Amoeba: An Autonomous Backup and Recovery SSD for Ransomware Attack Defense

Donghyun Min, Donggyu Park, Jinwoo Ahn, Ryan Walker, Junghee Lee, Sungyong Park, and Youngjae Kim

Abstract—Ransomware is one of growing concerns in enterprise and government organizations, because it may cause financial damages or loss of important data. Although there are techniques to detect and prevent ransomware, an evolved ransomware may evade them because they are based on monitoring known behaviors. Ransomware can be mitigated if backup copies of data are retained in a safe place. However, existing backup solutions may be under ransomware's control and an intelligent ransomware may destroy backup copies too. They also incur overhead to storage space, performance and network traffic (in case of remote backup). In this paper, we propose an SSD system that supports automated backup, called Amoeba. In particular, Amoeba is armed with a hardware accelerator that can detect the infection of pages by ransomware attacks at high speed and a fine-grained backup control mechanism to minimize space overhead for original data backup. For evaluation, we extended the Microsoft SSD simulator to implement Amoeba and evaluated it using the realistic block-level traces, which are collected while running the actual ransomware. According to our experiments, Amoeba has negligible overhead and outperforms in performance and space efficiency over the state-of-the-art SSD, FlashGuard, which supports data backup within the device.

Index Terms—Solid-state drive (SSD), storage security, ransomware attack

1 Introduction

RANSOMWARE is a type of malware that takes user data files as hostage by encrypting them until the victim pays a ransom. In July and June 2017, massive ransomware attacks occurred and more than 12,000 computers were attacked [1]. Since ransomware may cause immediate financial damages, it is in urgent need to find an effective way to mitigate ransomware. For example, Atlanta had to spend more than $2.6M to recover from the recent ransomware attack in 2018 [2].

Existing techniques detect and prevent ransomware by identifying known behaviors of ransomware such as frequent access of cryptographic libraries and receiving encryptions keys from a remote server [3], [4], [5], [6], [7]. However, these techniques can be evaded [3]. The guaranteed solution to ransomware is data backup. A filesystem may retain backup copies and recover them if they are infected by ransomware [8].

NAND flash memory based solid-state drives (SSDs) are a good candidate to implement a data backup mechanism. It always retains a previous version of data because of its out-of-place update until an internal Garbage Collection (GC) process sweeps stale data. If an SSD backs up files internally, it is transparent to users, and backup copies cannot be destroyed by ransomware and privileged system software.

FlashGuard [9] and SSD-Insider [10] are systems that manage backup data inside the SSD for ransomware attack protection, but each system has limitations in efficiency in detecting ransomware attacks and managing backup data.

For example, the limitations of FlashGuard [9] are manifolded: (1) it generates many backup pages, which results in performance degradation, because it triggers frequent garbage collection; (2) it requires frequent communication between the host and the SSD, because the host makes the final decision of the ransomware attack; (3) manual investigation is required for recovery; and (4) it does not have a method to control the backup space. SSD-Insider [10] adopts a method to detect ransomware attack pattern only by overwrite pattern in particular. However, this system has the problem that it can not distinguish between normal overwrite and ransomware overwrite attack.

In this paper, we propose Amoeba, an autonomous backup and recovery SSD, to solve these problems. Amoeba automatically performs the infection detection of the ransomware, alerts, backup data management and recovery inside the SSD. Unlike FlashGuard [9] and SSD-Insider [10], Amoeba implements data content-based inspection for high-accuracy ransomware detection. Specifically, first, the ransomware detection is accelerated and the communication with the host is minimized by designing a hardware accelerator to calculate the Ransomware Attack Risk Indicator (RARI) for the write request in the SSD. Second, Amoeba supports the fine-grained management of the backup page by determining the backup page, according to the probability of ransomware infection for each page. Consequently, Amoeba manages only one backup page of each page, minimizing GC overhead as well as backup space.

2 Ransomware Attacks and Defense

2.1 Internal Operations for SSDs

SSDs can read and write page by page, and erase block by block. In NAND flash, however, the erase operation unit is a block that is composed of several pages. A whole block should be erased during in-place update, which causes considerable overhead. So, SSDs perform an out-of-place update that writes data to a new empty page and invalidates an existing page to minimize the overhead of the erase. Invalid pages are collected after the erase during Garbage Collection. To perform the out-of-place update, the SSD manages each page which has states of *free*, *valid*, and *invalid*. *free* is a blank page that can be written. When data is written to a page, its state changes to the *valid* state. If data is overwritten on the same logical page afterward, the mapped physical page changes to the *invalid* state and the data is written to a new *free* page, which is to be mapped with the logical page address. A page in *invalid* state cannot be used until it is erased by the GC.

