

Study on Systematic Ransomware Detection Techniques

Sun-Jin Lee*, Hye-Yeon Shim*, Yu-Rim Lee*, Tae-Rim Park*, So-Hyun Park†, Il-Gu Lee*†

* Department of Convergence Security Engineering, Sungshin University, South Korea

†Department of Future Convergence Technology Engineering, Sungshin University, South Korea

{20180935, 20180922, 20180940, 20180917, 220206035, iglee}@sungshin.ac.kr

Abstract—Cyberattacks have been progressed in the fields of Internet of Things, and artificial intelligence technologies using the advanced persistent threat (APT) method recently. The damage caused by ransomware is rapidly spreading among APT attacks, and the range of the damages of individuals, corporations, public institutions, and even governments are increasing. The seriousness of the problem has increased because ransomware has been evolving into an intelligent ransomware attack that spreads over the network to infect multiple users simultaneously. This study used open source endpoint detection and response tools to build and test a framework environment that enables systematic ransomware detection at the network and system level. Experimental results demonstrate that the use of EDR tools can quickly extract ransomware attack features and respond to attacks.

Keywords—Ransomware, ransomware detection, endpoint detection and response (EDR), Google rapid response, osquery, Open Source hids SEcurity (OSSEC), open-source EDR

I. INTRODUCTION

Stronger network ultra-high-speed, ultra-low-delay, hyper-connected, and super-intelligent features result in a greater extent and scale of damage from cyberattacks. Recently, cyberattack incidents have increased in which advanced persistent threat (APT)-type ransomware is propagated using network vulnerabilities. Ransomware is malicious software that demands a cost when the computer system is encrypted to then decrypt it. Types of ransomware include device encryption, data encryption, data destruction, data hijacking, and sending threatening messages [1].

Ransomware has become intelligent and advanced, producing various mutations, and it has developed in a direction that is difficult to detect, causing great damage. In 2017, the ransomware Wannacry infected more than 230,000 computers in 150 countries through data encryption. Recently, ransomware has appeared that uses obfuscation techniques, and ransomware-as-a-service (RAAS) is a type of ransomware provided as a service for immediate use for attacks [2, 3].

Starting in 2016, the damage due to ransomware mutations, which have increased exponentially, is also increasing rapidly

every year [4, 5]. In 2018, ransomware was considered the most damaging attack of malware, and the damage caused by ransomware in 2019 reached 11.5 billion dollars. In 2021, it is expected to cause about 20 billion dollars in damage, up 8.5 billion dollars from 2019 [5, 6]. To prevent and respond to the damage that has exponentially increased, a response system must be developed that is fast and efficient.

The importance of digital forensics has also increased to detect and respond to intelligent and advanced cyberattacks. In particular, research is being actively conducted on endpoint detection and response (EDR), which applies a behavior-based detection and response method, and remote digital forensics [7, 8]. The remote forensic tools and techniques allow investigators to remotely access EDR clients for a quick response.

This study identifies in real time whether clients are infected with open-source cryptographic ransomware, RAASNet, through Google Rapid Response (GRR), Facebook's osquery, and Open Source hids SEcurity (OSSEC) among systematic ransomware detection techniques [9-12].

The contributions of this study are as follows:

- 1) A new framework is proposed to respond to intelligent ransomware using open source EDR at the network and system level, not at a single terminal or file.
- 2) The feasibility of the proposed new ransomware response method is proven in the actual system.

The rest of the paper is organized as follows. Section II provides relevant research. Section III presents the evaluation and analysis of open source-based ransomware detection techniques. Section IV provides a comprehensive summary of the EDR-specific detection methods evaluated in Section III and concludes the work in Section V.

II. RELATED WORK

As the effect of ransomware increases, research continues to detect and defend against ransomware. Previous research on ransomware detection has suggested a network-based detection approach that uses the characteristics of ransomware communication by analyzing the sequence of HTTP messages and the amount of content of each message [15]. This study designed a detection system using a software-defined network

This work was partly supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2020R1F1A1061107) and the Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (No. 2020-0089).

solution to quickly respond to discovered threats based on observations of network communications from ransomware suites, such as Locky and CryptoWall. However, a limitation is that size of the HTTP response is not considered for a command and control server, which is primarily used for malicious attacks. Several other studies have aimed to defend ransomware by analyzing its characteristics. In addition, research on ransomware detection using open-source EDR is currently underway.

