Security Threats to Hadoop: Data Leakage Attacks and Investigation

Xiao Fu, Yun Gao, Bin Luo, Xiaojiang Du, and Mohsen Guizani

Abstract

As one of the most popular platforms for processing big data, Hadoop has low costs, convenience, and fast speed. However, it is also a significant target of data leakage attacks, as a growing number of businesses and individuals store and process their private data in it. How to investigate data leakage attacks in Hadoop is an important but long-neglected issue. This article first presents some possible data leakage attacks in Hadoop. Then an investigation framework is proposed and tested based on some simulated cases.

Introduction

Hadoop is one of the most popular platforms for big data storage and analysis. It is widely used in many fields, such as manufacturing, healthcare, insurance, and retail because of its powerful processing capacity, huge storage capacity, scalability, and relatively low cost. Nowadays, a growing number of individuals and businesses store and process their private data in Hadoop, and this valuable data has become an important target of hackers [1].

In order to prevent these types of attacks, investigating them and reconstructing the entire scenario is very important. Based on the forensic results [5] [6], the vulnerability of Hadoop can be found and the attackers can be accused. Although the Hadoop investigator often faces many challenges, there is still little research work in this area. Current challenges faced by the Hadoop investigator include, but are not limited to, how to locate the data leakage node among thousands of Hadoop nodes, how to obtain reliable evidence in a complex and rapidly changing Hadoop environment, and how to investigate attacks based on Hadoop audit logs, which usually contain a large quantity of redundant and multi-user data.

Considering these challenges, in this article, we first present some possible data leakage attacks in Hadoop, then analyze the difficulties of investigation. After that, an investigating framework is proposed and tested based on simulated cases. The framework is composed of the data collector and data analyzer. The data collector collects Hadoop logs, Fsimage files, our own monitor logs, and other information from each node actively or on demand, and transmits them to the data analyzer. Then, the data analyzer analyzes the data with automatic methods to find the stolen data, find the attacker who stole this data, and reconstruct the crime scenario.

Related Work

Current Hadoop security research mainly includes trusted audit mechanisms, access control, data encryption, and so on. In [1], an architecture was proposed to fight against advanced persistent threats (APT) targeted against data stored in the HDFS (Hadoop distributed file system). This architecture is based on the trusted platform group (TPM) and trusted computing group (TCG). With the help of this architecture, all of the operations triggered by users can be audited. In this way, suspicious actions can be discovered and evidence can be retrieved for future investigation. However, it results in a serious impact on performance and a huge size of logs, which make it impractical.

In [2], a complicated access control mechanism was adopted to ensure the security of Hadoop. It protects data in Hadoop from unauthorized access, accidental leakage and loss, and breach of tenant confidentiality. "ACL Access Control" and "Kerberos" are two of them that have been adopted by the new versions of Hadoop. Although this can make Hadoop more secure, it is not able to prevent attacks if criminals use legal user accounts, and nothing can be done for direct data access in the operating system layer because these mechanisms are only for the application layer.

In [3], a secure Hadoop architecture was proposed that adds encryption and decryption functions in the HDFS. Through this method, data in the HDFS will not be readable even if it is stolen because the attacker does not have the secret key. Data is protected in this way. Although it is a fundamental solution for securing Hadoop, what cannot be ignored is the impact on performance and the fact that this approach does not guard against attackers who use a legal account.

Unlike [1], our work concentrates on investigating (including evidence collection and analysis) data leakage attacks in Hadoop. Although [2] and [3] present some good solutions to prevent data leakage, they are far from silver bullets. This type of attack still takes places frequently, so research on how to investigate them after they happen is necessary and important. We have not found any other work on such an issue.

Data Leakage Attack in Hadoop

The data leakage attacks in Hadoop mainly include but are not limited to the following categories.

