. The comparison shows that *Kwak* opened *secret.doc* and then used the "Save as" function in the file menu to save the file on his computer.

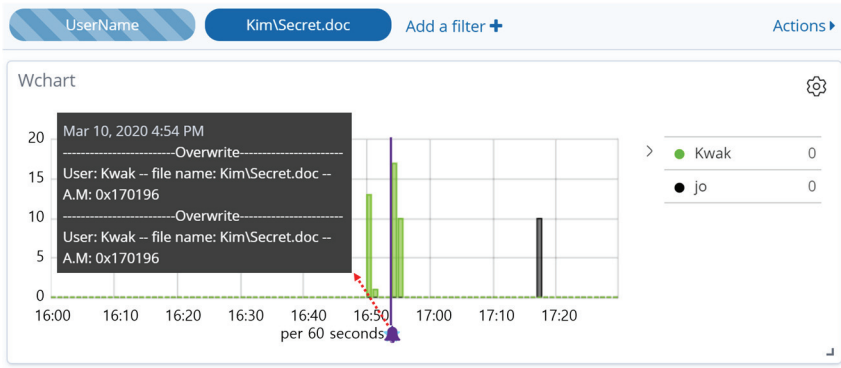

Fig. 9. Identifying the suspect and write time using *Wchart*.

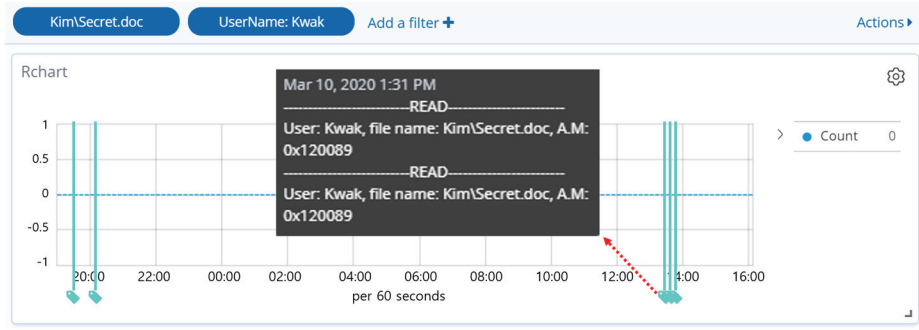

Fig. 10. Identifying the reading action of the suspect using *Rchart*.