A prior study using GRR, which is a typical EDR tool, aimed to detect attacks with GRR using vulnerabilities in the file transfer protocol servers and Adobe Reader [13]. The study proved that it is possible to detect abnormal flows of network and memory using GRR hunts when an attack is attempted. A limitation of this study is that the EDR tool for detecting attacks was limited to GRR, so it was impossible to compare the differences between other analysis tools.

Another prior study using osquery, another open-source tool, involved detecting Trojan horses, such as Applejeus: Lazarus and jRAT/Adwind using osquery, fleet, and elastic stacks [14]. The study found that Trojan horses could be effectively detected using osquery. A limitation of the study is that only Trojan horses were used for detection; thus, the possibility of other types of malware cannot be determined.

This study attempts to demonstrate the potential for ransomware detection in various EDR tools using GRR, osquery, and OSSEC to detect the abnormal behavior of attackers. In this study, tests were conducted regarding whether each EDR tool can detect file changes caused by ransomware. Detection results were displayed in the form of notifications or logs when all files in a specific directory were encrypted, indicating that all three EDR tools in the Linux environment are capable of detecting ransomware.

III. OPEN-SOURCE ENDPOINT DETECTION AND RESPONSE-BASED RANSOMWARE DETECTION TECHNIQUES

A. Systematic Ransomware Detection Framework

The goal is to use the EDR system to detect when a specific memory space of a victim is infected by RAASNet, an open-source ransomware. Ransomware payload, which is generated by RAASNet, uses the “override and rename” type to overwrite existing files with encrypted files and change the extensions to DEMON. The methods for detecting ransomware in GRR, osquery, and OSSEC are as follows:

1) Google Rapid Response

Using File Finder Flow, which allows the GRR server to gather information concerning the files in a specific memory space among the flows, the query of the infected memory space of the victim is performed twice, before and after infection. This compares the file status before and after the victim’s environment was infected with the ransomware.

2) Osquery

In osquery, a file query proceeds to query all files present in the victim’s infected memory space. As with GRR, two

queries, before and after the ransomware infection, check for status changes, such as file extensions.

3) Open Source hids SEcurity (OSSEC)

Using an integrity check, which is one of the functions of OSSEC, OSSEC detects changes in ransomware infection files. Moreover, the configuration sets the memory space where integrity checks are performed to detect encryption and extension changes in files present in the memory space.

B. Experimental Environment

The experimental environment for ransomware detection was conducted by separating the server and client. The servers were deployed in different EDR (GRR, osquery, and OSSEC) environments, and /testt directory was created on the client. The /testt directory consists of files to be infected. In addition, the RAASNet ransomware was used to infect the files in the /testt directory. Table 1 summarizes the components and versions employed in the construction of the experimental environment.

TABLE 1. COMPONENTS AND VERSIONS USED TO BUILD THE EXPERIMENTAL ENVIRONMENT

Component	Version
CPU	Intel® Core™ i7 Processor 8565U (1.80 GHz up to 4.60 GHz 8 MB L3 Cache)
Server, Client (agent) OS	Ubuntu 18.04.4 LTS
RAASNet	https://github.com/leonv024/RAASNet

In the case of the CPU, Intel’s 8th generation i7 was used, and the operating system used in each server and client was Ubuntu 18.04.4 LTS version.

C. Experimental Process

1) Google Rapid Response

The GRR server and client versions performed in this experiment are 3.4.0-1. Fig. 1 illustrates the GRR experimental environment.



Fig. 1. Configuration of the Google Rapid Response (GRR) experiment environment.

Among the flow functions of the GRR server, File Finder was used to retrieve information about files in a specific memory space. The GRR server sends a query to detect the infected memory space (specific space) of the client (victim). For comparison, queries were conducted twice, before and after infection by RAASNet, from the GRR server to the GRR client. Then, the path should be designated as the location of the folder or file to be monitored.

2) Osquery

The osquery version is 4.3.0. Fig. 2 illustrates the experimental configuration of osquery.

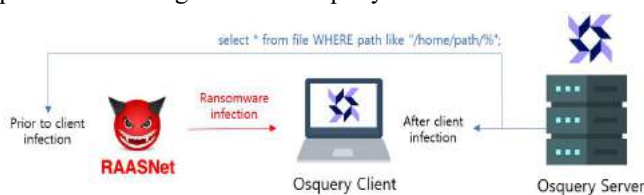


Fig. 2. Configuration of the osquery experiment environment.