2.2 Data Backup Inside an SSD

Ransomware is a malware that encrypts and takes the user data files as hostage until the victim pays a ransom. Ransomware reads and encrypts the user data, and then overwrites it. Therefore, pages that are infected with ransomware show a typical IO performance pattern of *Read After Write*. The solution to cope with ransomware is to back up the original data in advance and restore it when infected. However, existing methods, which backup and restore data by a file system, requires additional space cost for backup and IO performance overhead to calculate the ransomware infection, and implies the risk of the damage to the backup data copy, due to the intelligent ransomware attacks. To solve this problem, FlashGuard [9] proposes a mechanism to perform data backup in the SSD rather than the OS. If any disk page shows a pattern of *Read After Write*, it saves the backup of that page in the device space to solve the overhead issues of the existing file system.

FlashGuard adds a backup state (*backup*) between *valid* and *invalid* states considering the feature of an SSD device in which the

- *D. Min, D. Park, J. Ahn, S. Park, and Y. Kim are with Sogang University, Seoul 04107, South Korea. E-mail: {mdh38112, dgpark, jinu37, parksy, youkim}@sogang.ac.kr.*
- *R. Walker and J. Lee are with the University of Texas at San Antonio, San Antonio, TX 78249. E-mail: {ryan.walker, junghee.lee}@utsa.edu.*

Manuscript received 1 July 2018; revised 13 Sept. 2018; accepted 5 Oct. 2018. Date of publication 27 Nov. 2018; date of current version 10 Dec. 2018.
(Corresponding author: Youngjae Kim.)
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/LCA.2018.2883431

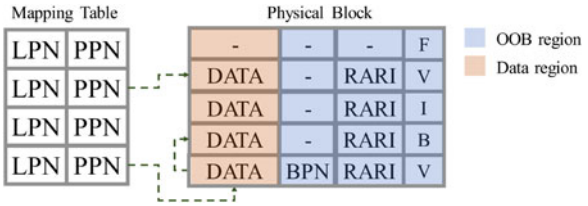


Fig. 1. A description of the OOB metadata area to support the backup mechanism. LPN: logical page number, PPN: physical page number, BPN: backup page number.

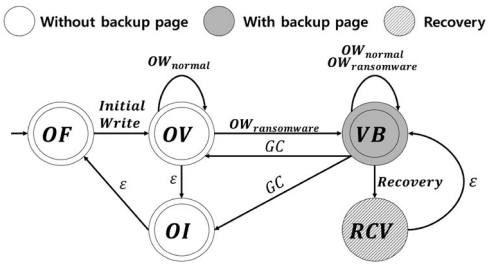


Fig. 2. Non-deterministic Finite Automata (NFA) for Amoeba.

valid state page is not deleted immediately but, is kept in the *invalid* state for a certain period of time. If *Read After Write* pattern, which is suspected to be an access pattern of ransomware, is found, it switches the *valid* state to the *backup* state instead of *invalid* for all overwrites. So, data backup is possible without allocating additional space. However, the *Read After Write* detection method used by FlashGuard has a false positive issue, which makes a backup copy even in normal IO performance. In particular, a backup is generated for every subsequent write, once the *Read After Write* pattern is executed on a specific page. So, the backup space rapidly grows and the space for *invalid* state page decreases, though there is no ransomware infection. It also increases the number of page moves during GC, resulting in greater performance overhead due to GC.

In addition, multiple backup pages of a logical page should be individually checked during the recovery, to find the one that is not infected by the ransomware. Finally, FlashGuard lacks proper mechanisms to control, when the backup page space massively grows in SSDs. FlashGuard merely keeps backup space for a specific period and then deletes it all. However, the speed of the backup page growth in the SSD can differ from the usage frequency, so that there is a risk that the backup space occupies a whole SSD within a short period of time.

3 Amoeba: Design and Implementation

This section describes the backup mechanism of Amoeba. In particular, Amoeba calculates the risk of ransomware attacks for all page write operations. The RARI hardware module is implemented by expanding the DMA controller for internal writing to NAND flash inside the SSD. The RARI module measures the risk of all pages, and it is used to determine whether to back up the page.