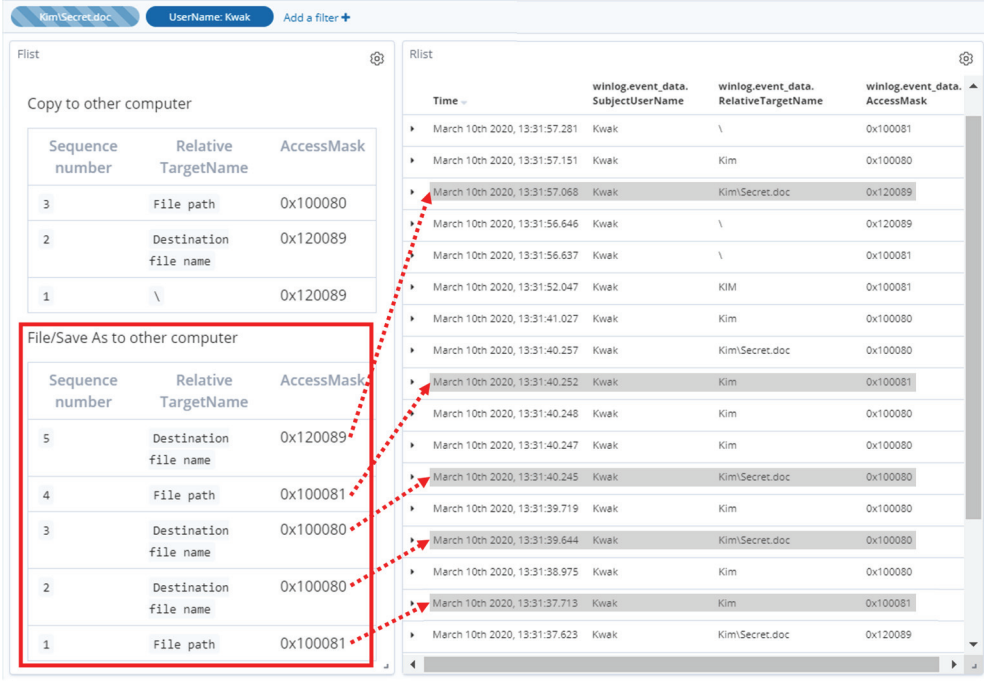

Fig. 11. Detailed actions of the confirmed perpetrator.

It can therefore be concluded that *Kwak* copied *secret.doc* to his computer by using "File/Save as" on March 10, 2020 at 13:31:57.068, modified and saved it in his computer on March 10, 2020 at 16:15, and the overwrote it on the server on March 10, 2020 at 16:54.

6. Conclusion

In this study, we built databases by analyzing complex Windows logs, extracting events that occurred in common for each action on shared files, identifying minor events to distinguish actions, and identifying common events for similar actions. In addition, we designed a dashboard template using Elastic Stack for visual analysis. When an action that requires auditing occurs, the stored event logs can be analyzed by comparing them with the reference databases in the dashboard templates. Evidences of the suspect and action for the events can be selected by adjusting the analysis depth.

In this study, we collected and analyzed logs only from the AD server. However, it is necessary to analyze the logs on the suspect's computer to produce a complete collection of behavior evidences. Therefore, further research is needed to extend the model so that client logs can also be sent to Elastic Stack for analysis. In addition, the databases must be extended to analyze the event logs for other actions and event logs on the client side. One of the limitations of our study is that Windows logs are not sufficient for file tracking. For example, it is not possible to distinguish whether "^C & ^V" or "drag & drop" is used as a file copy method. Elaboration of Windows log is necessary considering digital forensics.

Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. NRF-2017R1D1A3B03032637).

References

[1] A. Nieto and R. Rios, "Cybersecurity profiles based on human-centric IoT devices," *Human-centric Computing and Information Sciences*, vol. 9, article no. 39, 2019. https://doi.org/10.1186/s13673-019-0200-y

[2] P. K. Sharma, J. H. Ryu, K. Y. Park, J. H. Park, and J. H. Park, "Li-Fi based on security cloud framework for future IT environment," *Human-centric Computing and Information Sciences*, vol. 8, article no. 23, 2018. https://doi.org/10.1186/s13673-018-0146-5

[3] OpenText, "EnCase software," 2021 [Online]. Available: https://www.guidancesoftware.com.

[4] Exterro Inc., "Forensic Toolkit (FTK)," 2021 [Online]. Available: https://www.exterro.com/forensic-toolkit.

[5] Magnet Forensics, "AXIOM," 2021 [Online]. Available: https://www.magnetforensics.com.

[6] CaTalk, "Top 7 PCs shared by world/domestic," 2020 [Online]. Available: http://catalk.kr/information/desktop-operating-systems.html.

[7] G2 Inc., "Best Operating System," 2021 [Online]. Available: https://www.g2.com/categories/operating-system.

[8] Z. Zhang, C. Wang, and X. Zhou, "A survey on passive image copy-move forgery detection," *Journal of Information Processing Systems*, vol. 14, no. 1, pp. 6-31, 2018.

