

Visualization of File Transfer Route for Digital Forensics

Seiji Shibaguchi[†], Taro Inaba[†], Hidekazu Shiozawa[†] and Ken-ichi Okada[§]

Abstract

*In recent years, information technology has been developed and we have come to use electronic files very often. Along with this development, information leakage due to human errors has been increasing. The information leakage can be detected by monitoring transfer of electronic files. In this paper, we propose a visualization system that identifies where confidential files are in an enterprise network. Our system shows not only hosts that have confidential files, but also transfer routes of the files in the network. The proposed system monitors system calls executed on each internal host by hooking Windows APIs, such as *CreateFile* and *send*, to detect the transfer of files in an enterprise network. We can use the system as a digital forensic tool, which identifies criminals who leaked important information, or to prove the innocence of hosts that didn't concern confidential files.*

1 Introduction

In recent years, information technology has been developed and we have come to use electronic files very often. The technology enables us to manage mass information easily. But on the other hand, confidential information faces the risk of information leakage through networks and removable disks. The large part of the leakage is caused by human errors such as setting mistakes, false operation and management mistakes without malice. [1] showed more than 60% of the leakages are due to human errors. To tackle the problem, we propose a monitoring system of confidential electronic files in an enterprise network. The system illustrates the hosts holding the file and a propagation route in real time. By this visualization, we can grasp the transfer route of confidential files more easily than text based logs. In our proposal, we utilize Windows APIs such as *CreateFile* or *send* to get the logs of file transfer. Each host gathers the information of Windows API, and sends the data to a server which monitors a whole network. After the server gets the information from each host, it judges when, where,

and how electronic files are transmitted. When information leakage actually happens, the visualization of file transfer routes will help investigators understand what happens, and decide how they should respond. In other words, this approach aims to develop a support tool for digital forensics. The organization of this paper is as follows. We introduce related studies in Chapter 2, and propose the visualization technique of file transfer route in Chapter 3. Chapter 4 evaluates our proposal and Chapter 5 concludes this paper.

2 Related Work

In this section, we introduce some related works. At first, there is a file monitoring system by hooking Windows APIs[3][4]. This approach monitors system calls such as *open* and *write*. Then, it generates GUI alarms when a user makes or opens confidential files on removable disks. This method aims to monitor electronic files so that it does not leak out to the other hosts. This is because once electronic files have leaked to outside hosts, it is impossible to follow the electronic files. To address the issue, our proposal takes logs such as when, where, and how electronic files are transmitted, instead of caring if electronic files are leaked out. When important electronic files are actually leaked out to other hosts, the visualized file transfer route help us determine to see if a host is involved in file transfer.

[5] proposed another approach. It observes network communication data by hooking Window APIs. It pays attention to “Self-proliferation function of worms through the network” and detects unknown worms. It monitors the correlation of “Data received from outside” and “Data transmitted to outside” by hooking Windows APIs such as *send* and *recv*. When the system finds any programs that show worm-like behavior, the system calls *closesocket* to prevent them from communicating with other hosts. While our approach is somewhat similar to the proposal in terms that the both monitor *send* system calls, our purpose is to track the file transmissions over an enterprise network, and therefore is different from [5].

Third, there is a quite different approach[6]. This proposed system monitors “human behavior” and tries to detect suspicious behavior. It detects suspicious behavior from the living body information such

[†] Graduate School of Science and Technology, Keio University

[‡] Faculty of engineering, Tamagawa University

[§] Faculty of Science and Technology, Keio University and JST

as heartbeat counts and eyes movement.

At last, [7] proposed file access control software. It proposes a file access control software agent that provides users to use P2P file sharing software in safety, and monitors all file accesses and blocks them from unauthorized applications. Then, this system can avoid inappropriate sharing by blocking accesses to confidential files. The agent monitors GUI operations and analyzes process behaviors to detect critical accesses.

3 Visualization of File Transfer Route

Our system monitors electronic files by hooking WindowsAPIs and watches the movement of important files to other hosts to make logs. Then, each host sends logs to a server. The server visualizes the transfer route.

3.1 System Overview

We show the outline chart of our proposal in figure 1.