Application layer data leakage. This means attackers can obtain private data by application-layer vulnerability or malware. For example, a vulnerability in the current Hadoop audit mechanism is that it only records the operation type, time, and content, but no information on who did this operation. Suppose in a company, Alice, Bob and Cindy belong to a group named Hadoop, which is responsible for managing their compa-

Xiao Fu, Yun Gao, and Bin Luo are with Nanjing University.

Xiaojiang Du is with Temple University.

Mohsen Guizani is with the University of Idaho.

Digital Object Identifier: 10.1109/MNET.2017.1500095NM

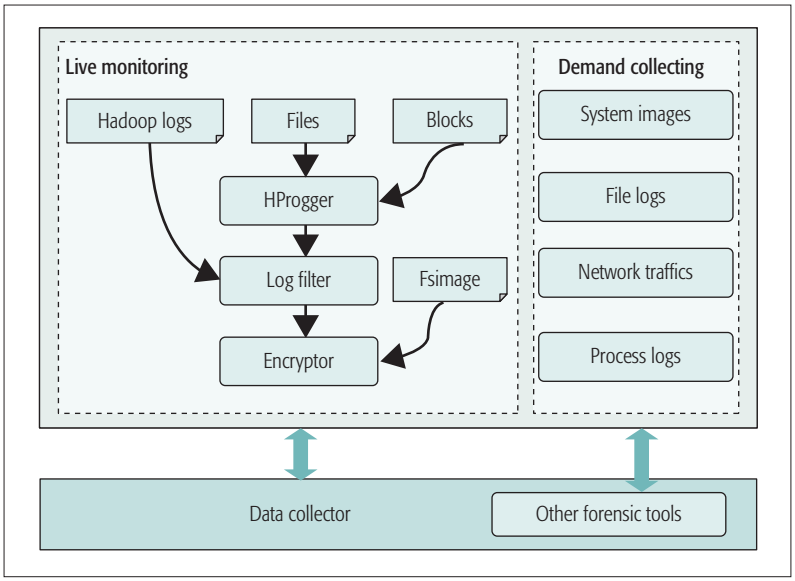

FIGURE 1. Architecture of the data collector.

ny's Hadoop system. One day, Bob stored a file named star-project.txt in the HDFS, and set the permission of this file to group readable. Soon after that, Bob found the content of his file was known by his rival, yet the content was not leaked by him, so who is the traitor, Alice or Cindy? We cannot find the answer just from Hadoop logs because they do not record who did this. The only clue we have is that Bob knows which file was stolen. In addition, the Hadoop audit logs might have been tampered.

Operating system layer data leakage. This means if attackers have the permission to log-in to the host operating system of a Hadoop node, they can bypass the monitor of Hadoop and steal data (not only the HDFS file block but also the temporary result of the MapReduce task) directly in the OS layer. For example, Bob stored a file named companyA.7z in the HDFS, and this file is larger than 64 Mb, so it was divided into several blocks, each block possibly being saved in different machines. Both the Hadoop administrator and the root user have write permission to all the blocks. Hence, if the attacker wants to steal the file, they do not have to get permission in the HDFS, they just need to get the locations of every block with the help of the name node logs (or through other ways such as the network monitor and so on), then steal the block directly from the physical machine if they have root permission of the host OS. The only clue we have is which file was stolen. The Hadoop audit logs might also have been tampered with by the attackers.

To investigate the cases above, we are faced with the following challenges. First, Hadoop clusters often contain hundreds or thousands of nodes. When attacks happen, it is almost impossible to investigate all the nodes because either it requires too much time or the evidence has been changed during the investigations. Most attacks involve only a few nodes, and we need to find an efficient way to locate the nodes being attacked and concentrate our resources on them. Second, the Hadoop environment is complex and fast changing. The evidence may be tainted by new data or tampered with by attackers. Hence, how can we obtain reliable evidence from such an environment? Third, current Hadoop audit logs are far from sufficient in investigations as they lack the information on who did the crimes. Furthermore, these audit logs contain a large quantity of redundant and multi-user data, making evidence analysis extremely difficulty.

