CARDR: DRAM Cache Assisted Ransomware Detection and Recovery in SSDs

Waqar Hassan Mir
waqar.19CSZ0018@iitrpr.ac.in
Indian Institute of Technology, Ropar
Rupnagar, Punjab, India

Neeraj Goel
neeraj@iitrpr.ac.in
Indian Institute of Technology, Ropar
Rupnagar, Punjab, India

Venkata Kalyan Tavva
kalyantv@iitrpr.ac.in
Indian Institute of Technology, Ropar
Rupnagar, Punjab, India

Abstract

Ransomware has emerged as one of the most prevalent and evolving cybersecurity threat in recent times, posing significant risks due to its potential to cause substantial financial damage and its ability to hold important data hostage. Numerous approaches have been proposed to defend against ransomware at the SSD controller. In this paper, we propose a novel SSD-level defense solution called Cache Assisted Ransomware Detection and Recovery (CARDR), that leverages the DRAM cache of the SSD to counter the potential ransomware attacks. CARDR extracts features from both the DRAM cache and incoming I/O requests for the early detection of ransomware using a machine learning model such as a decision tree. Additionally, CARDR effectively reduces data recovery overhead, making it a robust solution against evolving ransomware threats. Our experimental results demonstrate that CARDR helps in the early detection of ransomware. On average CARDR takes ~11 seconds as compared to SSDInsider++ (baseline) which takes ~14 seconds for ransomware detection. For data recovery, CARDR reduces the size of the recovery queue by ~42% in comparison with the baseline.

CCS Concepts

• **Security and privacy** → **Embedded systems security**; • **Hardware** → *External storage*.

Keywords

Ransomware, SSD-Security, Malware Detection, data recovery

1 Introduction

Ransomware is a type of malicious software that targets storage devices like HDDs and SSDs and encrypts files, effectively holding the data hostage. The attackers then demand a ransom payment in exchange for providing the decryption key to restore access to the affected data in the files. Recently, ransomware has emerged as one of the most severe and significant cyber security threat, occurring with alarming frequency [1, 32]. Recent reports indicate that a new ransomware attack happens after every 11 seconds [28].

Over time, ransomware has evolved significantly. Attackers have shifted their focus from targeting individuals to aiming at large corporations and government organizations, driven by the potential for higher financial gains and greater impact [24]. This transition involves the development of more sophisticated ransomware variants capable of infiltrating complex IT systems and evading sophisticated security measures. Attackers also adopted new tactics, such as double extortion, where they not only encrypt data but also threaten to release sensitive information if the ransom is not paid. The advent of Bitcoin further worsened the situation, making it nearly impossible to trace the attacker when ransom payments are made. Despite paying the ransom, there is neither the guarantee of data recovery nor that the attack will not repeat, leaving the victims in a vulnerable position. This evolution has made ransomware a major threat to organizations worldwide.

Numerous techniques have been proposed to counter ransomware attacks. Static approaches identify ransomware binaries before they are executed on the system [17, 25]. Software-based approaches detect ransomware by monitoring changes to the host system during execution [15, 21, 30]. Additionally, backup-based approaches create backup copies for data recovery [5, 31]. Several SSD-based solutions combat ransomware at the SSD controller level by monitoring I/O activities for abnormal patterns, such as sudden spikes in incoming I/O requests and/or increased overwrites [2, 3]. Solutions such as [19, 20] leverage entropy analysis at the SSD controller to identify ransomware by examining the entropy of incoming data, entropy is one of the strong indicators used in detecting ransomware activities but demands additional hardware resources. SSD-based ransomware defense solutions exploit SSD's out-of-place update property to recover pages encrypted by ransomware, assisting the process of restoration of the SSD data to its pre-ransomware state.

Reliably detecting ransomware at the SSD controller is a challenging task due to the limited visibility of system-level information. Each I/O request from the host system to the SSD controller includes minimal context information in its header. To detect ransomware attacks, machine learning models such as decision trees, Support Vector Machines (SVMs), and Convolutional Neural Networks (CNNs) are trained using features extracted from I/O sequences of both benign and ransomware applications. These models classify I/O sequences as originating from either benign applications or ransomware. However, the predictions can sometimes lead to false positives or false negatives, especially when ransomware mimics the behavior of benign applications. In such scenarios, machine learning models embedded within the SSD controller may struggle to effectively protect data from ransomware attacks. Moreover, advanced machine learning models, such as Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs), require significant computational resources that are typically beyond the capabilities of a resource-constrained SSD device. Balancing effective ransomware detection with the limited computational capabilities of SSD devices remains a significant challenge.

Modern SSD devices incorporate an onboard DRAM cache [23] to enhance performance and extend their lifespan. Pages destined for the SSD are first passed to the DRAM cache, and evicted pages from the DRAM cache are subsequently written to the flash memory. When a ransomware program attempts to encrypt multiple files rapidly, the pattern of I/O requests recorded in the DRAM cache

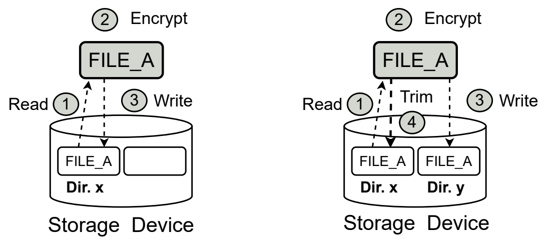


Figure 1: Different classes of ransomware (left side) Class A (right side) Class B.

is likely to show sudden spikes or irregular access patterns. These deviations from normal behavior can serve as warning signs for ransomware attacks and provide clues for detection. By monitoring these patterns, the DRAM cache can provide valuable insights for possible early detection. Additionally, since the data is written to the DRAM cache and stays in the cache for some time before being written to the flash memory, the DRAM cache can also aid in data recovery by holding encrypted data temporarily. This can reduce the total number of encrypted pages that are actually written to the flash memory. Hence, a DRAM cache can serve the dual purposes of detection and data recovery, making it a critical component in safeguarding against ransomware attacks.

Goal. This work aims to leverage the DRAM cache of SSDs for ransomware detection and recovery. Specifically, investigating how the context provided by the DRAM cache, along with the I/O information from the incoming requests can facilitate earlier detection of ransomware and, in turn minimize the overhead associated with data recovery.

The paper makes the following contributions:

- We experimentally analyze the effectiveness of leveraging DRAM cache in SSDs for ransomware detection and recovery, introducing the Cache-Assisted Ransomware Detection and Recovery (CARDR) method.
- We demonstrate that CARDR facilitates earlier ransomware detection compared to the state-of-the-art detection mechanism, namely, SSDInsider++. CARDR detects ransomware within 11.41 seconds on an average, compared to 14.31 seconds for SSDInsider++.
- We find that CARDR reduces the size of the recovery queue by about 42% in comparison with the SSDInsider++, thereby minimizing the overhead associated with the data recovery process.

