In-Storage Anti-Virus System via On-Demand Inspection

Jaehwan LEE, Member, Youngrang KIM, and Ji Sun SHIN, Nonmembers

SUMMARY We propose a new signature-based, on-demand anti-virus solution using in-storage processing (ISP) to inspect the inside of a storage device. In-storage anti-virus systems are able to isolate malicious effects from main computing platforms, and they reduce the system overhead for virus detection. We implement our in-storage anti-virus platform using cost-effective, open-source hardware, and we verify that it is practically applicable to storage devices.

key words: anti-virus, signature-based detection, in-storage processing

1. Introduction

Signature-based [1] virus detection systems are widely used because they provide effective performance and acceptable overheads. Basic signature-based anti-virus algorithms detect viruses by downloading all of the virus signature data to the target system and comparing it to file signatures. As new virus patterns are added, the size of the database increases, which increases detection time. Various studies have sought strategies to prevent performance degradation. Typically, the pre-matching method [2], [3] cloud method [4], or a combination of the two [5], [6] are employed. However, these studies assumed that virus scans would have to be carried out in advance because the comparison overhead was significant. In addition, previous systems needed to open files that might have already been infected in the main computing platform, thus making them vulnerable.

In this paper, we propose a new on-demand, in-storage anti-virus framework to remove the overhead incurred by signature-based virus detection by performing pre-detection in the storage device rather than on a user’s PC equipped with client software. On the proposed architecture, we use a Bloom filter to efficiently detect viruses in a storage device, which is mostly a pervasive hardware environment. Moreover, we check for infections before a main computer opens a file to guarantee virus detection accuracy.

For evaluation, we implement our algorithm in a distributed in-storage computing (DiSC) platform [7] which is cost-effective, open-source based hardware using a single board computer (SBC). We check the accuracy of our virus detection method and measure the latency of our detecting process in DiSC platform. The experimental results show that our inspection processing does not incur significant overhead, and they prove the functional accuracy of the system.

2. Related Works

There are three categories of signature-based virus detection systems. The first is the pre-matching method, which performs a simple check, such as that via a Bloom filter, in advance before carrying out detection against the entire signature database. A Bloom filter is a probabilistic data structure used to check whether an element belongs to a set. Bloom filters can produce false positives; when it is judged that an element belongs to a set, there is a potential error in which it actually does not belong to that set. However, Bloom filters do not produce false negatives. When pre-detecting is used to narrow down the number of suspicious files and conduct further inspections, it is advantageous to save a user’s resources, such as through reducing memory access, by filtering out uninfected files. Typical tools used to do this are HashAV [2], and SigMatch using SigTree [3].

The second detection system category uses a cloud server like CloudAV [4]. In a resource-constrained environment such as mobile device, matching against a signature database can have a major impact on other tasks by placing a significant load on the device. Therefore, it is possible to send the signature data of the file to be checked to cloud server to request further inspection. Then results can be received, and a decision can be made regarding whether the file is infected. This process reduces the aforementioned load.

The third category employs a hybrid algorithm that combines the methods used in the first two categories. Hybrid algorithm performs pre-check through generated signatures and sends only the suspicious files to the cloud server for asking further inspection. It can reduce not only number of further inspections for suspicious files but also load on the device. Our proposal uses this type of hybrid algorithm. Especially, Splitscreen [5] is the baseline of our work, so the next subsection describes Splitscreen in detail.

2.1 Splitscreen

Splitscreen is a typical example of a hybrid method; it de-
detects suspicious files and uses cloud servers. Splitscreen receives a Bloom filter that hashes a portion of its signature on the server, performs pre-checks on that portion, and sends the suspicious pattern to the server. The server compares the entire signature with the suspicious pattern, sends the actual signature data to the client, and performs a close inspection through actual pattern matching to check for infection. This method can improve performance by providing a lighter client to reduce the number of close inspections that are unnecessary when performing the pre-inspection and to perform the close inspection through the cloud server.

2.2 In-Storage Processing

In-storage processing (ISP) is a technique designed to solve the problem of the degradation of overall system performance due to the I/O interface between the storage device and memory [7]. ISP provides a method of transferring data to the main memory after pre-processing or performing specific operations. It can be used in a variety of tasks because it can reduce some of the bottlenecks caused by the I/O as the main memory can take some of the pre-processed data and omit some of the work. Not only can we relieve the performance bottleneck in the interface between storage device and main CPU, but also we can achieve the separation of the processing in CPU from the storage device in the security context. Our work leverage this architectural advantage to improve security level.

3. Architecture and Implementation

In this paper, we propose a new signature-based virus detection method by performing pre-detection in storage using a Bloom filter. In addition, we adopt a hybrid method to use the Cloud like Splitscreen, which is different from CloudAV; CloudAV detects viruses by sending all signature data to the Cloud. In case of Avfs [8], it detects viruses through on-access scanning at file system level. Our proposed system also performs virus detection on demand basis, but performs the detection on the storage device itself using an ISP, not on the file system.