An Investigation Framework for Hadoop Forensics

Considering the above challenges, an investigation framework aimed at data leakage attacks in Hadoop was proposed. In fact, it can also be extended to investigate other kinds of attacks in Hadoop. This framework is composed of many data collectors and one data analyzer. The data collector is located in the kernel of the host operating system in Hadoop nodes (mainly data nodes). It is designed based on a demand collection policy. First, it actively monitors the accesses to important data on each node and transmits these behavior logs to the data analyzer. Then it collects the required evidence (such as disk images and network traffic logs) from specific nodes according to the analysis results. The data analyzer is located in a specific forensic server. It analyzes the data with automatic algorithms to find the stolen data, find the attackers who stole this data, and reconstruct the entire scenario. It also sends commands to specific collectors to obtain more evidence.

There are serval advantages in the above design. Through the cooperation between the data collectors and the data analyzer, the data leakage attacks can be detected immediately by the live monitor and the automatic analysis algorithm, and the node where it happened can be located quickly according to the information in the collected behavior logs. The demand collection policy can reduce cost because the large-sized evidence (e.g., disk images) is only collected on several nodes based on analysis results. The live kernel-level monitors ensure the evidence is collected before contamination, and the collector itself is transparent to normal Hadoop users and attackers. Moreover, all evidence is stored in an independent and carefully protected forensic server. This makes the evidence more reliable.

Data Collector

The architecture of the data collector is shown in Fig. 1. The data it collects includes Hadoop logs, the Fsimage file, our own monitor (i.e., HProgger) logs and images or logs of files, processes, networks, and system. In the live monitoring stage, data collectors on each node obtain Hadoop logs, HProgger logs, and the fsimage first. Then, the log filter changes them to a unified format. Finally, before transmitting them to the server, the log encryptor encrypts them to keep their credibility. In the demand collecting stage, the data collector and even some other forensic tools will collect file logs, network logs, process logs, and system images, and then transmit them to the server for analyzing. This data can also be encrypted before transmission. The two-stage design gives our framework the ability to collect data according to what the investigation needs, with reduced consumption and improved efficiency.

Hadoop logs include name node logs, job tracker logs, data node logs, and task tracker logs. Our framework will mainly focus on name node logs and data node logs because data leakage attacks are mainly recorded in them.

The Fsimage is the image file of the Hadoop system. Unlike general system image files, the fsimage only records the data directory and data distribution. In simple words, the fsimage records the meta information of files in the file system and the distribution of blocks.

HProgger is our own monitor tool which is implemented based on Progger [4]. Progger is an open source data monitor tool running in the kernel of Linux. It modifies the addresses of certain system calls in the system call table, allowing it to record the invocations of these calls. Progger is able to help us acquire more reliable evidence because data leakage attacks in Hadoop will inevitably invoke some system call to access certain files and directories. Although Progger is powerful in auditing, it causes a serious downgrade in performance and delays every operation. Additionally, the amount of logs that Progger generates is unacceptable, i.e. it generates hundreds of megabytes of logs in just a few minutes. To make it applicable in Hadoop forensics, performance must be improved and the number of logs must be reduced as much as possible. Thus, we design HProgger (i.e. Hadoop-Progger). It is improved in several aspects:

- Only three system calls (i.e. OPEN, COPY, MOVE) will be recorded.

2. The log format is minimized; only necessary information such as operation type, time, operator, and so on remains.
3. Instead of monitoring all the directories, only those associated with the Hadoop system, such as the directory where Hadoop stores HDFS files, are retained. Hence, compared to Progger, HProgger is lighter and more applicable for Hadoop forensics.

Data Analyzer

The data analyzer is composed of a decryptor, a log database, a fsimage manager, an automatic detector, and the management and presentation module (as shown in Fig.2). When data arrives at the analyzer, it is first processed by the decryptor. Then Hadoop logs and HProgger logs are stored in the log database and the fsimage files are stored in the fsimage manager. Users can query and manage the logs and the fsimage in the management and presentation module. This data is formatted before presentation. The automatic detector analyzes the logs automatically to find the data that was stolen, the attacker who stole the data, and the process of the attacks. The results will also be presented in the management and presentation module.