Organization of the remaining paper: Section 2 delves into the background and ransomware detection/recovery related work. Section 3 outlines the insights and motivation for our approach. The proposed methodology, CARDR, is detailed in Section 4. Followed by experimental evaluation in Section 5, and finally, the paper concludes with Section 6.

2 Background and Related Work

2.1 Background

2.1.1 Different classes of Ransomwares. Rapid emergence of new ransomware variants presents a serious threat, yet they can be broadly categorized into two primary types: in-place write ransomware (Class A) and out-of-place write ransomware (Class B) [3].

Class A ransomware performs in-place write of encrypted pages, causing the host system to write the encrypted page to the same logical address from which it previously read it. This type of ransomware can be effectively detected by monitoring for specific patterns of write-after-read operations. Figure 1 illustrates this process: Class A ransomware reads a file_A from directory x, encrypts it, and writes it back to the same directory x, using same LBA (logical block address). Detecting this repetitive read-encrypt-write pattern to the same LBA is a key indicator of Class A ransomware.

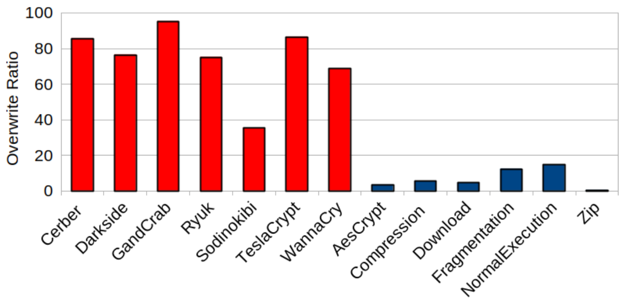
In contrast, Class B ransomware is relatively harder to detect because it writes the encrypted pages to a different directory after reading it from the original directory. As illustrated in Figure 1, Class B ransomware first reads a file from directory x, encrypts the file contents, and then writes the encrypted file to another directory (say) y. Finally, removing the original file from directory x. Detecting this type of ransomware is challenging because it does not perform overwrite operation, which is common in most ransomware variants. Ransomware detectors at the SSD controller can be easily misled by Class B ransomware because a write-intensive application is likely to have a similar activity of reads and/or writes to disjoint LBAs.

2.2 Related Work

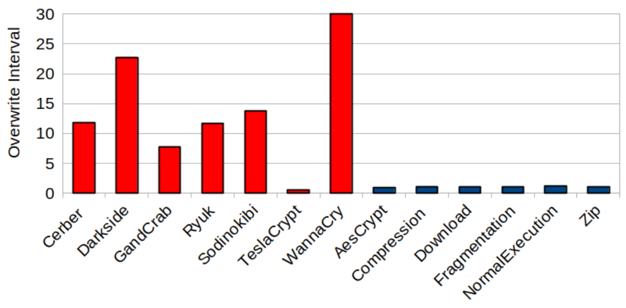
There are various types of ransomware, each with distinct characteristics and methods of attack [8]. The existing literature offers multiple solutions to safeguard against these threats. In this work, we specifically focus on Crypto-Ransomware. The following sections detail the solutions available for combating this particular type of ransomware.

2.2.1 Static analysis. This technique, also known as signature-based detection, is one of the classical malware detection methods [17, 25]. It aims to identify potential ransomware attacks before an actual attack occurs. In this approach, the malware/ransomware binary is analyzed, and code string patterns are extracted from the ransomware binary using tools such as PE-Studio [27] to find possible indications about the presence of the ransomware. The extracted signatures are then compared with known malware signatures from online resources like VirusTotal [29], that provides hints about the presence of any malware in the binary file. However, this method does not always perform well because the signature database might not be updated with the continuously evolving ransomware variants.

2.2.2 Dynamic analysis. In this method the presence of ransomware is identified by monitoring the system changes that occur after a ransomware binary is executed on the system [15, 21, 30]. This approach entails running a ransomware binary in a controlled environment and monitoring the subsequent changes occurring in the system. Commonly observed changes include process creation, high memory usage, file type modifications, high frequency of data read and/or write operations [15], and high entropy value of newly written data. These methods impose significant overhead on the



(a) Overwrite Ratio.



(b) Overwrite Interval.

Figure 2: Offline profiling insights for benign and ransomware applications. Red indicates ransomware, and blue indicates benign applications.

CPU and memory, as they require real-time monitoring of large amounts of data, placing an extra burden on the system resources. Further, these methods are vulnerable to new ransomware variants that are increasingly trying to mimic the behavior of benign applications, making them less effective against evolving threats. Methods like [7, 9, 14] have been developed to safeguard against ransomware threats using machine learning at the host system.

2.2.3 *SSD level solutions.*

Many SSD-level solutions have been proposed in the literature to safeguard data stored on SSDs against ransomware attacks. These solutions leverage internal characteristics of flash memory, such as out-of-place updates and erase-before-write properties. The out-of-place update feature requires the SSD controller to write each overwritten page to a new location, while the old page remains on the flash memory in an invalid state until garbage collection (GC) is invoked. This out-of-place update feature can provide a potential opportunity to recover data if it has been encrypted by a ransomware attack.

FlashGuard [13] aims to defend against ransomware by implementing a backup strategy for each page. It introduces an additional page state, termed the backup state, alongside the valid and invalid states of a flash memory page. Pages suspected of being encrypted due to a Write-After-Read (WAR) pattern are marked as backup pages. Unlike regular pages, these backup pages are not immediately subjected to GC, allowing them to be retained. However, while FlashGuard effectively preserves SSD data, it has significant overhead in terms of space consumption, limiting its practicality as a solution.

SSDInsider [2] detects ransomware by analyzing the header of each I/O request, which includes details like the LBA, read/write operation, and length of data. It monitors metrics such as the number of erases, the ratio of a number of erases to the number of writes, length of erased data over a time period to differentiate between I/O sequences originating from benign or ransomware applications. Additionally, SSDInsider++ [3] enhances this by incorporating background checks on the entropy of written data, alongside the features used in SSDInsider [2], for more robust ransomware detection. SSDInsider++ is prone to false positives and false negatives. SSDInsider++ handles the data recovery using a recovery queue that stores previous address mappings of encrypted pages, facilitating rollback from the ransomware state to the pre-ransomware state.

RansomBlocker [20] defends against ransomware attacks using hardware accelerators to compute the entropy of incoming write requests. Meanwhile, methods like [18, 19] detect ransomware through content-aware analysis, calculating the Ransomware Attack Risk Indicator (RARI) based on the entropy of each page after it is written to flash memory. RansomBlocker utilizes convolutional neural networks (CNNs) to effectively detect and alert about ransomware attacks. In this work we are focusing only on cryptographic ransomware types.