3.1 Architecture

Figure 1 shows the system structure of the proposed method. Our system is composed of client and server architecture. The server maintains a signature database of all malicious files, and it generates the Bloom filter periodically and sends it to the client. The client is a user’s desktop or mobile device with a storage device; it is composed of an ISP-enabled storage system, validation manager, and update manager. The update manager periodically receives the Bloom filter generated in the cloud server, and it updates the Bloom filter to the storage. The validation manager deals with the further inspection of potentially malicious files after pre-scanning. Figure 2 presents the process of the proposed system. The detailed inspection process is described as follows.

- Step 1. When a file open request such as fopen is issued
from a user, a vendor command is generated from the driver of the storage, and Storage is made to perform pre-detection through In-storage processing. Through the vendor command, the storage first checks the cache storing recently checked File ID information. Thus if it can find the file ID in the cache memory, we can assume that the file is benign. The cache is updated when a file is decided as a benign throughout our algorithm. Thus the storage performs the actual file read according to the request.

- Step 2. If the file ID to open does not exist in the cache, the MD5 hash of the target file is generated for pre-detecting. The generated signature data is pre-scanned using the Bloom filter, which is received from the cloud server through Update Manager. If the file is not infected according to the pre-scanning results, the file ID is stored in the cache, and the file is transferred to the client as a response to file open. However, if the file is determined to be suspicious, the storage sends the signature of the file to the validation manager for further inspection.

- Step 3. The validation manager receives the signature of the suspicious file from the storage device and sends it to the cloud server to request close inspection.

- Step 4. If the results of the close inspection indicate an uninfected file, then the validation manager sends the results to the storage. In the storage, the file ID is stored in the storage device’s cache, and the file is transferred to the client. If the results indicate an infected file, the validation manager notifies the client and updates the results to the storage. The storage does not allow the file to transfer to the client.

The advantages of the proposed system are as follows.

- **Separation of main computing system from malicious files in the storage:** The existing virus detection system runs on the user’s PC and performs a direct inspection on a specific file or a check on a PC when a new file is received. However, in the case of the proposed system, it is possible to separate the resources of the user’s PC from the storage system so that we provide higher security for the main computer, since direct inspection is done inside the storage device before the main system loads the malicious code.

- **On-demand detection of the suspicious file before the user opens the file:** We check the file whenever file open/read operations are requested. The previous system has to scan the files periodically rather than on an on-demand basis, which can increase the possibility of infection when scans are carried out.

- **Lowering the demand on resources, including the CPU and memory in the main computer:** A basic pre-scanning process is carried out inside the storage system, so we can reduce the demand on the main computer’s resources in most cases.

- **Ensuring the user’s privacy for non-suspicious files in proactive detection:** It is possible to protect personal information in most benign files because a file’s signature is sent to the cloud server only when that file is determined to be suspicious during the proactive detection process. Even with suspicious files, only the file signature generated via hashing is sent, so the user’s privacy is thoroughly ensured.

- **Reducing unnecessary operations by caching files that are repeatedly accessed:** With repeatedly accessed files, the storage device itself stores them in the cache memory. Thus, inspections are not iterated, and operations such as signature message generation are omitted.

### 3.2 Implementation

In order to evaluate the proposed system, we implement the model using a Distributed in-Storage Computing Platform Using Cost-effective Hardware Device (DiSC)[7] as a ISP model. The DiSC consist with DiSC endpoint and main processing machine (desktop). DiSC endpoint is a modeled a general-purpose ISP device using an inexpensive single-board computer and we using Bananapi computer as the DiSC endpoint. DiSC use a Mesos-based resource manager as task scheduler DiSC endpoint [7]. We use Bananapi DiSC endpoint as the storage device and main processing machine for client PC. The DiSC endpoint and client PC are connected by 1GB Ethernet. Figure 2 is a flowchart of the processes performed during actual operation. In this experiment, the generated MD5 message is used during pre-detecting as the signature of the file. We use a 2.5Mbits array for the Bloom filter with two hash functions-sax and sdbm. The main specifications of Bananapi is octa-core ARM CPU and 2GB LPDDR3 memory.

### 4. Evaluation

For evaluation, we check the two points. First, we compare the actual false-positive probability to verify whether infected and uninfected files are properly detected via the Bloom filter-based pre-detection. Second, we measure the latency of pre-scanning for the ISP to determine the overheads.

The Probability $P_{falsePositive}$ is that all the resulting bit values of the hash function are 1. In this case, the element is positive, but it is not included in the set. This can be expressed as:

$$P_{falsePositive} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$  \hspace{1cm} (1)

where $k$ is the number of hash functions, $m$ is the length of the Bloom filter, and $n$ is the number of elements in the set.