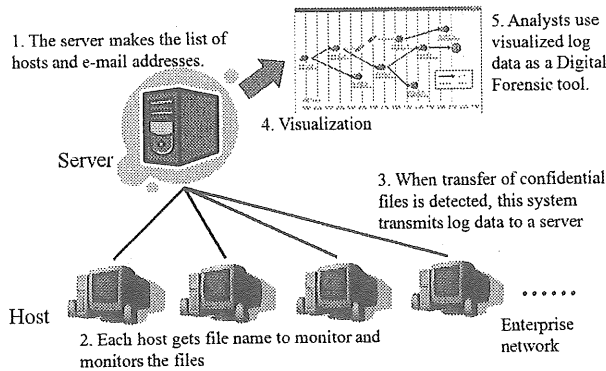


Figure 1: System overview

The steps of the visualization procedure are as follows.

1. The Server makes the list of hosts and e-mail addresses

We assume that each host corresponds to a certain e-mail address. The server makes and holds a list of the relation.

2. Hosts Monitor files

The monitoring of each host is enabled by hooking WindowsAPIs, which will be mentioned in section 3.2. There are various methods for the transfer of electronic files, but our system supports e-mails and removable disks in this paper.

We mention the monitoring method in section 3.3.

3. Hosts send logs to the server

When transfer of electronic files is detected in a host, the host sends logs to the server at any time. Logs are text based data which include time of transfer and IP addresses of related hosts.

The log information is mainly two kinds: "Information of files" and "Transfer methods".

- Information of files

For information of files, a host sends file names and hash values to the server. The hash value is used to verify important files. The hash algorithm is SHA-1.

- Transfer methods

There are 2 patterns of file transfer methods: "e-mails" and "removable disks". In the case of e-mails, the server obtains the source/destination e-mail addresses from the monitoring program in the sender host, and resolves the IP addresses of sender and receiver hosts with e-mail-host relation list.

4. The server visualizes logs

The server manages two information mentioned above and judges when, where, and how electronic files are transmitted. Then, our system visualizes a file transfer route so that people can understand the route easily. We explain the concrete visualization technique in section 3.4.

5. Analysts use visualized log data in Digital Forensics

Analysts use this visualization system as a support tool of digital forensics.

3.2 Monitoring of hosts

Our proposed technique uses hooking WindowsAPIs of all processes. Rewriting of the import address table realize hooking WindowsAPIs. In our technique, the system calls rapper functions before WindowsAPI functions are called. Although there are various methods to transmit electronic files, in this paper we handle e-mails and removable disks cases. Now this two methods are popular ways to transfer electronic files, so we can monitor large part of file transfer[8][9][10].

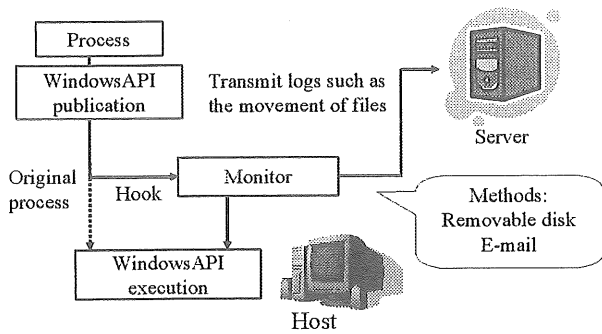


Figure 2: A design of the API hook

Figure 2 is a design of the API hook in each host. This system performs rapper functions before specific WindowsAPI's functions are called, and analyzes an argument and watches if the host accesses to electronic files. If it accesses to electronic files, this system sends the log data to the server. Finally this system calls a genuine WindowsAPI functions after a series of operation is over. In this phase, this system hooks two WindowsAPI functions: *CreateFile* and *send*. *CreateFile* is an opening function to electronic files. When Windows applications access files, they must call this function. By hooking this function, we can monitor access to files. We show a rapper function pseudocode of *CreateFile*, *Hook_CreateFile*, below.

```

Hook_CreateFile(PCTSTR fileName,.....){
    if(fileName == Monitored_electronic_file){
        //Take the hash of the file.
        //Transmit a hash value to the server.
    }
    CreateFile(PCTSTR fileName,.....);
}

```

This system monitors which file is opened by watching a *fileName* argument in *Hook_CreateFile*. When an important file is opened, this system acquires file data from the absolute pass of a file and makes the hash value of the file to transmit log data to the server. Finally *Hook_CreateFile* calls original function *CreateFile*.