3 Insights and Motivation

3.1 Features of Ransomware and Benign Applications

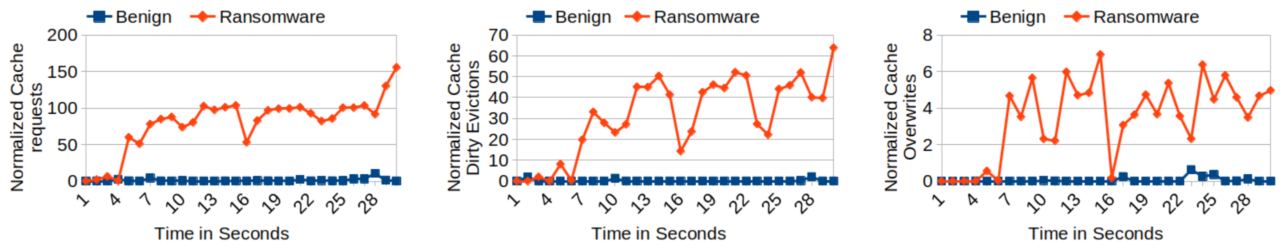
We conduct offline profiling of I/O requests from trace files of both benign and ransomware applications. Detailed information of these applications and experimental setup is provided in Section 5. Through a thorough analysis of these raw trace files, we identify several key features that differentiate ransomware application execution from benign application behavior.

3.1.1 *Overwrite Ratio.*

The overwrite ratio is the proportion of write operations performed on pages that have previously been read by the host system, relative to the total number of read requests. Ransomware operates by reading pages from the flash memory, encrypting the data with a cryptographic algorithm, and writing the encrypted pages back to the flash memory. As illustrated in Figure 2a, ransomware applications tend to exhibit a high percentage of overwrites, while benign applications show a comparatively low percentage of such overwrites. This pattern, where ransomware frequently reads and then writes back to the same address can serve as a one of the key indicators to distinguish ransomware applications from benign applications.

3.1.2 *Overwrite Interval.*

The overwrite interval is the time between reading a page and writing it back. From the observed average overwrite times depicted in Figure 2b, it can be concluded that



(a) Normalized I/O Requests. (b) Normalized Cache Dirty Evictions. (c) Normalized Cache Overwrites.

Figure 3: DRAM cache behavior for Wannacry ransomware and data-compression (benign) application. The Y-axis presents the numbers in the multiples of 1K.

different ransomware samples exhibit varying overwrite behaviors. Some ransomware variants demonstrate rapid overwrites in short bursts of less than 10 seconds, indicating aggressive encryption tactics. In contrast, other ransomware variants exhibit delayed overwriting, taking more than 20 seconds to perform overwrites, resembling the behavior of benign applications.

This variability highlights the difficulty of detecting ransomware based solely on overwrites, as the pattern can differ significantly among different ransomware implementations. Therefore, effective ransomware detection mechanisms must consider a broader range of behavioral indicators and patterns beyond the overwrites to accurately distinguish ransomware from benign applications.

3.2 DRAM Cache for Ransomware Detection and Recovery

Flash memory has asymmetric read and write operations, with write operations typically taking about ten times longer than read operations [16],[4]. Additionally, it faces challenges concerning its lifespan and endurance [12], [22]. To effectively handle these inherent issues of flash, an onboard DRAM cache is used to provide uniform read and write speeds [26]. The DRAM cache intercepts write requests destined for flash memory and serves frequently accessed reads directly from itself, reducing the I/O activity of the flash memory, thereby extending SSD lifespan and performance.

When a ransomware application runs, it attempts to encrypt multiple files rapidly, leading to distinct patterns of I/O requests recorded in the DRAM cache. These patterns result in sudden spikes of I/O request processing per unit time, an increased number of cache overwrites, and a higher number of dirty page evictions. Figure 3a illustrates the normalized value of number of cache requests per second (normalized to 1K) for both benign and ransomware applications. And Figures 3c, and 3b illustrate the normalized values of cache overwrites and dirty page evictions per second, respectively, using the same normalization. Ransomware's write-after-read behavior leads to an increased number of write hits in the DRAM cache. The elevated I/O activity of ransomware triggers more frequent dirty page evictions from the cache.

Such abnormal access patterns and activity spikes provide valuable hints that help distinguish ransomware from that of benign applications. Furthermore, the DRAM cache plays a crucial role in data recovery. Once a ransomware attack is detected, some of the encrypted pages may still reside in the DRAM cache, reducing the need for rollback for those pages as they have not yet been written to the flash memory and the original files are not marked invalid. Thus DRAM cache can help reduce the data recovery time by decreasing the number of encrypted pages written to the flash memory. Hence, we find that the DRAM cache can play a vital role in both detection and data-recovery from ransomware attacks.

4 Design of Cache Assisted Ransomware Detection and Recovery (CARDR)

4.1 Basic Idea

CARDR operates at the SSD controller to safeguard against ransomware attacks. CARDR continuously monitors incoming I/O requests from the host system, extracting features from the requests and the activities in the DRAM cache in order to service these requests. Every second, it samples these features and passes them to a machine-learning model, such as a decision tree, in the form of a feature vector. The decision tree is trained to detect ransomware patterns. If it indicates the presence of ransomware for *k*-consecutive time slices, CARDR alerts the host system about the threat. In response, the incoming I/O requests to the SSD are halted, and the SSD controller initiates the data recovery process. CARDR uses a recovery queue that stores LBA-to-PPA (physical page address) translations for encrypted pages. This queue retains mapping entries of LBAs, linking them to their previous PPAs. During the data recovery process, mappings in the recovery queue are updated in the SSD's mapping table, effectively restoring the SSD to its pre-ransomware attack state.

4.2 Working of CARDR

The flowchart depicted in Figure 4 outlines the CARDR's process of ransomware detection. Initially, ① incoming requests from the host system are directed to the SSD device. Subsequently, ② feature extraction process starts by sampling incoming I/O requests over a specified time period (or slice), extracting key features from both the I/O requests and the ongoing DRAM cache activities. At the end of each time slice, these extracted features are consolidated into a feature vector, which is then fed into a decision tree. The decision tree evaluates each feature to predict the presence of ransomware ③ or benign activity ④. To ensure robust detection,

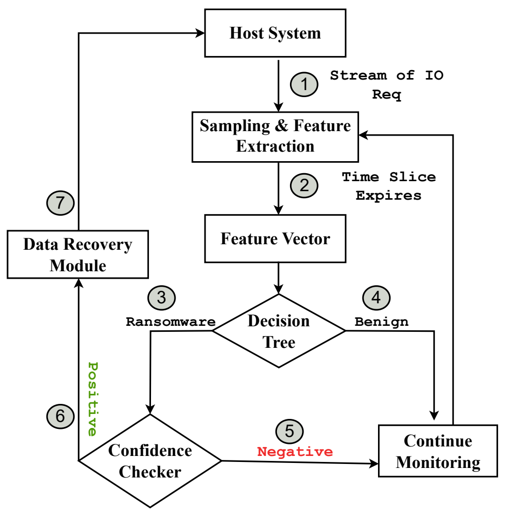


Figure 4: Flowchart explaining the process of ransomware detection in CARDR.