3.1 Ransomware Risk Calculation

Amoeba's RARI uses intensity, similarity, and entropy as indicators to determine the risk of potential ransomware attacks.

The intensity can be easily obtained by counting the number of write requests. It is implemented by the firmware, Flash Translation Layer (FTL) at a negligible cost. However, to compute similarity and entropy, every byte of incoming data as well as old data need to be accessed, which may incur excessive overhead if computed by the firmware. Thus, we propose to execute them by a Direct Memory Access (DMA) controller.

An SSD controller usually has an internal DMA to transfer data from temporary buffer of the main memory to the destination NAND flash memory. When a write request arrives from the host interface, it is temporarily stored in the main memory of an SSD. The data is transferred to the destination NAND flash memory by a DMA controller. While the data is being transferred, similarity and entropy are calculated.

To calculate similarity, the DMA controller needs to access both new and old data. Thus, it needs to issue an additional page read to the NAND flash memory where the old data is stored. The DMA controller reads new data from the main memory, old data from the flash memory, and compute their similarity. At the same time, the number of occurrences counts up according to the new data to compute the entropy. Then, the DMA controller writes the new data to its destination flash memory. After transferring all of the new data, it finalizes similarity and entropy, and reports them to the firmware (FTL). Since similarity and entropy are computed while data is transferred, most of their performance overhead is hidden by parallelizing data transfer and computation.

It is difficult to judge ransomware attack by considering only similarity, entropy, and intensity individually. Amoeba uses a strong ransomware classifier that considers similarity, entropy, and intensity. The RARI value is obtained by normalizing the three indicators with the MinMaxScaler method and taking the logistic classification. The RARI value can be formulated by the following equation:

$$\rho = \frac{1}{1+e^{-\zeta}}, \zeta = \alpha * SIM + \beta * ENT + \gamma * INT + \delta. \quad (1)$$

In Equation (1), ζ represents the result of linear equation of the page for the write request. SIM, ENT, and INT mean similarity, entropy, and write Intensity. α, β and γ mean weights and δ means bias. In particular, we obtained all these values using logistic regression from machine learning. ρ is the RARI value of the write request page, which means the possibility of being ransomware and is computed using Equation (1). Amoeba computes the RARI value for each page write operation and identifies ransomware attacks.

3.2 Backup and Recovery

Amoeba uses the Out-of-band (OOB) region of a page to implement the backup and recovery mechanism as described in Fig. 1. The OOB region of a page contains the backup page number (BPN) and the RARI value of the page.

Ransomware Detection and Backup Operation. Fig. 2 describes the state transition behavior mechanism of Amoeba logical pages using Non-deterministic Finite Automata (NFA) and its states and transitions are described in Table 1. The graphical representation of the NFA consists of states (Node) and inputs for state transition (Edge). Table 2 presents a description of the state and input of Amoeba NFA. When the first page write occurs to a free page, its state changes from OF to OV through the state transition. Next, when the page is in the OV state, normal write requests on that page overwrite the page.

Amoeba backup mechanism first checks if the write request is a pattern of the *Read-After-Write* for every overwrite request. Second, it checks whether the RARI value of the current valid page is larger than the defined threshold value. If both conditions are satisfied, Amoeba will regard it as a write request of ransomware attack ($OW_{ransomware}$). If either of the two conditions is not satisfied, it will be regarded as a normal write request (OW_{normal}).

The state transitions of the ransomware overwrite and normal overwrite depend on the state of the page (OV or VB).

- $OW_{ransomware}$ or OW_{normal} upon OV State: Suppose that a logical page $LPN_{(a,ov)}$ has been mapped to a physical page $PPN_{(b,valid)}$.

TABLE 1
A Description of the State and Input of the NFA in Amoeba

(a) Each state description in the NFA.	
Q (State)	Description
OF (Only Free)	Data is not yet written, so it is a Free page
OV (Only Valid)	Data is written and only Valid page exists
OI (Only Invalid)	Data has been invalidated
VB (Valid & Backup)	Both valid and backup pages exist
RCV (Recovery)	Recovering data using the backup page
(b) Each state transition description in the NFA.	
Σ_ε (Input)	Description
Initial Write	Write data to free page
$OW_{ransomware}$	Ransomware overwrite
OW_{normal}	Normal overwrite
Recovery	Recover the valid page using the backup page
GC	Valid page or backup page collected by GC