Using osqueryi, queries related to ‘file’ were sent to monitor all files in the infected memory space of the client. Similar to GRR, a comparison was conducted before and after the ransomware infection, using the “select * from file WHERE path like “/home/path/%”;” query statement. The infection was determined based on the query result. The path specified in the query statement should be the location of the folders and files to be checked.

3) Open Source hids SECURITY (OSSEC)

The server and client ossec-hids-3.1.0 versions were used in this experiment. Fig. 3 below displays the experimental setup of OSSEC.

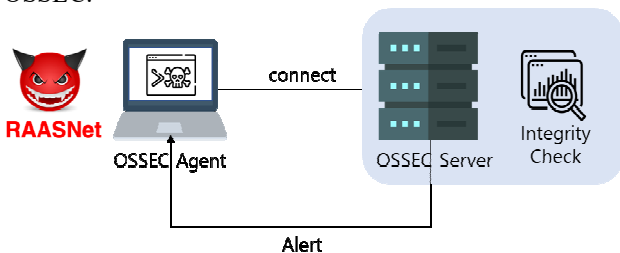


Fig. 3. Configuration of the Open Source hids SECURITY (OSSEC) experiment environment.

The integrity check function of OSSEC was used for ransomware experiments. In the OSSEC configuration, the memory space for the integrity check was set to the /testt directory, and monitoring was scheduled for every 30 s. Then, the experiment was conducted. The RAASNet infection was determined by detecting that the characteristics of the files in the specified memory space were modified.

D. Experiment Results

To confirm ransomware detection, each detection was performed before and after RAASNet execution. Targets to be infected by the ransomware were 10 TXT files, 10 JPEG files, 10 MP4 files, and 10 PNG files. The results of the ransomware detection experiments based on the experimental environment are discussed in the following sections.

1) Google Rapid Response

Figs. 4 and 5 depict the detailed results detected by the GRR File Finder Flow before and after the execution of RAASNet. As a result, the filename, st size, st blocks, atime, mtime, and ctime were changed. The abbreviation atime refers

to the file access time, which is a value that is recorded when a file is opened or read by command or script. In addition, ctime refers to the file change time and is a value recorded at the time when attribute values, such as file data or authority, owner, and file size, are changed. Moreover, mtime refers to the file modify time, which is the data modification time of the file and is recorded when the file content changes. If the file size or mtime changes by modifying the data, ctime also changes. It was found that atime, mtime, and ctime were changed, as the file properties were altered due to the execution of RAASNet.

Field	Value
Atime	2020-11-05 23:50:52 UTC
St mode	-rwxr-xr-x
St ino	329584
St dev	2049
St nlink	1
St uid	1000
St gid	1000
St size	1000
St atime	2020-11-05 23:50:52 UTC
St mtime	2020-11-05 23:50:52 UTC
St ctime	2020-11-05 08:13:35 UTC
St blocks	16
St blksize	4096
St rdev	0
St flags_rwx	0
St flags_nlink	0
Pathname	/home/sjlee/testt.png
Path type	OS
Path options	CASE_LITERAL

Fig. 4. Results of Google Rapid Response File Finder Flow before running RAASNet.

Field	Value
Atime	2020-11-05 08:41:21 UTC
St mode	-rwxr-xr-x
St ino	329584
St dev	2049
St nlink	1
St uid	1000
St gid	1000
St size	1000
St atime	2020-11-05 08:41:21 UTC
St mtime	2020-11-05 08:41:21 UTC
St ctime	2020-11-05 08:41:21 UTC
St blocks	16
St blksize	4096
St rdev	0
St flags_rwx	0
St flags_nlink	0
Pathname	/home/sjlee/testt/3 test.DEMON
Path type	OS
Path options	CASE_LITERAL

Fig. 5. Results of Google Rapid Response File Finder Flow after running RAASNet.

2) Osquery

Figs. 6 and 7 list the results of osqueryi, and the file was infected before and after RAASNet was run. After executing the RAASNet ransomware, the query statement “select * from file WHERE path like “/home/path/%”;” was performed to check the status list of the files. As a result, it was confirmed that the file name, atime, mtime, and ctime were changed. That is, the properties of the files in the /testt directory changed when RAASNet was executed.

```
osquery> select * from file WHERE path like "/home/sjlee/testt/%";
```

directory	filename	atime	mtime	ctime	btline
/home/sjlee/testt	1.txt	0	1604619644	1604737087	0
/home/sjlee/testt	10.txt	0	1604619836	1604737087	0
/home/sjlee/testt	2.txt	0	1604619672	1604737087	0
/home/sjlee/testt	3.txt	0	1604619690	1604737087	0
/home/sjlee/testt	4.txt	0	1604619708	1604737087	0
/home/sjlee/testt	5.txt	0	1604619730	1604737087	0
/home/sjlee/testt	6.txt	0	1604619750	1604737087	0
/home/sjlee/testt	7.txt	0	1604619770	1604737087	0
/home/sjlee/testt	8.txt	0	1604619794	1604737087	0
/home/sjlee/testt	9.txt	0	1604619820	1604737087	0