CARDR incorporates a confidence checker. When the confidence checker indicates possibility of benign activity (5), no further action is taken. However, (6) if the decision tree indicates the possibility of ransomware over *k*-consecutive time slices, CARDR flags the presence of ransomware, prompting the SSD controller to halt incoming I/O requests, (7) alerts the host system about the ransomware and initiate data recovery procedures.

4.2.1 *Feature Extraction Process.*

The feature extraction process in CARDR involves sampling the incoming I/O requests directed to the SSD over a specified time, similar to the approach used in SSDInsider++. At the end of each one-second time slice, CARDR extracts the values of predefined features from the incoming I/O requests and DRAM cache activities. These feature values are then compiled into a feature vector, and the feature variables are reset, preparing for the next time slice. This continuous sampling and feature extraction ensures that, from time-to-time the feature vector accurately represents the ongoing I/O activity, providing the necessary context for the machine learning model to detect potential ransomware attacks.

4.2.2 *Features extracted for Ransomware detection.*

CARDR extracts features from incoming I/O requests and the activities triggered by these I/O requests at the DRAM cache. Features derived from the cache are known as cache-driven features, while those extracted directly from the I/O requests without any cache involvement are non-cache-driven features. The extracted features are further categorized into two classes: primary features and secondary features. The primary features capture the system's behavior in real-time, within the current time slice of 1 second. To provide a broader context, we also track the system's historical behavior using a n-1 second time window, organized as a circular list, representing the system's activities over the past n-1 seconds. This dual-class feature extraction approach, illustrated in Figure 5, enables us to capture both current and historical behaviors, offering a comprehensive view of the system's activities. This thorough analysis allows for a more reliable ransomware detection. We use a value of 10 for n.

(i) Cache-driven features: There are four important cache-driven features that CARDR uses for ransomware detection.

1. Cache Overwrites (`CO`): CO is collected for each time slice and is governed by the write-after-read property of ransomware. When data is requested from the host system, it is retrieved from the flash memory in case of a cache miss, and then forwarded to the host system, and a copy of this data is maintained (until the time of eviction) in the DRAM cache of the SSD to exploit temporal or spatial locality. In case there is another read or write request from the host for this data, it will result in a DRAM cache hit. CO represents the count of cache write hits per time slice, indicating the number of pages that have been overwritten in the cache. Ransomware samples typically exhibit a quick write-after-read pattern, resulting in a higher value of cache overwrites compared to benign applications. Tracking this feature provides a useful indicator of the presence of ransomware.

2. Cumulative Cache Overwrites (`CCO`): In order to avoid detection due to sudden spikes in the write activity, some ransomware samples may spread the writes resulting in cache overwrites over an extended time slices. These writes are potentially missed if the observation window is small. To account for such delayed overwrite patterns of ransomware, we use the cumulative cache overwrite feature. This feature provides the total number of cache write hits over the last n-1 time slices. It is extracted by placing the CO values in a circular queue and returning the sum of the last n-1 COs from the queue.

3. Dirty Evictions (`DE`): As ransomware begins execution, it starts reading and/or writing data in a burst manner. Burst reads cause more pages to be loaded into the cache from the flash memory, while burst writes involve either overwriting existing pages (write-hits) in the cache or writing new pages (write-miss) to it. This burst I/O activity can result in increased eviction of pages from the DRAM cache, possibly leading to a higher count of dirty page evictions. This pattern is particularly noticeable in ransomware samples. To monitor such activity, we use Dirty Evictions (DE), that is a counter to keep track of the number of dirty page evictions from the DRAM cache within a time slice, providing a key indicator of ransomware activity.

4. Cumulative Dirty Evictions (`CDE`): CDE tracks the total number of dirty pages evicted from the DRAM cache over a time window. This metric provides a cumulative measure of how frequently pages have been evicted from the cache due to burst overwrite characteristics of ransomware. Unlike Dirty Evictions (DE), that captures evictions per time slice, CDE offers a broader perspective by accumulating these evictions over the last n-1 time slices. This cumulative view helps in understanding the sustained impact of ransomware behavior on the DRAM cache eviction.

(ii) Non-Cache Driven features: There are three important non-cache driven features that CARDR uses for ransomware detection.

1. Erases or Overwrites (`OV`): The erase feature `OV` is a critical metric in detecting Class A ransomware. This ransomware family exhibits a distinctive behavior in that it writes to the same logical

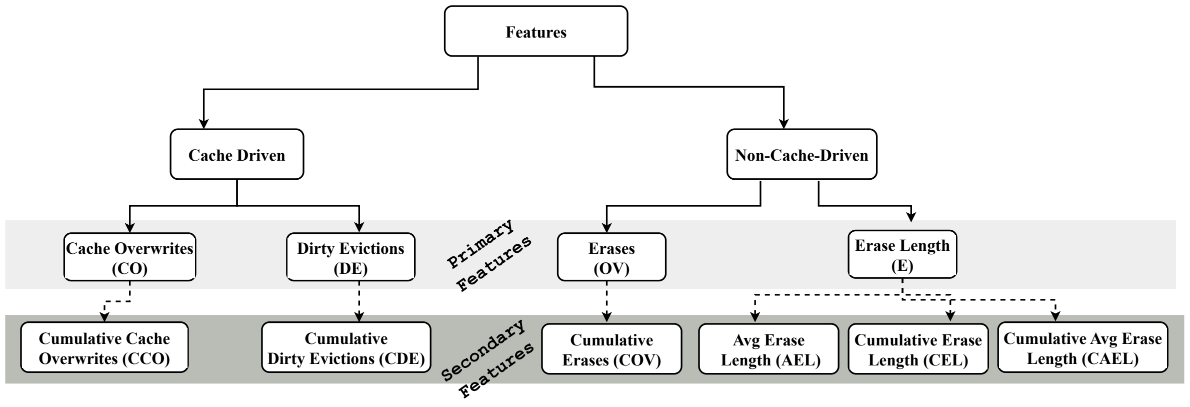


Figure 5: Extracted cache-driven and non-cache driven features used in CARDR.

address from which data was previously read. OV is computed by analyzing the I/O requests contributing to these erase operations occurring within each second. Unlike cache write hits, which focus on tracking overwrites only when the page is present in the cache, OV monitors all I/O operations which are previously read and subsequently written back to the same LBA. OV ensures that even if pages are evicted from the cache, the erase operations are still accurately counted.

2. Cumulative Overwrite (COV): This feature captures the persistent overwrite patterns observed over a defined time window. This feature provides historical context on the frequency of erases occurring as ransomware executes. It sums up the number of overwrite operations performed within the last n-1 time slices.