In order to detect data leakage attacks as soon as possible, some automatic detection algorithms are designed. For an OS-layer data leakage attack, the detection algorithm works in four dimensions: abnormal directory, abnormal user, suspicious block proportion, and abnormal operation. The algorithm detects abnormal directories and abnormal users in the file datasets to find out if suspicious blocks exist, and it calculates the suspicious block proportion. In the meantime, the algorithm monitors the trend of operations in the block datasets to detect abnormal values, and also calculates the suspicious block proportion. If any of these four dimensions gives a warning, then attacks may have happened and the investigators can continue their investigation according to the warning information.

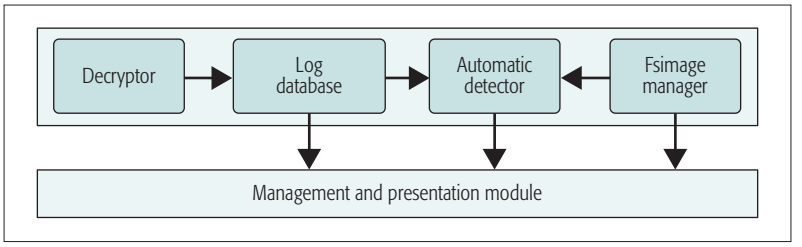

FIGURE 2. Architecture of the data analyzer.

Abnormal Directory (AD): Normal Hadoop system file operations will involve fixed directories because these directories are configured by Hadoop system profiles. Hence, in collected logs, if any directory that is out of this range is found, an attack may have happened. The suspicious blocks can thus be found by analyzing records that contain these abnormal directories.

Abnormal User (AU): By studying the HProgger logs, it can be concluded that all of the Hadoop system file operations except copying files from the HDFS to a local machine will involve only one operator, i.e. the Hadoop super user. Hence, if any other user instead of the Hadoop super user is found in HProgger logs, an attack may have happened. The suspicious blocks can thus be found by analyzing records that contain these abnormal directories. On the other hand, if an attacker wants to steal data directly from the operating system, they must have root permission or Hadoop super user permission. However, because more than one user can switch to the root, how can the real user who conducted the operations in root permission be found? Fortunately, HProgger is able to keep track of all users' data as long as they log in to the system, so the HProgger logs can be analyzed to find the real user.

Abnormal Operation (AO): Every Hadoop system operation in the HDFS is composed of reading files from the HDFS and writing files into the HDFS. By studying the HProgger logs generated by all kinds of Hadoop system operations, it can be concluded that every block is written only once and read at least twice during its life cycle (other backups of this block are not considered). However, if an attacker wants to steal a block from the physical machine, they need to copy (move) it to another directory and sometimes may rename the block, so if any of these operations are found, the attack may have happened. Moreover, the attack scenario can be reconstructed based on these operations. The attacker and the lost files may also be identified based on them.

Block Proportion (BP): Let SusBlockNum (FileID) represent the number of suspicious blocks in a file dataset and AllBlockNumber (FileID) represent the total number of blocks in a file dataset. We define block proportion (BP) as the result of dividing SusBlockNum (FileID) by AllBlockNumber (FileID). Under normal circumstances, the value of BP should be 0 as no suspicious block should be found. The larger BP is, the greater the probability there has been an attack, so if BP is larger than 0, the algorithm will give a warning.

Fsimage is the image file of Hadoop system. Unlike general system image files, fsimage only records the data directory and data distribution. In simple words, fsimage records the meta information of files in file system and the distribution of blocks.