Fig. 6. Results of osqueryi before RAASNet execution.

```
osquery> select * from file WHERE path like "/home/sjlee/testt/%";
```

directory	filename	atime	mtime	ctime	btme
/home/sjlee/testt	1.txt.DEMON	1604739121	1604739121	1604739121	0
/home/sjlee/testt	10.txt.DEMON	1604739120	1604739120	1604739120	0
/home/sjlee/testt	2.txt.DEMON	1604739121	1604739121	1604739121	0
/home/sjlee/testt	3.txt.DEMON	1604739121	1604739121	1604739121	0
/home/sjlee/testt	4.txt.DEMON	1604739121	1604739121	1604739121	0
/home/sjlee/testt	5.txt.DEMON	1604739120	1604739120	1604739120	0
/home/sjlee/testt	6.txt.DEMON	1604739121	1604739121	1604739121	0
/home/sjlee/testt	7.txt.DEMON	1604739120	1604739120	1604739120	0
/home/sjlee/testt	8.txt.DEMON	1604739123	1604739123	1604739123	0
/home/sjlee/testt	9.txt.DEMON	1604739120	1604739120	1604739120	0
/home/sjlee/testt	9.txt.DEMON	1604739123	1604739123	1604739123	0

Fig. 7. Results of osqueryi after RAASNet execution.

3) Open Source hids SECURITY (OSSEC)

Moreover, OSSEC was configured to continuously monitor every 30 s and to check for changes after executing RAASNet. Figs. 8 and 9 list the file changes before and after RAASNet ran. Fig. 9 is the result of comparing each hash value before and after RAASNet execution using the integrity verification function of OSSEC. After running RAASNet on a file in the /ttest directory, the changes in the hash values were detected.

/home/sjlee/testt/png-4.png	md5 396415b1f051882e7b6152b39ed8b616 sha1 e25081497613633b0db050e709f8a12a8c75d
/home/sjlee/testt/mp4-9.mp4	md5 91ae092829bce25747d14243047e033 sha1 25c1c0055ce81247af9935389e1acea490db68f
/home/sjlee/testt/peg8.jpg	md5 d38a8b14e4d598d3010d010d220a3af sha1 2a23e0ba64d7b5c149c1f1788bba25860cbab3
/home/sjlee/testt/peg4.jpg	md5 ala744843ab45e17f669389a2f4d0 sha1 9e1fa72161250234ae60b523aabc01f3ee12d9
/home/sjlee/testt/7.txt	md5 3d1555580cd423c0e800000c24653 sha1 0eb60e940755ec53bf7ae87b2c369672ce4d3
/home/sjlee/testt/png-2.png	md5 d95a9249ae199e26360dc4c915ce05 sha1 555da7864848a1b0e670c4c084406680b678
/home/sjlee/testt/9.txt	md5 d73d7bc13e82416e050e639090521a sha1 9bb0ca09c54c5f8970eeb351d0cca8472672956
/home/sjlee/testt/png-3.png	md5 50894ad5b310ace639439c3a4678f0 sha1 3c91d98c3671442b4b1cc10d5c20bb1a6a8334
/home/sjlee/testt/peg1.jpg	md5 00bb804c2c01c0526cde83bc9c952b sha1 62070962bac7ccc63e069d56c419968d3c39b
/home/sjlee/testt/10.txt	md5 d7a1f666ac139594f2889de43ae07 sha1 c96017b04c4b4d4a97ca7d4b0f4a49f7b

Fig. 8. Hash value of files in /testt directory before RAASNet execution was checked using Open Source hids SECURITY (OSSEC).