1) When $OW_{ransomware}$ is written to $LPN_{(a,ov)}$, the physical page $PPN_{(b,valid)}$ in the valid state becomes $PPN_{(b,backup)}$ in the backup state and a new physical page $PPN_{(c,free)}$ becomes $PPN_{(c,valid)}$ in valid state. Therefore, $LPN_{(a,ov)}$ will have both the valid page $PPN_{(c,valid)}$ and the backup page $PPN_{(b,backup)}$. Then, $LPN_{(a,ov)}$ changes to $LPN_{(a,vb)}$ in VB state.
2) If a normal write request OW_{normal} occurs to $LPN_{(a,ov)}$, a page overwrite request is immediately executed without creating a backup page, and remains in OV state. In other words, the existing mapping $PPN_{(b,valid)}$ of the $LPN_{(a,ov)}$ becomes $PPN_{(b,invalid)}$, and the new $PPN_{(c,free)}$ becomes the $PPN_{(c,valid)}$ being mapped to $LPN_{(a,ov)}$.

- $OW_{ransomware}$ or OW_{normal} upon VB state: Suppose that a logical page $LPN_{(a,vb)}$ is mapped to a physical page $PPN_{(c,valid)}$ and has a backup page $PPN_{(b,backup)}$. In other words, a logical page in the VB state has both a physical page in valid state and a physical page in backup state. The backup state is maintained since one or more ransomware requests have been received before. When an overwrite request is received in the VB state, a new physical page $PPN_{(d,free)}$ is allocated and ready for writing. Since Amoeba defines all write requests ($OW_{ransomware}$, OW_{normal}) to ransomware attacks, $PPN_{(b,backup)}$ is still maintained as a backup page, $PPN_{(c,valid)}$ becomes $PPN_{(c,invalid)}$ and $PPN_{(d,valid)}$, and $LPN_{(a,vb)}$ has $PPN_{(d,valid)}$ and $PPN_{(b,backup)}$.

Recovery Operation after Detecting Ransomware. Pages in VB state changed to RCV state, when the recovery is requested. The valid page is substituted by the current backup page. Existing method of FlashGuard unconditionally creates a backup page for every overwrite operation that shows a *Read-After-Write* pattern on a logical page. Therefore, pages infected with ransomware are also stored as backups, and these many backup pages should be individually investigated to find the one to be recovered by the user. However, in the proposed system a logical page has only one backup page, so that it can be restored without the intervention of the user.

GC Selection Algorithm. When the collection operation is conducted by the GC, the existing SSD uses the Greedy algorithm, which selects a block with the smallest number of valid pages as a victim block, in order to reduce the number of valid page copies. In order to prevent the backup page from disappearance, the GC has been designed to consider the backup page to be valid as well. However, to maintain SSD not to be running out of free pages, Amoeba manages the size of backup space by setting a backup level for each backup page. Each backup state has L backup levels following the equation: $Backup\ Level = \lceil \rho \times L \rceil$, depending on the RARI value of the corresponding page in the valid state.

TABLE 2
SSD Simulator Configurations

SSD parameters.	
Pages per Block	64
Planes per package	8
Blocks per plane	2048
PLane block Mapping	Full stripping
Cleaning Policy	Greedy
Page size	8KB
Page Read Latency	0.25
Page Write Latency	0.2
Block Erase Latency	0.025

RARI weight	
α	10.761035
β	-3.3430316
γ	11.890282
δ	-15.452141
error rate	0.001

Invalidation of Backup page	
SSD Occupancy	90%
Backup level	2

As the backup level is lower, the importance of the backup becomes lower since its connected valid page has less possibility of being infected by the ransomware. Accordingly, if the backup state page space becomes too large, the SSD invalidates the low level backup pages, which have a low probability of ransomware. It restricts the size of backup state page space, thereby stabilizing the performance of SSD.

4 Evaluation

4.1 Experimental Setup

Implementation. In order to evaluate the efficacy of Amoeba, we enhanced the DiskSim augmented SSD simulator developed by Microsoft Research [11] and implemented both Amoeba and FlashGuard [9]. In particular, FlashGuard keeps the backup pages until it times out. For simulating this, as the timeout correlates to the number of backup pages, we controlled the number of backup pages from 1 to 8. On the other hand, Amoeba maintains only one backup page per valid page. For more accurate ransomware attack detection, Amoeba uses the RARI hardware module, which is executed at the DMA controller. The RARI calculation for each page is very fast, however, it has the following additional overheads: (1) an additional page read for similarity computation and (2) extra clock cycles to compute entropy after data transfer. Other computations (e.g., counting the number of occurences of bytes) are hidden by parallelizing with data transfer. The overheads are measured by simulation with the register-transfer level model of the DMA controller. The latency to transfer data while computing similarity and entropy on an 8 KB page increases by 2 percent compared to the conventional DMA without computation. All these overheads are counted in the Amoeba DiskSim simulation.