In addition, this system hooks *send* so that it watches transfer of e-mail messages. Then, it finds out e-mail addresses of relative hosts. We show a pseudocode of the rapper function *Hook_send*.

```

Hook_send(..,void *buf,..){
    if(file_open_process is email software){

```

```

        //Analyze buf data.
        //Specify destination address.
    }
    send(..,void *buf,..);//Original process.
}

```

At first, this system judges whether the original process that calls *send* is e-mail software or not. If this is from e-mail software, *Hook_send* sees a *buf* argument, which is the data of mail transfer. It identifies the source and the destination of the message, and transmits them to the server. Finally *Hook_send* calls the original function of *send* in rapper function *Hook_send*.

3.3 Monitoring of the individual technique

This technique monitors transfer of electronic files through e-mails and removable disks.

3.3.1 e-mail

Here, we consider the case where electronic files are transferred to other hosts by attachments of e-mails.

We show the judgment method below.

1. For simple analysis, we assume one host has one e-mail address. And this system makes the correspondence list of hosts and e-mail addresses on the server.
2. This system monitors all *CreateFiles* which the e-mail software performs and checks if the software accesses to the monitoring electronic files. If a process of e-mail software accesses to monitored files, we regard the process will send mails with the files to other hosts.
3. This system transmits log data if an electronic file is transmitted to other hosts.

If the destination address does not exist in the address - host relation list, which is preserved in the server, this system judges that the message and the files are transmitted to an outside host.

3.3.2 Removable disk

In this section, We mention the case where removable disks bring electronic files to other hosts. There are 2 patterns for file transfer.

- Host → Removable disks

- Removable disks → Host

We show these detecting methods below.

1. Monitoring CreateFile

(1) Host → Removable disks

When file transfer is detected in removable disks newly, the host must call *CreateFile*. Then, our system watches *CreateFile* in removable disks, and if its argument "fileName" is equal to monitoring file name, sends log data to the server.

(2) Removable disks → Host

To copy files copied from removable disks, the files must be read by some processes. Thus, before reading files, the processes need to call the *CreateFile*. So, the system can detect such activities by monitoring *CreateFile*, like the case (1).

2. The server receives log data and checks "Has this host already had a monitor file?"

When a server receives log data, it checks whether the sender host actually owns the confidential files. If so, the server estimates that files are moved to removable discs, otherwise files are copied from removable discs.

3.4 Visualization

Our system handles monitoring information with an individual hosts in the server side in real time and visualizes it. Figure 3 shows icons used in our visualization. Each icon shows where confidential files are moved. Figure 4 shows an example of visualization. In this case, host A transmitted a confidential file to host B at 0:40. We will show more concrete example in section 4, with an evaluation.

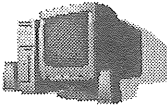


	Host
	Removable disk
	Outside host

Figure 3: A list of icons

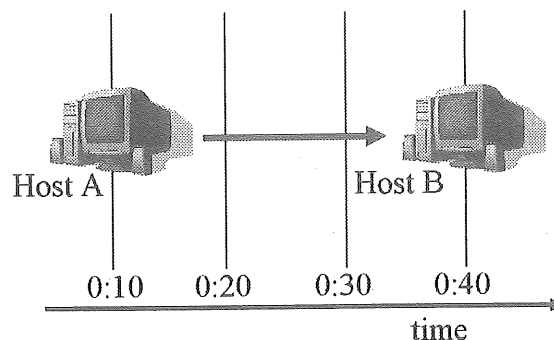


Figure 4: An example of file transfer

4 Evaluation

We conducted two following evaluations.

1. Evaluation of system loads of the hosts.

We measured three following loads here.

- CPU Load
- Memory Load
- Network Load

We examined the CPU utilization rate of the monitoring application. In addition, we examined the network load and the memory load as well.