3. Erase Length (E): This feature measures the total size of data that has been overwritten during a time slice. Ransomware applications tend to perform more erase operations compared to benign applications, resulting in higher erase data lengths. Additional features extracted from the erase length include:

- **Average Erase Length (AEL)**: This provides the average size of erase operations during the current time slice. AEL is calculated by dividing E by OV.
- **Cumulative Erase Length (CEL)**: This feature represents the total erase length over the last n-1 time slices. It captures the cumulative size of data overwritten, providing insight into the long-term behavior of the application.
- **Cumulative Average Erase Length (CAEL)**: This feature indicates the average erase length over the past n-1 time slices, reflecting the sustained overwrite behavior of the application over a longer period.

4.3 ML Model in CARDR

Given the limited computational resources of SSD controllers, selecting an efficient and lightweight machine-learning model for ransomware detection is crucial. We chose a supervised learning algorithm, decision tree for its simplicity and low resource requirements, making it well-suited for this constrained environment. The choice of a decision tree is carefully balanced between maintaining prediction efficiency and adapting to the limited resources available at the SSD controller. Unlike more complex models like Deep Neural Networks (DNNs) and Recurrent Neural Networks (RNNs), which demand significant computational power, the decision tree offers a minimal computational footprint. It works by dividing the dataset into branches based on key features, creating an "if-then" logic path for decisions. This straightforward structure integrates easily with existing simulator code, as the tree's branching decisions translate into simple conditional statements with predictions as return values.

Figure 6 depicts the decision tree CARDR uses for detecting ransomware threats, the features are displayed at the internal and root nodes of the tree, while the leaf nodes represent the predictions made by the decision tree. For instance, if the value of COV exceeds 1000 and the CAEL value is greater than 570, the decision tree classifies the application as ransomware. Conversely, if the CAEL is below 570, the application is classified as benign. Similarly, if the COV is less than 1000 but CDE exceeds 17,000, the application is identified as ransomware. The decision tree continues to apply these criteria across other scenarios. Algorithm 1 provides a pseudo-code of CARDR's detection algorithm, .

4.4 Data Recovery in CARDR

When CARDR's detection module alerts about a ransomware attack, the SSD controller invokes the data recovery module. CARDR's data recovery module is designed to roll back changes caused by ransomware and restore the SSD to its pre-ransomware state. This module operates within the Flash Translation Layer (FTL) of the SSD. CARDR leverages the out-of-place update feature of SSD to facilitate data recovery. When an updated page is written to the flash memory, it is written to a new flash memory page, while the old copy of the same page remains on the flash memory in an invalid state until GC reclaims the pages. The address mapping in the mapping table is updated to point to the new PPA. Until the invalid page is erased by GC, the old page can always be rolled back by updating the entry in the mapping table to point to the old page. This mechanism allows the system to revert to the old copies of pages if ransomware activity is detected.

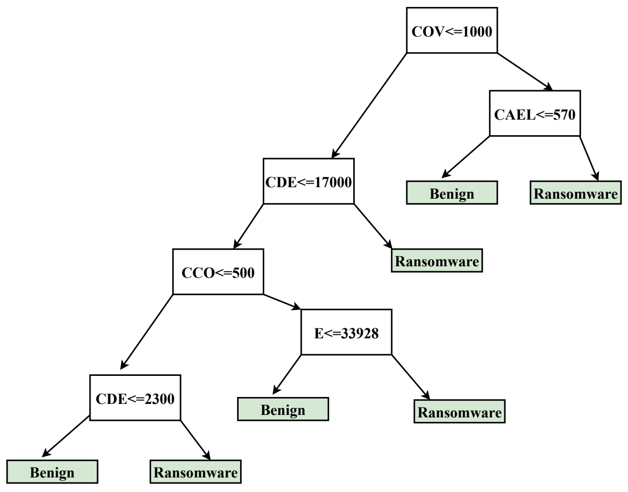


Figure 6: Decision tree used in CARDR.

Algorithm 1 Ransomware Detection

```
1:  function DetectRansomware(n)
2:      counter ← 0
3:      for each I/O request do
4:          featureVector ← ExtractFeatures()
5:          if time slice expires then
6:              MLOutput ← predict(featureVector)
7:              if MLOutput == Ransomware then
8:                  counter ← counter + 1
9:                  if counter == $k$ then
10:                     AlertHost("$R$")
11:                     DataRecovery
12:                     break
13:                 end if
14:             else
15:                 counter ← 0
16:             end if
17:         end if
18:     end for
19: end function
```

CARDR utilizes a recovery queue similar to that of SSDInsider++ that contains the mapping entries of pages encrypted by ransomware. Each entry in the recovery queue consists of a pair of logical-to-physical address mappings. Essentially, it holds mapping entries of LPA's pointing to the old PPA's of encrypted pages, the original physical pages from which data was read. This allows CARDR to identify the old flash pages and restore them back.

CARDR identifies pages that are first read by the host system and subsequently written back to the flash memory by leveraging the address mapping table. CARDR modifies the mapping table structure by introducing an additional bit called ReadBit (RB) with each entry of the mapping table. RB acts as a flag to categorize pages that have undergone a write-after-read sequence, effectively classifying them as encrypted pages.

When the host system issues a page read request, the mapping table provides the address translation for the corresponding LBA. After obtaining the PPA from the mapping table, a page is read from the flash memory. Once the read operation from the flash memory is completed, RB associated with the corresponding LBA in the mapping table is set. This RB if set indicates that the page has been successfully read from the flash memory.

During a page write request to the flash memory, if the mapping table shows the RB set for the corresponding LBA, it signifies that the page was previously read and is now being written back. CARDR identifies such pages as encrypted pages. Before updating the PPA in the mapping table, CARDR adds the LBA and its current PPA to the recovery queue. Subsequently, it updates the mapping table so that the LBA points to the new PPA while retaining the old PPA in the recovery queue. The RB for the LBA is reset, indicating that the page is no longer a write-after-read page.

Upon detecting a ransomware threat, the recovery module is activated to perform the data recovery by scanning through the entries in the recovery queue, taking each entry from the queue, and updating them in the mapping table. CARDR restores the SSD to its pre-ransomware state by rolling back the encrypted pages.

Data recovery process in CARDR. In Figure 7, the detailed flow of the data recovery module in CARDR is depicted. (1) Incoming I/O requests from the host system are placed in the DRAM cache. (2) If it is a read request and is a cache miss, the requested page is to be fetched from the flash memory. (3) The address mapping is obtained from the address mapping table. (4) Upon obtaining the PPA from the address mapping table, the corresponding page is read from the flash memory and is placed into the DRAM cache. (5) RB bit of the corresponding LBA is set to signify that this page has been read from the flash memory. (6) When a dirty page is evicted from the DRAM cache, it has to be written back to the flash memory on the address provided by the mapping table. If the mapping entry exists in the mapping table, (7) the RB bit is checked. If it is set, it means that the corresponding page was previously read from flash memory and is now being written back. CARDR classifies this write as an encrypted write and (8) copies the LBA to PPA mapping from the mapping table into the recovery queue. (9) The mapping entry in the mapping table is modified by pointing the LBA to the new PPA. (10) page is then written to the flash memory pointed by the new PPA, and at the same time, (11) the RB bit is reset, signifying that the corresponding page has no longer been read from the flash memory. (12) Upon invocation of the data recovery process, CARDR scans through the recovery queue, and (13) updates the mapping entries in the mapping table by corresponding entries from the recovery queue, which effectively rolls back the SSD state to its pre-ransomware state.