In Amoeba, the RARI value is used to detect ransomware attack. We used a logistic classification method to obtain weights for entropy, similarity, and intensity in Equation (1) using 10 percent of test workloads. The weights are shown in Table 2. Using these values, the ransomware attack detection error of the model was less than 0.1 percent. The backup level used for the GC selection algorithm is 2. The backup page can be only deleted when the SSD occupancy exceeds 90 percent. All the parameters used for the SSD simulation are summarized in Table 2.

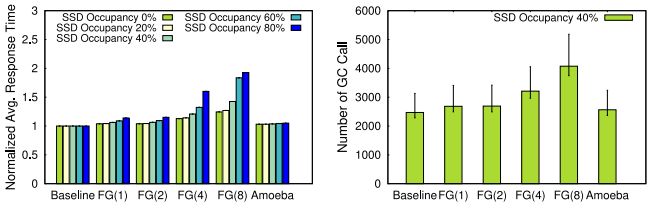
Workloads. We developed an in-house block-level content extractor as a Linux kernel module. The IO block content is required to calculate the RARI value of each block. For trace collection, 3,211 files with a total size of 4.84 GB with characteristics of the files in Table 3, were generated in advance. IO block and content traces were captured using the in-house block content extractor while running the Erebus ransomware [12] and normal synthetic IO application on those files in Table 3. In our collected traces, ransomware IOs are around 70 percent. In particular, the indicator values for each block were calculated by offline analysis with the IO trace.

4.2 Results

Fig. 3 shows the backup performance comparison between FlashGuard and Amoeba. For a fair comparison, the average IO response time was measured while varying the SSD's initial occupancy ratio. In the figure, FG(i), i means FlashGuard with the

TABLE 3
Workload Characteristics for Ransomware Attacks

Type	Number		Size		
	Num	%	Avg (KB)	Total (MB)	%
pdf	1114	34.69	926.58	1008.02	20.31
html	307	9.56	51.70	15.50	0.31
image files	357	11.12	166.97	58.21	1.17
xls	318	9.90	363.07	112.75	2.27
ppt	125	3.89	1483.33	181.07	3.65
doc	74	2.30	428.70	30.98	0.62
zip files	123	3.83	12292.58	1476.31	29.75
others	793	24.69	4404.65	2080.24	41.92
Total	3211	100	1582.745	4963.08	100



(a) Response Time (b) GC Overhead

Fig. 3. Performance comparisons of response times and GC counts for different SSD occupancy ratios. In (b), the error bars are shown in min and max of GC calls across SSD elements in the SSD.

number of backup pages in i. In Fig. 3a, we observe that Amoeba has response times almost similar to the baseline. FlashGuard, on the other hand, shows a drop in performance as the number of backup pages increases and the SSD occupancy ratio increases. From Fig. 3b, we observe that this performance degradation should be attributed to the increase in the number of GC calls. The more backup pages the SSD keeps, the less often free pages are freed, and the GC is more frequently invoked. Specifically, in Fig. 3a, when comparing FG(4) and Amoeba, we see that Amoeba's response time is up to about 12 percent higher than FlashGuard. This is because Amoeba identifies the ransomware by considering both the *Read-After-Write* I/O patterns and the RARI value of each page, while FlashGuard detects ransomware only with the *Read-After-Write* patterns, thus FlashGuard generates more unnecessary backup pages than Amoeba.

Next, we compare the detection accuracy of ransomware and recovery cost for Amoeba and FlashGuard. Table 4a shows the comparisons for the number of FP and FN determinations while running the workloads on each system. Overall, we see that Amoeba shows high ransomware detection accuracy because in Amoeba, the total ratio of FP and FN is 2.58 percent while FlashGuard shows 27.59 percent of the total false ratio. In FlashGuard, IO is determined as ransomware attack if it only follows a pattern of *Read-After-Write*. On the other hand, Amoeba compares both RARI values as well as *Read-After-Write* patterns, thus allowing to identify ransomware more accurately.