To confirm this, we add the signatures of 1k, 10k, and 100k files to generate a 2.5Mbits Bloom filter. We then check additional 10k files that are not included in the set with the Bloom filter, and we compare the theoretical false-positive probability from Eq. (1) and the rate of the actual
occurrence of the positive error. Table 1 shows the results. When 1k and 10k elements are included, the false-positive probability is low and no positive errors occur. However, it can be seen that the false-positive probability and the actual positive error ratio are very close when the Bloom filter includes 100k elements.

In order to verify that our system is suitable for ISP, we measure the latency taken in the Bananapi model. The MD5 message generated by hashing the entire file is used as a signature, and the Bloom filter is pre-created and loaded into the Bananapi memory for detection. The contents of Table 2 show the latency taken to detect the files for the 64Mbyte, 128Mbyte, and 1Gbyte sizes. Looking at each result, we can see that it is proportional to the size of the file.

The designed pre-detection process can be divided into three steps. First, the requested file is read. Then the signature is generated by performing the MD5 hashing on the read contents. Finally, the generated signatures are compared via the Bloom filter. Although we attempt to verify the longest interval by measuring the time for each process separately, it is difficult to acquire precise measurements because the second and third steps occur within a very short period of time. However, it is observed that the first step (reading the data) takes up most of the total execution time, as revealed in Table 2. Table 2 presents the total execution time (Total), the execution time to read the file (Read), and the fio execution time (fio) for the 64MB, 128MB, and 1GB file sizes. fio is a benchmark test tool to measure file read/write times and throughputs [9]. Table 2 shows that the computing time (the execution time to derive the hash signatures) is practically negligible compared to the file read time. In addition, we verify that our read process works well compared to the fio execution time. From this experiment, we conclude that our ISP-based pre-detection algorithm does not incur significant overhead, so it can be applied in practice.

5. Conclusion

In this paper, we propose a new signature-based, on-demand anti-virus solution using in-storage processing (ISP) to detect viruses inside a storage device. Our system uses a hybrid solution that combines Bloom filter-based signature detection and cloud-based close inspection. However, we hide the Bloom filter-based comparison inside the storage device from the main computing system. This enables us to improve the level of security in the main system and reduce resource utilization. We implement the proposed algorithm on our open-source based ISC device and confirm that our system works well functionally, and it executes tasks efficiently considering the file read latency. We used DiSC as our ISP platform and it is scalable to distributed computing clusters. It can help to extend our idea to parallel, distributed file systems such as Lustre and S3. In the distributed file system environment, it may take a lot of time to collect the data divided into blocks and generate the signature data through the network. However, it can be expected that the signature data will be generated in parallel, and the pre-test will be performed more effectively in each storage device. Moreover, we can extend our idea to parallel, distributed file systems such as Lustre and S3. Future research can include the application of ISP-based virus checks to a distributed file system in the cloud data center environment.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. 2015R1C1A1A02036524), and by the GRRC program of Gyeonggi province [GRRC-KAU-2017-B01, “Study on the Video and Space Convergence Platform for 360VR Services”].

References

[1] J.O. Kephart and W.C. Arnold, “Automatic extraction of computer virus signatures,” Proc. 4th Virus Bulletin Int’l Conf., Abingdon, pp.178–184, 1994.
[2] O. Erdogan and P. Cao, “Hash-AV: Fast virus signature scanning by cache-resident filters,” International Journal of Security and Networks, vol.2, no.1/2, pp.50–59, 2007.
[3] R. Kandhan, N. Teletia, and J.M. Patel, “SigMatch: Fast and scalable multi-pattern matching,” Proc. VLDB Endowment, vol.3, no.1-2, pp.1173–1184, 2010.
[4] J. Oberheide, E. Cooke, and F. Jahanian, “CloudAV: N-version antivirus in the network cloud,” USENIX Security Symposium, pp.91–106, 2008.
[5] S.K. Cha, I. Moraru, J. Jang, J. Truelove, D. Brumley, and D.G. Andersen, “SplitScreen: Enabling efficient, distributed malware detection,” Communications and Networks, vol.13, no.2, pp.187–200, 2011.
[6] J. Lee, M.J. Jo, and J.S. Shin, “LigeroAV: A light-weight, signature-based antivirus for mobile environment,” IEICE Trans. Inf. & Syst., vol.E99-D, no.12, pp.3185–3187, Dec. 2016.
[7] J. Lee, J. Kim, and J.S. Shin, “DiSC: A distributed in-storage computing platform using cost-effective hardware devices,” IEICE Trans. Inf. & Syst., vol.E100-D, no.12, pp.3018–3021, Dec. 2017.
[8] Y. Miereskiy, A. Das, C.P. Wright, and E. Zadok, “Avfs: An on-access anti-virus file system,” Proc. 13th USENIX Security Symposium, 2004.
[9] https://github.com/axboe/fio.git, “Flexible I/O Tester”