5 Experimental Setup

To evaluate the efficiency of CARDR, we implemented it using the state-of-the-art SSD simulator, SimpleSSD-Standalone [10]. This simulator offers a detailed model of an SSD system, encompassing the cache subsystem, FTL, NAND flash array, and SSD interface,

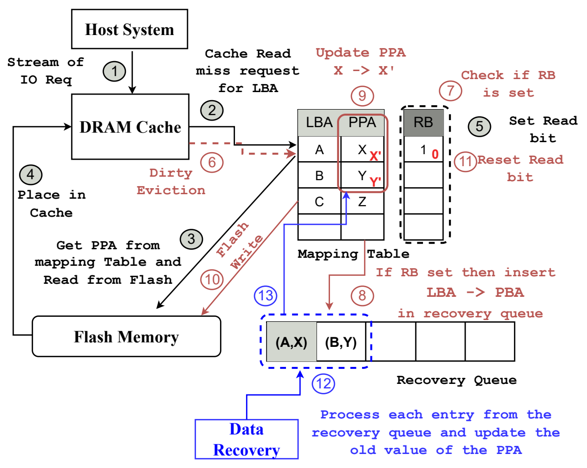


Figure 7: Data recovery process in CARDR.

Table 1: Simulation Configurations.

Configuration Parameter	Value
Simulator	SimpleSSD2.0
SSD Size	128 GB
Channels	2
Chips	4
Die	8
Plane	64
Block	128
Pages	64
Page Size	4KB
DRAM Cache	128MB
DRAM Cache Page Size	4KB
DRAM Eviction Policy	LRU
Baseline Victim Block Selection	Greedy Algorithm

among other components. The specific configurations of the simulator used in our experiments are detailed in Table 1.

We compared the performance of CARDR with SSDInsider++ [3], focusing on two key metrics: ransomware detection time and data recovery overhead. Our tests measured how quickly each method detected ransomware and the associated overhead for data recovery. The results of these comparisons highlight the effectiveness of CARDR in both detecting ransomware promptly and minimizing recovery time.

For the implementation of SSDInsider++ and CARDR on the SimpleSSD-Standalone simulator, we modified the Host Interface Layer (HIL) code of the simipleSSD2.0 to incorporate the logic of the ransomware defense methods. SSDInsider++ also observes incoming I/O requests and extracts features from these requests. Both SSDInsider++ and CARDR utilize a decision tree model trained on a dataset containing known ransomware and benign applications.

We used Python scripts to train the decision tree for both SSDInsider++ and CARDR. The training involved various combinations of training and testing datasets, as detailed in Table 3. Each combination produced different detection accuracies. We selected the decision tree that achieved the highest accuracy and incorporated its c_code into the simulator in the form of "if-then" statements.

This experimental setup allowed us to rigorously test and compare the performance of CARDR and SSDInsider++, ensuring reliable and early detection of ransomware while maintaining high detection accuracy.

5.1 Ransomware and Benign Application Dataset

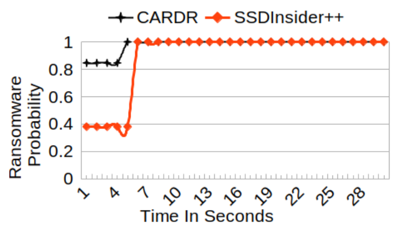
Table 2: Workload Details.

Application	Dataset	IO's(M)	Write Ratio
Ransomware			
TeslaCrypt (R1)	Ransap	3.32	49.77
Cerber (R2)	Ransap	3.13	47.52
WannaCry (R3)	Ransap	8.92	61.26
GandCrab (R4)	Ransap	7.17	50.41
Ryuk (R5)	Ransap	1.91	49.40
Sodinokibi (R6)	Ransap	1.14	38.06
Darkside (R7)	Ransap	0.52	58.24
Benign			
AESCrypt (B1)	Ransap	1.21	46.13
Zip (B2)	in-house	0.063	23.44
Data Compression (B3)	in-house	0.18	27.70
Download (B4)	in-house	0.035	43.65
NormalExecution (B5)	in-house	0.67	31.81
Fragmentation (B6)	in-house	0.269	44.45

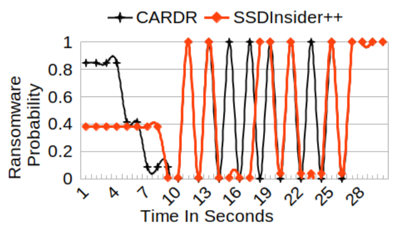
To validate CARDR, we utilized the Ransap [11] dataset, a publicly available collection containing time series storage access patterns of various well-known ransomware and benign applications provided in the form of trace files. Each trace file contains I/O requests, where each request is accompanied by its arrival time information. The I/O requests in the trace file are sorted according to their arrival times and have been collected via a hypervisor. Ransap is being extensively employed for ransomware research studies, enabling the analysis and comprehension of behaviors exhibited by various ransomware and benign applications.

To enrich our dataset, we collected the I/O access patterns of several benign applications. These include monitoring tasks such as zip file extraction, memory fragmentation, and data compression, and the I/O sequences from Internet data downloading. Each application was closely observed to capture its unique I/O access patterns. For the I/O collection, we utilized the DiskMon utility from Sysinternals, operating on the Microsoft Windows operating system [6]. This utility allowed us to monitor and log the I/O activities performed by various applications. Table 2 provides comprehensive details of I/O traces of various ransomware and benign applications. Traces obtained from the Ransap repository are distinguished with the "Ransap" label, while traces collected using the Diskmon utility are labeled as "In-house". This categorization allows for a clear differentiation between the sources of the IO traces, facilitating a thorough understanding of the dataset's composition and origin.