The above describes the automatic detection algorithm for an OS-layer data leakage attack. Detection of an application-layer attack is much easier based on the HProgger log. For example, regarding stealing behavior using a legal user account, we are not able to find the thief based on Hadoop audit logs because necessary information is missing. However, based on HProgger, the thief can be identified according to the following steps:

Step 1: Find the HDFS blocks according to the file name and obtain their block ID from Hadoop logs or fsimage files.

Step 2: Filter out HProgger logs that contain "copyToLocal" or other necessary key operation types.

Step 3: Filter out the Hadoop logs that contain the block ID found in Step 1, then map these logs to the HProgger logs obtained in Step 2 based on their operation time. If such mapping records are found, the attacker name can be found in the relevant HProgger logs, and these mapping records can support each other.

Case Study

In order to test our method, a small Hadoop cluster using Virtual Box including one master node and 15 slave nodes is set up. The node environment is an i5-4590 3.3 GHz 2G Ubuntu 12.04 and Hadoop Version is 1.2.1.

Application-Level Data Leakage Investigations

We first conduct an investigation at the Hadoop application level. Suppose attackers stole a secret file named star-project.txt using a legal account. As we mentioned before, the Hadoop audit log cannot record the operator and there might be CSP employees who are authorized to modify the Hadoop logs and the Fsimage, so both the Hadoop logs and Fsimage may be contaminated. However, this case can be investigated based on the HProgger log, the Hadoop log, and the Fsimage according to following steps.

Step 1. Find corresponding block IDs by file name in the Hadoop logs that come from the name node. The records show that block 8241916282566986668 belongs to the file star-project.txt.

Step 2. In application-level attacks, criminals access data through Hadoop commands, e.g. some Hadoop shell commands for file operations, such as CopyToLocal, so we can filter out this type of record from the HProgger logs. In this case, we can know from the filtered logs that in 21:04:28, User Alice used the CopyToLocal command to copy a file to a local machine, renamed it to abc.txt, and put it to /home/Alice. However, we cannot confirm that abc.txt is star-project.txt.

Step 3. Find the records containing the investigated file name or block IDs around the key records found in Step 2 in the HProgger logs, then obtain the timestamp of this operation. For example, we found a record showing that block 8241916282566986668 was OPEN in 21:04:28, and this record is just the former one of the key records we found in Step 2.

Step 4. Search the records in the Hadoop logs according to the timestamps and block ID we found in the above steps. The results show that in 21:04:28, a HDFS_READ was executed on block 8241916282566986668 and there were no other records at the same time. This record and the two records we found in HProgger can prove to each other that the criminal is User Alice. She copied star-project.txt to local by CopyToLocal and renamed it abc.txt.

In conclusion, Alice is the attacker. We can conduct second-stage forensics and find more information about this attack, for example, obtaining the network traffic in that node to detect whether Alice transmitted this file to another location. However, if attackers tampered with the Hadoop logs from the data node or the name node, how do we investigate such a case?

Hadoop logs from the data node are tampered with: In this case, we cannot obtain the relevant record from the Hadoop logs as was done in Step 4. However, any changes to the file or directory can be detected by HProgger. If we set HProgger to monitor /usr/local/hadoop/logs/, which is the directory where Hadoop logs are stored, we can find the illegal tampering from attackers.

Hadoop logs from the name node are tampered with: In this case, we cannot obtain the block ID based on file name as was done in Step 1. Fortunately, we can obtain this relation from the fsimage. That is why the data collector collects the fsimage from each node.

The fsimage is broken: If the Hadoop logs and the fsimage both cannot be trusted, how do we investigate such a case? In this situation, we can first obtain the fsimage from the secondary name node, which is the backup for the name node and synchronizes with it periodically in most Hadoop environments. Second, we can use the old version of the fsimage that was saved by the fsimage file manager.

In conclusion, even if the Hadoop logs and fsimage files are tampered with, the investigator can still find the attackers based on our method.