The cost of recovery determines the time until the system is restored to normal. In particular, it is very important to quickly find the pages that need to be restored during the recovery. Thus, for evaluations, we compared the total number of accumulated pages searched during the recovery attempts for one to four ransomware attacks. Table 4b shows FlashGuard needs up to four times more search cost than Amoeba.

5 Conclusion

This paper proposes Amoeba, an autonomous backup and recovery SSD to defend against ransomware attacks. Specifically,

TABLE 4
Results for Ransomware Detection Accuracy and Recovery Cost Comparisons

(a) Detection Accuracy for Ransomware.

	Amoeba	FlashGuard
False Positive (FP)	35304 (2.5732%)	371166 (27.051163%)
False Negative (FN)	170 (0.0123%)	7192 (0.52422%)
Total (FP & FN)	35474 (2.5856%)	378318 (27.575384%)

(b) Page Search Cost during Recovery.

Recovery Attempt	FG(1)	FG(2)	FG(4)	FG(8)	Amoeba
1	26416	40005	56338	71892	23303
2	41322	62879	90014	112840	13721
3	55922	85759	121279	152199	51999
4	71410	109850	155151	194420	42527

Amoeba embeds a special RARI computation hardware on the DMA module for fast ransomware detection and performs autonomous backup and recovery within the device. The experimental results demonstrate the backup overhead is negligible while minimizing the error of ransomware attacks.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. NRF-2018R1A1A1A05079398).

References

[1] D. Goodin, "A new ransomware outbreak similar to WCry is shutting down computers worldwide," 2017. [Online]. Available: https://arstechnica.com/security/2017/06/a-new-ransomware-outbreak-similar-to-wcry-is-shutting-down-computers-worldwide/
[2] J. Lambert, "Atlanta spent $2.6m to recover from a $52,000 ransomware scare," 2018. [Online]. Available: https://www.wired.com/story/atlanta-spent-26m-recover-from-ransomware-scare/
[3] N. Scaife, H. Carter, P. Traynor, and K. R. B. Butler, "Cryptolock (and drop it): Stopping ransomware attacks on user data," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, Jun. 2016, pp. 303–312.
[4] C. Moore, "Detecting ransomware with honeypot techniques," in *Proc. Cybersecurity Cyberforensics Conf.*, Aug. 2016, pp. 77–81.
[5] M. M. Ahmadian and H. R. Shahriari, "2entFOX: A framework for high survivable ransomwares detection," in *Proc. 13th Int. Iranian Soc. Cryptology Conf. Inf. Secur. Cryptology*, Sep. 2016, pp. 79–84.
[6] K. Cabaj and W. Mazurczyk, "Using software-defined networking for ransomware mitigation: The case of cryptowall," *IEEE Netw.*, vol. 30, no. 6, pp. 14–20, Nov. 2016.
[7] M. M. Ahmadian, H. R. Shahriari, and S. M. Ghaffarian, "Connection-monitor connection-breaker: A novel approach for prevention and detection of high survivable ransomwares," in *Proc. 12th Int. Iranian Soc. Cryptology Conf. Inf. Secur. Cryptology*, Sep. 2015, pp. 79–84.
[8] A. Continella, A. Guagnelli, G. Zingaro, G. De Pasquale, A. Barenghi, S. Zanero, and F. Maggi, "ShieldFS: A self-healing, ransomware-aware filesystem," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl.*, 2016, pp. 336–347.
[9] J. Huang, J. Xu, X. Xing, P. Liu, and M. K. Qureshi, "FlashGuard: Leveraging intrinsic flash properties to defend against encryption ransomware," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 2231–2244.
[10] S. Baek, Y. Jung, A. Mohaisen, S. Lee, and D. Nyang, "Ssd-insider: Internal defense of solid-state drive against ransomware with perfect data recovery," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst.*, 2018, pp. 875–884.
[11] N. Agrawal, V. Prabhakaran, T. Wobber, J. D. Davis, M. Manasse, and R. Panigrahy, "Design Tradeoffs for SSD Performance," in *Proc. USENIX 2008 Annu. Tech. Conf.*, 2008, pp. 57–70.
[12] T. MICRO, "Erebus Linux ransomware: Impact to servers and countermeasures," 2017. [Online]. Available: https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/erebus-linux-ransomware-impact-to-servers-and-countermeasures