/home/sjlee/testt/mp4-3.mp4	md5 b5f2a937772814a80b6d0c0450ead5a6 sha1 2555d59d1348de21890d8e6a44db9906a9597
/home/sjlee/testt/mp4-6.mp4	md5 b5f2a937772814a80b6d0c0450ead5a6 sha1 2555d59d1348de21890d8e6a44db9906a9597
/home/sjlee/testt/peg2.jpg	md5 31920671825c183b65510ae255b64702 sha1 a3b7568b82b3968d30f9c4881c13d2de16edec
/home/sjlee/testt/png-6.png	md5 31920671825c183b65510ae255b64702 sha1 a3b7568b82b3968d30f9c4881c13d2de16edec
/home/sjlee/testt/mp4-1.mp4	md5 4ed9c74b4879a9e44f3a614730ef358 sha1 5d16cbce777240754b99a1583ace0e9089f5
	md5 4ed9c74b4879a9e44f3a614730ef358 sha1 5d16cbce777240754b99a1583ace0e9089f5
	md5 a3da3b78541b76004bb104e0671a350 sha1 3b3c1a160b616ea81442553ch2a1ea8b5553d2e0
	md5 a3da3b78541b76004bb104e0671a350 sha1 3b3c1a160b616ea81442553ch2a1ea8b5553d2e0
	md5 75546abc5e1932b1d1c1f9dc89e2466 sha1 5f241568408d853ed92615af5f59716d3afad
	md5 75546abc5e1932b1d1c1f9dc89e2466 sha1 5f241568408d853ed92615af5f59716d3afad

Fig. 9. Hash value of files in /testt directory after RAASNet execution was checked using Open Source hids SECURITY (OSSEC).

IV. EXPERIMENT SUMMARY ANALYSIS

When a file was infected by RAASNet ransomware, for GRR with File Finder Flow, it was confirmed that the file name, atime, mtime, ctime, st size, and st blocks were changed. In the case of osquery, a query statement (select * from file

WHERE path like "/home/path/%";) was used to detect the ransomware. Thus, the changes in the file name, atime, mtime, and ctime could be detected. Finally, in the case of OSSEC, the integrity verification function was used. The integrity verification function demonstrated that the hash values before and after the execution of the ransomware were changed. Fig. 10 displays the differences in the detection methods of the EDR tools used in this study.

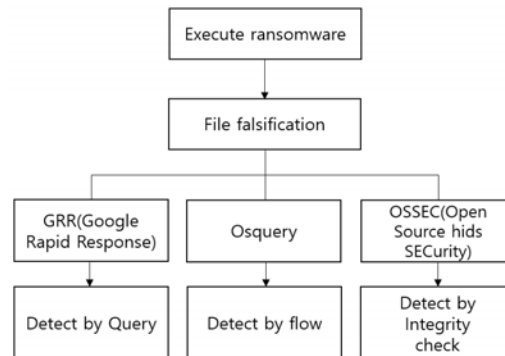


Fig. 10. Different detection methods of endpoint detection response tools.

As illustrated in Fig. 10, each EDR tool uses its own function to send a request to detect a file infection caused by ransomware. The GRR can detect files and directories through Flow, and osquery can send queries to determine the changes. Moreover, OSSEC detects changes in files using integrity check, a real-time monitoring function. Through this experiment, detecting ransomware was possible using each feature of various open-source EDRs.

V. CONCLUSION

Cyberattack detection is an essential element for incident response. Therefore, in this study, tests were conducted regarding whether each EDR tool can detect file changes caused by ransomware. Moreover, GRR, osquery, and OSSEC were chosen as representative EDR tools. Detection results were displayed in the form of notifications or logs when all files in a specific directory were encrypted, indicating that all three EDR tools in the Linux environment are capable of detecting ransomware. Through this open source-based threat detection, it is possible to determine the point at which the ransomware was executed and to analyze the attack method. Although this study has not been tested under various environmental change conditions, in future work, the efficiency of the EDR tool will be compared by measuring the accuracy and speed with which EDR detects ransomware when EDR is deployed to large clients.

REFERENCES

- [1] S. Aurangzeb, M. Aleem, M. Iqbal, and A. Islam, "Ransomware: A Survey and Trends," Journal of Information Assurance and Security, Vol. 6, No. 12, pp. 48-58, 2017.
- [2] O. Aslan and R. Samet, "A Comprehensive Review on Malware Detection Approaches," IEEE Access, vol. 8, pp. 6249-6271, 2020.