Operating System-Level Data Leakage Investigations

In this section, we conduct an investigation at the operating system level. Suppose an attacker knew the location of a secret file companyA.7z from the name node logs or some other way. Then they could read this file directly from these data nodes by logging in to the host OS. Such an operation cannot be monitored by the Hadoop audit mechanism, but we can find this attacker based on the HProgger logs, the Hadoop logs, and the Fsimage according to the following steps.

Step 1. Find the corresponding block IDs by file name in the Hadoop logs that are from the name node. Different from the first experiment, we obtained three block IDs, i.e., 4067922487609870000, 6299596445748830000 and 7132029012670650000. That is because companyA.7z is larger than 64 Mb, so it is divided into three blocks by Hadoop.

Step 2. Find the HProgger records by block ID. These records show that User Alice copied block 5255825402 465249432 from the Hadoop system directory to home/alice at 11:41:44. We find Alice did the same thing to two other blocks that belonged to companyA.7z, but in different data

nodes. That is to say, Alice copied all the blocks of companyA.7z directly from the operating system in different machines. We can conclude that Alice stole companyA.7z. We can conduct a second-stage forensic investigation and find more information about this attack, e.g., obtain the network traffic in that node to detect whether Alice transmitted this file to another location.

Similar to the first experiment, if the Hadoop logs cannot be trusted, we can obtain the mapping between the file and the block through the fsimage, and if the fsimage is broken, we can obtain the old version from the secondary name node or the fsimage file manager.

We have shown how to find attackers when the only clue is the name of the file that was stolen, but how do we find which file was stolen? Earlier an automatic detection algorithm was proposed. In order to test its efficiency, we run the algorithm every five minutes in our cluster (the frequency can be customized). Figure 3 and Figure 4 show the monitoring results of companyA.7z during a period of 30 minutes.

Figure 3 shows that all three dimensions of the file companyA.7z changed from 21:20 to 21:55, and all values of the three dimensions are greater than their normal value. For other files, the values of all dimensions remained unchanged the entire time.

Figure 4 shows the AO values of all three blocks of the fle companyA.7z growing to 7 or 8, both greater than the normal value 6. For the blocks of the other files, the values remained at 6 all the time. Hence, suspicious operations must have happened to companyA.7z during 21:20 to 21:55. So the detailed suspicious HProgger logs were found, which showed that Alice copied block 6299596445748830000 and block 713202901267065000 from /app/hadoop/tmp/dfs/data/ current to /home/abc/, and she copied block 40679224 87609870000 to home/Alice and then to home/abc. All the blocks of companyA.7z were stolen by Alice.

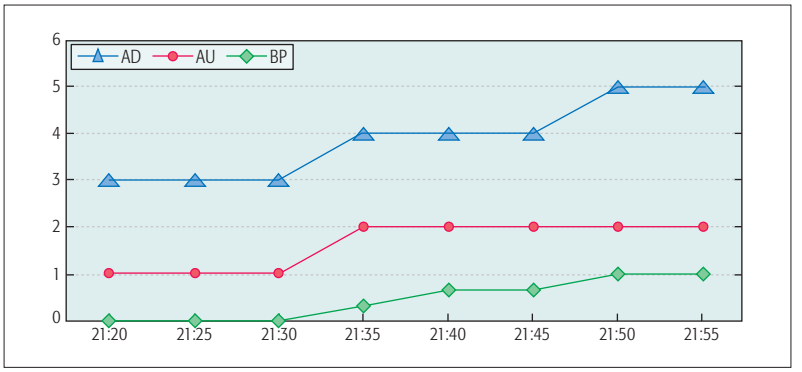

FIGURE 3. Experimental results of companyA.7z, file dataset.