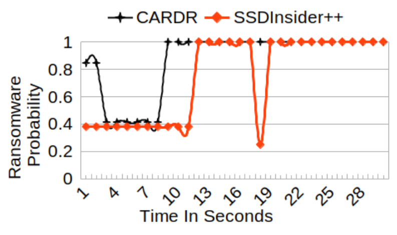
Ransomware applications



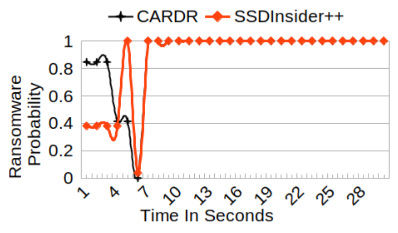
(a) GandCrab



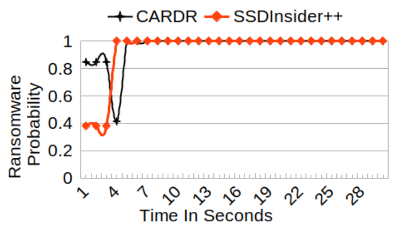
(b) Cerber



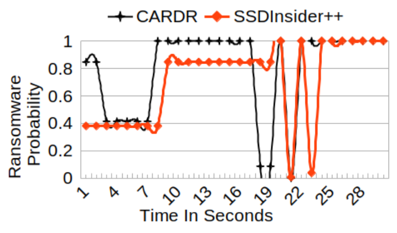
(c) Darkside



(d) Wannacry

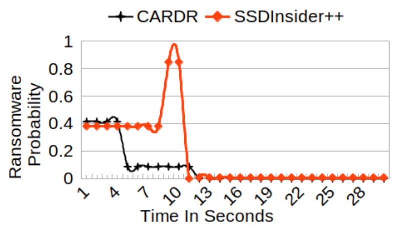


(e) TeslaCrypt

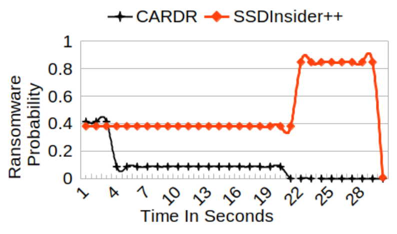


(f) Sodinokibi

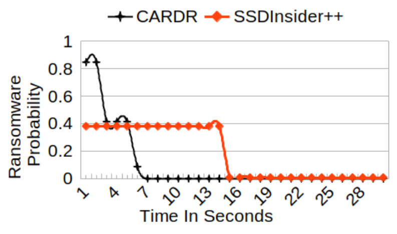
Benign applications



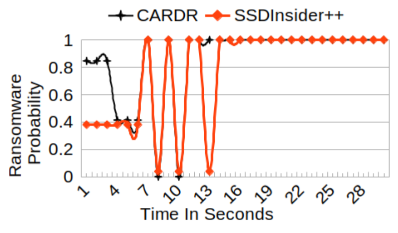
(g) Download



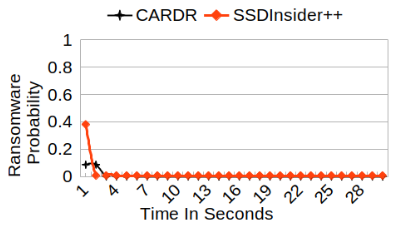
(h) Normal Execution



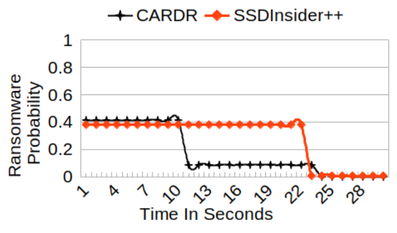
(i) Fragmentation



(j) AesCrypt



(k) Compression



(l) Zip

Figure 8: Behavior exhibited by ransomware and benign applications across different time intervals in CARDR and SSDInsider++.

5.2 Evaluation of Detection Algorithm

To evaluate the effectiveness of the CARDR's detection module, we focused on detection time as the primary metric. Detection time within the context of ransomware refers to how quickly the system identifies ransomware activity once it begins execution. Comparing CARDR with SSDInsider++, we observed that CARDR consistently detected ransomware more swiftly. Figure 9 illustrates this, showing CARDR taking less time for ransomware detection as compared to SSDInsider++. The features used by SSDInsider++ rely solely on erasures, whereas CARDR utilizes additional cache-driven features. This enables CARDR to achieve a lower detection latency of ~11 seconds, compared to SSDInsider++, which takes ~14 seconds for ransomware samples like `Darkside`, `Gandcrab`,

Table 3: Five combinations of test and training sets along with accuracy.

#	Training Set	Testing Set	CARDR Accuracy	SSDInsider++ Accuracy
1	R1, B6, R7, R4, B1, R5, B5	R6, B2, R2, B4, B3, R3	0.9896	0.9842
2	R4, R7, B3, R3, R5, B1, B5	R1, R6, B2, B6, B4, R2	0.9125	0.9865
3	R6, B2, R4, B4, R7, R3, B5	R1, B6, B3, R5, B1, R2	0.8925	0.9784
4	R1, B2, R4, B4, R5, B1, R2	R6, B6, R7, B3, B5, R3	0.6975	0.7686
5	B2, R4, R7, B3, R3, B1, R2	R1, R6, B6, B4, R5, B5	0.6332	0.8968

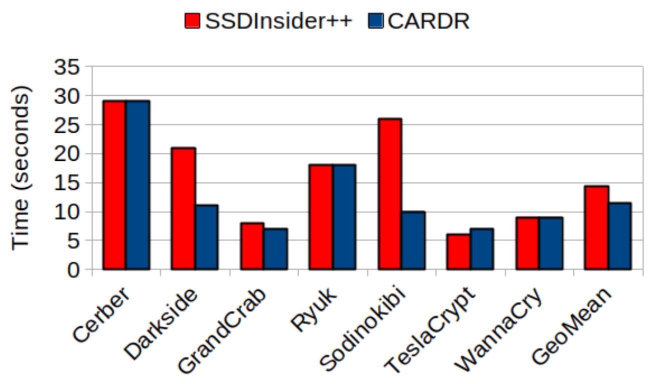


Figure 9: Ransomware detection latency.

and Sodinokibi. In contrast, for other ransomware samples such as WannaCry, Ryuk, and Cerber, CARDR and SSDInsider++ have similar detection latencies. To further understand the detection module's timing latency, we plotted several graphs showing the probability of ransomware at various time intervals during execution for both benign and ransomware applications. Individual timing graphs for each ransomware and benign application show the differences clearly.

In Figure 8, the first section of the graphs represents the ransomware applications. We observe that the ransomware probability reaches its highest level for samples like TeslaCrypt, WannaCry, and GandCrab within 10 seconds due to the burst I/O and high erasure count exhibited by these ransomware applications. The Cerber ransomware, which employs delayed overwriting is detected around the 29^{th} second by both SSDInsider++ and CARDR. In contrast, for ransomware applications like Darkside, Gandcrab and Sodinokibi, cache-driven features help CARDR detect them much faster compared to SSDInsider++. This is evident in the Figures 8c, 8a, and 8f, where the ransomware probability for CARDR rapidly converges to the highest value for Darkside, GrandCrab and Sodinokibi samples, outperforming SSDInsider++.