- [3] P. H. Meland, Y. F. F. Bayoumy, and G. Sindre, "The Ransomware-as-a-Service economy within the darknet," *Computers & Security*, vol. 92, 2020.
- [4] Arora, Arsh, Hasan, Ragib, and Warner, Gary, "Obsolete Ransomware: A Comprehensive Study of the Continued Threat to Users," *International Journal of Internet Technology and Secured Transactions*, vol. 7, 2020.
- [5] A. H. Mohammad, "Ransomware Evolution, Growth and Recommendation for Detection," *Modern Applied Science*, vol. 14, no. 3, p. 68, 2020.
- [6] F. Tang, B. Ma, J. Li, F. Zhang, J. Su, and J. Ma, "RansomSpector: An introspection-based approach to detect crypto ransomware," *Computers & Security*, vol. 97, 2020.
- [7] W. U. Hassan, A. Bates, and D. Marino, "Tactical Provenance Analysis for Endpoint Detection and Response Systems," *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1172-1189, 2020.
- [8] D. Delija, "Remote digital forensics practices," *International Journal of Digital Technology & Economy*, vol.2, no. 1, pp. 27-36, 2017.
- [9] "GRR Flows," GRR Flows - GRR documentation. [Online]. Available: <https://grr-doc.readthedocs.io/en/v3.2.1/investigating-with-grr/flows/what-are-flows.html>. [Accessed: 07-Nov-2020].
- [10] "Anomaly detection with osquery," osquery. [Online]. Available: <https://osquery.readthedocs.io/en/stable/deployment/anomaly-detection/>. [Accessed: 08-Nov-2020].
- [11] "Syscheck," OSSEC. [Online]. Available: <https://www.ossec.net/docs/docs/manual/syscheck/index.html>. [Accessed: 08-Nov-2020].
- [12] leonv024, "leonv024/RAASNet," GitHub. [Online]. Available: <https://github.com/leonv024/RAASNet>. [Accessed: 08-Nov-2020].
- [13] H. Rasheed, A. Hadi, and M. Khader, "Threat Hunting Using GRR Rapid Response," *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 155-160, 2017.
- [14] Christopher Hurlless, "Exploring Osquery, Fleet, and Elastic Stack as an Open-source solution to Endpoint Detection and Response," *SANS Institute Reading Room site*, 2019. Available: <https://www.sans.org/reading-room/whitepapers/detection/paper/39165>. [Accessed: 08-Nov-2020].
- [15] K. Cabaj, M. Gregorczyk, and W. Mazurczyk, "Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics," *Computers & Electrical Engineering*, vol. 66, pp. 353-368, 2018.



Sun-Jin Lee is a student of the Integrated B.S./M.S. course in the Department of Convergence Security Engineering in Sungshin University, Seoul, Korea. Her current research interests are in the area of deep learning, Internet of Things, malware detection, voice security, image security, and video security.



Hye-Yeon Shim is a student of the Integrated B.S./M.S. course in the Department of Convergence Security Engineering in Sungshin University, Seoul, Korea. Her current research interests are in the area of artificial intelligence, deep learning, malware detection, and programming.



Yu-Rim Lee is a student of the Integrated B.S./M.S. course in the Department of Convergence Security Engineering in Sungshin University, Seoul, Korea. Her current research interests are in the area of artificial intelligence, threat defense, malware detection, and Internet of Things.



Tae-Rim Park is a student of the Integrated B.S./M.S. course in the Department of Convergence Security Engineering in Sungshin University, Seoul, Korea. Her current interests are in the areas of artificial intelligence, network security, malware detection, cloud computing, and Internet of Things.



So-Hyun Park received her B.S. degree in Convergence Security from Sungshin University, Seoul, Korea, in 2020. She joined Convergence Security Engineering Laboratory, Sungshin University, Seoul, Korea in 2020, where she is currently pursuing M.S. Her research interests include wireless networks and communications security, digital forensics, cyber security, Internet of Things (IoT) security, and voice recognition security.



Il-Gu Lee received his B.S. degree in electrical engineering from Sogang University, Seoul, Korea, at 2003, and his M.S. degree in the Department of Information and Communications Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, at 2005. He received his PhD degree in the Graduate School of Information Security in Computer Science & Engineering Department from KAIST at 2016. He is a professor at the Department of Convergence Security Engineering, Sungshin University (SU), Seoul, Korea. Before joining SU in March 2017, he was with the Electronics and Telecommunications Research Institute (ETRI) as a senior researcher from 2005 to 2017, and served as a principal architect and a project leader for Newratek (KR) and Newracom (US) from 2014 to 2017. His current research interests are in the area of wireless/mobile networks with an emphasis on information security, networks, wireless circuit and systems. He has authored/coauthored more than 55 technical papers in the areas of information security, wireless networks and communications, and holds about 160 patents. He is also an active participant of and contributor to the IEEE 802.11 WLAN standardization committee.