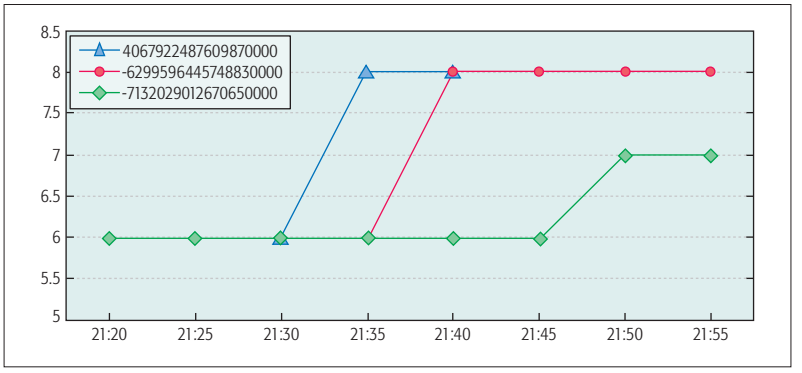

FIGURE 4. Experimental results of companyA.7z, block dataset.

Conclusion

This article presents a forensic framework including an on-demand data collection method and an automatic analysis method for data leakage attacks in Hadoop. It collects data from the machines in the Hadoop cluster to our forensic server and then analyzes them. With the automatic detection algorithm, it can find out whether there exist suspicious data leakage behaviors and give warnings and evidence to users. This collected evidence can be used to find the attackers and reconstruct the attack scenarios. Some simulated investigating cases have shown its efficiency.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (61100198/F0207, 61100197/F0207)

References

[1] J. Cohen and S. Acharya, "Towards a More Secure Apache Hadoop HDFS Infrastructure, Network and System Security," *Lecture Notes in Computer Science*, Berlin: Springer, 2013, vol. 7873, pp. 735–41.
[2] D. Das and O. O'Malley, "Adding Security to Apache Hadoop," *Hortonworks Technical Report*, 2011.
[3] S. Park and Y. Lee, "Secure Hadoop with Encrypted HDFS," *GPC 2013*, Seoul, Korea, 2013, May 9–11.
[4] R. K. L. Ko and M. A. Will. "Progger: An Efficient, Tamper-Evident Kernel-Space Logger for Cloud Data Provenance Tracking," *CLOUD 2014*, Anchorage, AK, 2014, June 27–July 2.
[5] Y. Cheng *et al.*, "A Lightweight Live Memory Forensic Approach Based on Hardware Virtualization," vol. 379, pp. 23–41, *Elsevier Information Sciences*, Feb. 2017.
[6] X. Fu, X. Du, and B. Luo, "Data Correlation-based Analysis Method for Automatic Memory Forensics," *Security and Communication Networks*, Wiley, article first published online: Sept. 2015, DOI: 10.1002/sec.1337

Biographies

Xiao Fu [M] (fuxiao@nju.edu.cn) is an assistant professor at the State Key Laboratory for Novel Software Technology at Nanjing University. She received her B.E., M.S., and Ph.D. degrees from Nanjing University in 2002, 2005, and 2010, respectively, all in the Department of Computer Science and Technology. Her research interests are security and cloud computing.

Yun Gao (gaoyun@nju.edu.cn) is a master student at the State Key Laboratory for Novel Software Technology at Nanjing University. He received his B.E. degree from Nanjing University in 2012 in the Software Institue. His research interests are cloud and big data security.

Bin Luo [M] (luobin@nju.edu.cn) is a professor at the State Key Laboratory for Novel Software Technology at Nanjing University. He received his B.E., M.S., and Ph.D. degrees from Nanjing University in 1989, 1992, and 2000, respectively, all in the Department of Computer Science and Technology. His research interests are systems and networks.

Xiaojiang Du [SM] (xjdu@temple.edu) is an associate professor in the Department of Computer and Information Sciences at Temple University. His research interests are security, cloud computing, wireless networks, and computer networks and systems. He has published over 200 journal and conference papers.

Mohsen Guizani [F] (mguizani@ieee.org) is a professor and the Chair of the ECE Dept. at the University of Idaho, USA. He has published six books and approximately 200 articles in the areas of wireless networking and communications, mobile computing, optical networking, and network security. He is the founder and editor-in-chief of *Wiley Wireless Communications and Mobile Computing Journal*.