2. We monitored specific electronic files in a real network, and visualized a propagation route.

In this evaluation, we monitored specific files in the real network and evaluated whether this system was able to visualize the transfer of the files. Here we used 4 hosts for this evaluation.

4.1 Implementation

We implemented the visualization of file transfer route system. The programming languages are as follows.

- C++
 - The monitoring application program of each host.
 - Server program.
- Java
 - Visualization program.

Table 1: Specifications of machines

	CPU	Memory	OS	CPU name
Server	2.8GHz	512Mbyte	Windows XP SP2	Pentium4
Host	1.6GHz	256Mbyte	Windows XP SP2	Pentium4

In addition, specifications of machines are shown in table 1, and we used Becky as an e-mail software [11].

4.2 The system load

4.2.1 (a) CPU Load

We investigated the CPU utilization of the monitoring application. We used a task manager for the investigation. When there was no transfer of files, CPU utilization are approximately 0%. Even when other application started with our application, the CPU utilization was approximately 0%.

Second, we consider the case where monitoring electronic files are transferred. Even when it is necessary for the host to send the log data to the server, CPU utilization were between only 0% - 20% in 2 - 3 seconds. This resembles the behavior of a simple application such as notepads. In addition, confidential files are not transferred very often. This result shows that there is not approximately CPU load of the monitoring application of hosts.

4.2.2 (b) Memory Load

From our investigation, the amount of memory which the proposal application uses is 6 - 7 Mbytes. It is not high memory load because general Windows applications use 10M - 20Mbytes[12]. Therefore, our system does not consume so large memory that performances of other applications will not be affected.

4.2.3 (c) Network Load

In our system, each host transmits about 100 byte data to the server when confidential files are accessed. The size is so small that our system does not burden the network very much. Therefore, our system does not burden the network very much.

Thus, we can say loads of this system are small.

4.3 Visualization

We visualized file transfer route in a real network. And we show it in figure 5. Figure 5 indicates when,

where, and how electronic files are transmitted,

This application updates information of transferred electronic files from the logs. Since all hosts provide the information to the server at any time, this indication changes in real time. Our proposal can monitor the movement of a lot of files, but this system decides to display one type of the file transfer route. This is because our system aims to visualize the flow of file transfer so that analysts are able to see the relation at a glance, and if there are many hosts and transfer routes in the windows, it is hard to see. In addition, the horizontal axis indicates the flow of time, and this application changes time interval into the most suitable value. This visualization is easy to see file transfer route. And it identifies the criminal that leaked important information, or which proves the innocence of hosts that didn't concern confidential files. Then, This visualization can be usable in digital forensics.

5 Summary

In recent years, information technology has been developed and we have come to use electronic files very often. Along with this development, information leakage due to human errors has been increasing.

Against this, We propose a visualization system that identifies where confidential files are in an enterprise network. Our system shows not only hosts that have confidential files, but also transfer routes of the files in the network. The proposed system monitors system calls executed on each internal host by hooking Windows APIs, such as *CreateFile* and *send* to detect the transfer of files to the network and removable disks. This time, our proposal monitors transfer of electronic files through e-mails and removable disks. And evaluation results shows that the system loads are small and our proposed system realizes visualization of file transfer route in a real network. Then, we can use the system as a digital forensic tool, which identifies the criminal that leaked important information, or which proves the innocence of hosts that didn't concern confidential files.