[9] C. Wang, H. Zhang, and X. Zhou, "LBP and DWT based fragile watermarking for image authentication," *Journal of Information Processing Systems*, vol. 14, no. 3, pp. 666-679, 2018.
[10] Microsoft, "Active Directory Domain Services overview," 2017 [Online]. Available: https://docs.microsoft.com/ko-kr/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview.
[11] J. Kim, M. Kwak, S. Lee, and T. Cho, "File tracking technique with active directory event log," in *Proceedings of the 2020 World Congress on Information Technology Applications and Services*, Seoul, Korea, 2020.
[12] Microsoft, "Audit policy," 2017 [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/audit-policy.
[13] Microsoft, "Advanced security audit policy settings," 2017 [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/advanced-security-audit-policy-settings.
[14] Microsoft, "Basic security audit policies," 2017 [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/basic-security-audit-policies.
[15] Microsoft, "5145(S, F): a network share object was checked to see whether client can be granted desired access," 2017 [Online]. Available: https://docs.microsoft.com/ko-kr/windows/security/threat-protection/auditing/event-5145.
[16] K. Kim and Y. Cho, "Multi-index approach to search Chinese, Japanese, and Korean text with Elasticsearch 6.6," *Proceedings of International Conference on Future Information & Communication Engineering* , vol. 11, no. 1, pp. 257-260, 2019.
[17] S. Persada, A. Oktavianto, B. Miraja, R. Nadlifatin, P. Belgiawan, and A. P. Redi, "Public perceptions of online learning in developing countries: a study using the ELK Stack for sentiment analysis on twitter," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 15, no. 9, pp. 94-109, 2020.
[18] ElasticSearch, "ELK Stack," 2021 [Online]. Available: https://www.elastic.co/what-is/elk-stack.
[19] J. Park and J. Hyun, "Web artifacts visualization using ElasticSearch and Kibana," in *Proceedings of the IEEK Summer Conference*, 2019, pp. 1350-1353.
[20] Y. Kim and T. Shon, "Cyber-threat detection of ICS using Sysmon and ELK," *Journal of the Korea Institute of Information Security & Cryptology*, vol. 29, no. 2, pp. 331-346, 2019.
[21] B. H. Lee and D. M. Yang, "A security log analysis system using Logstash based on Apache Elasticsearch," *Journal of the Korea Institute of Information and Communication Engineering*, vol. 22, no. 2, pp. 382-389, 2018.
[22] J. Kim, M. Kwak, S. Lee, and T. Cho, "File tracking technique with active directory event log," in *Proceedings of the 14th KIPS International Conference on Ubiquitous Information Technologies and Applications*, Macau, China, 2019.
[23] J. Krause, *Mastering Windows Server 2016*. Birmingham, UK: Packt Publishing, 2016.

Jisun Kim **https://orcid.org/0000-0003-3637-9844**

She received a B.S. degree in information security from Woosuk University in 2021. She is currently a staff of Inc. Igloo Security, Seoul, Korea. Her current research interests include digital forensics and vulnerability analysis.

Eulhan Jo **https://orcid.org/0000-0002-0870-1130**

He received a B.S. degree in forensics and information security from Jeonju Kijeon College in 2021. He is currently a staff of Bidcoaching Research Institute, Daejeon, Korea. His current research interests include digital forensics and cloud computing.

Sungwon Lee **https://orcid.org/0000-0002-2356-7662**

He received a B.S. degree in information communication radio engineering from Kunsan University in 2006. He then graduated from law school. He is currently working for the Izerone Digital Forensics Company and is an adjunct professor in the Department of Information Security, Woosuk University, Jeonbuk, Korea. His current research interests include digital forensics, IoT, security monitoring, and effective security training.

Taenam Cho **https://orcid.org/0000-0002-5191-0130**

She received a B.S. degree in computer science in 1986, an M.S. degree in computer science in 1988, and a Ph.D. degree in computer science in 2004, all from Ewha Womans University. She developed a satellite control system at the Electronics and Telecommunications Research Institute. She is currently a professor in the Department of IT and Electronics Engineering, Woosuk University, Chungbuk, Korea. Her current research interests include Android security, Bluetooth, digital forensics, and block chain.