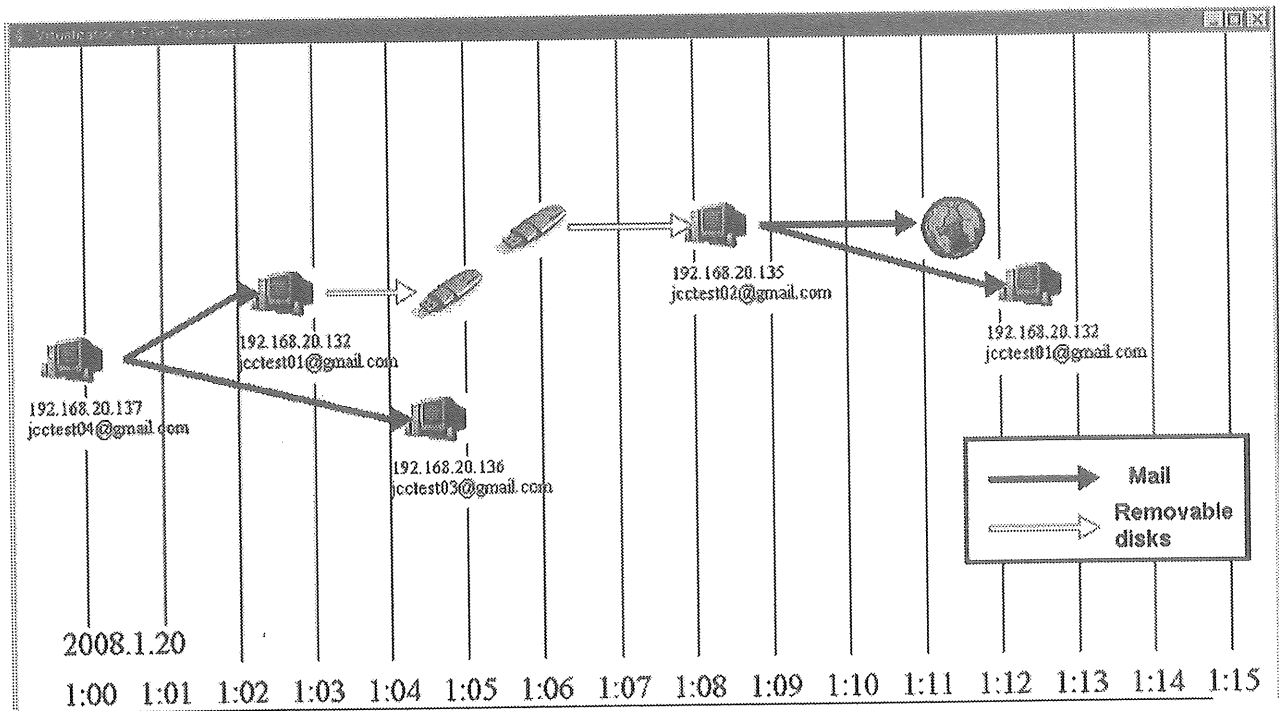


Figure 5: Visualization

Acknowledgements

This research was supported in part by JST, CREST and Secom Science and Technology Foundation.

References

- [1] Information-technology promotion agency,japan Incident survey 2005. http://www.jnsa.org/result/2005/20060803_pol01/2005incidentsurvey_060731.pdf
- [2] Wiki http://ja.wikipedia.org/wiki/Windows_API
- [3] Kei Ohashi.: Write Control Method by Using Diffusion Tracing Function of Classified Information: Multimedia, Distributed, Cooperative, and Mobile Symposium, pp. 690-697 (2007.7).
- [4] Satoshi Hakomori,Hideo Taniguchi.: Classified Information Diffusion Tracing Triggered by Open System Call: IPSJ Journal Vol2005,No79,pp.1-8
- [5] Takaaki Matsumoto,Masakatsu Nishigaki.: An Unknown-worm Spread Prevention Based on the Correlation between the Transmission and Reception Data: IPSJ Journal Vol.47 No.6 pp1941-1954(2006.6)
- [6] Hirokazu Maruoka,Masakatsu Nishigaki.: A countermeasure against insider with detection of suspicious behavior: IEICE Technical Committee on Biometric System Security, pp.27-33 (2007.3)
- [7] Koji Kida Hiroyuki Tarumi: A Proposal of File Access Control Software Agent Toward Using P2P File Sharing System in Safet: IPSJ Journal Vol.48 No.1 pp200-212(2007.1)
- [8] IDNetworks http://www2.idnetworks.co.jp/column/security_risk/
- [9] TOSHIBA solution <http://www.toshiba-sol.co.jp/nsd/sec/p012b.htm>
- [10] Leak Proof <http://www.hitachisystem.co.jp/leakproof/sp/truth/index.html>
- [11] Becky! <http://www.rimarts.co.jp/becky-j.htm>
- [12] @IT <http://www.atmarkit.co.jp/fwin2k/win2ktips/166memoryusage/memoryusage.html>