Similarly, for benign applications, the plotted graphs for applications like Fragmentation, Data-Download, Normal Execution, Data Compression, and Zip, we notice that CARDR quickly shows the probability of ransomware approaching to zero more swiftly than SSDInsider++. However, for applications like AesScript that exhibit behavior similar to ransomware, the timing diagrams indicate that both SSDInsider++ and CARDR classify this application as ransomware. This classification raises the possibility of false positives in detection.

5.2.1 *Detection threshold.* To minimize the count of false positives and false negatives, we implement a threshold counter. This threshold counter controls the maximum number of consecutive outputs from the machine learning model required to reliably classify an application as ransomware or a benign application. This approach ensures more accurate predictions by allowing the model to confirm its decision over multiple time slices before making a final classification.

Table 4 consists of five columns, each representing different aspects of our evaluation metrics. The first column lists the threshold values used for detection. The second column is labeled ransomware detected as ransomware (RR), indicating the number of applications that were originally ransomware and were correctly identified as ransomware by the detection module. The third column, ransomware detected as benign (RB), represents the count of applications that were originally ransomware but were incorrectly classified as benign. Similarly, the fourth column, benign detected as ransomware (BR), shows the number of benign applications that were mistakenly detected as ransomware. The fifth and final column, benign detected as benign (BB), reflects the count of benign applications that were correctly identified as benign.

Table 4: Impact Of different threshold values on detection accuracy.

Threshold	RR	RB	BR	BB
1	7	0	3	3
2	7	0	2	4
3	7	0	1	5
4	7	0	1	5

From the data presented, it is evident that starting with a threshold value of 3, the detection module consistently identifies all ransomware applications correctly (RR). However, there are still some false positives present, as indicated by the BR column, where one benign application out of six is incorrectly classified as ransomware. This observation highlights the trade-off between improving true positive rates and minimizing false positives, emphasizing the importance of selecting an optimal threshold value to balance detection accuracy and reliability. Additionally, a higher threshold value results in a larger recovery queue, which introduces another trade-off between accuracy and recovery overhead that must be carefully considered.

5.3 Recovery

To evaluate the effectiveness of the CARDR's recovery module, we focused on the number of pages needed to rollback upon a ransomware presence alert generated by the CARDR's detection

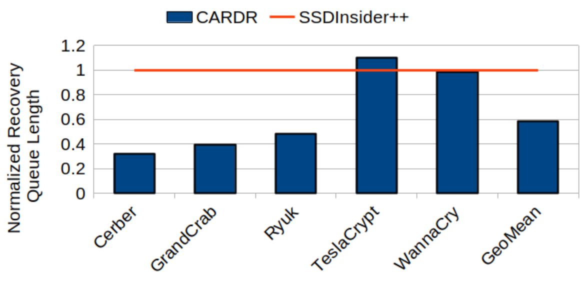


Figure 10: Normalized recovery queue length, normalized by recovery queue length of SSDInsider++.

module. We compared the length of the recovery queue of CARDR with SSDInsider++. Figure 10 presents the recovery queue length of CARDR normalized by the recovery queue length of SSDInsider++. CARDR reduces the recovery queue length by an average of 42% compared to SSDInsider++. This reduction in the length of the recovery queue is due to the DRAM cache holding the encrypted pages, preventing them from being written to the flash memory. We have not shown the bars for ransomware samples like `Darkside` and `Sodinokibi` in Figure 10. For these ransomware samples, the size of the recovery queue is almost zero. The ransomware signal is generated when all the encrypted pages are present in the DRAM cache. For such ransomware, CARDR achieves a 100% reduction in the recovery queue, meaning that no rollback is necessary. To combat the ransomware attack, it is sufficient to just flush the contents of the DRAM cache. For `TeslaCrypt`, CARDR generates a slightly longer recovery queue compared to SSDInsider++. This is due to the burst I/O behavior of `TeslaCrypt` and CARDR detecting `TeslaCrypt` one second later than SSDInsider++.

6 Conclusion

Ransomware attacks are rapidly increasing. Solutions like SSDInsider++, which operates at the SSD controller, represent a promising direction in combating these threats. However, since modern storage systems such as SSDs utilize DRAM caches for performance and lifespan enhancement, it is crucial that machine learning models for ransomware detection incorporate features from DRAM caches for early detection. CARDR leverages these DRAM cache-driven features for the early detection of ransomware and reduces the size of the recovery queue, thereby minimizing rollback overhead during data recovery. Looking ahead, CARDR's effectiveness can be further enhanced by training it with sophisticated neural network models, provided there is sufficient budget for deploying these models at the SSD controller level.

7 Acknowledgement

This work was supported in part by 2022 Qualcomm Faculty Award. The authors are grateful for the insightful feedback and comments from our anonymous reviewers. We would also like to express our heartfelt thanks to the members of the Computer Architecture - Special Interest Group (CA-SIG) Lab at IIT Ropar for their valuable discussions and constructive feedback.

References

[1] Appknox. 2019. 20 Cybersecurity Statistics That Matter in 2019. https://www.appknox.com/blog/cybersecurity-statistics-2019 Accessed: 2024-05-25.
[2] SungHa Baek, Youngdon Jung, Aziz Mohaisen, Sungjin Lee, and DaeHun Nyang. 2018. SSD-insider: Internal defense of solid-state drive against ransomware with perfect data recovery. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 875–884.
[3] Sungha Baek, Youngdon Jung, David Mohaisen, Sungjin Lee, and Daehun Nyang. 2020. SSD-assisted ransomware detection and data recovery techniques. *IEEE Trans. Comput.* 70, 10 (2020), 1762–1776.
[4] Feng Chen, David A Koufaty, and Xiaodong Zhang. 2009. Understanding intrinsic characteristics and system implications of flash memory based solid state drives. *ACM SIGMETRICS Performance Evaluation Review* 37, 1 (2009), 181–192.
[5] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. 2016. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd annual conference on computer security applications*. 336–347.
[6] Microsoft Corporation. [n. d.]. DiskMon. https://learn.microsoft.com/en-us/sysinternals/downloads/diskmon. Accessed: 2024-06-01.
[7] Damien Warren Fernando and Nikos Komninos. 2024. FeSAD ransomware detection framework with machine learning using adaption to concept drift. *Computers Security* 137 (2024), 103629.
[8] Recorded Future. 2024. Types of Ransomware. https://www.recordedfuture.com/threat-intelligence-101/cyber-threats/types-of-ransomware. Accessed: 2024-08-31.
[9] Mohana Gopinath and Sibi Chakkaravarthy Sethuraman. 2023. A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review* 47 (2023), 100529.
[10] Donghyun Gouk, Miryeong Kwon, Jie Zhang, Sungjoon Koh, Wonil Choi, Nam Sung Kim, Mahmut Kandemir, and Myoungsoo Jung. 2018. Amber: Enabling precise full-system simulation with detailed modeling of all SSD resources. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 469–481.
[11] Manabu Hirano, Ryo Hodota, and Ryotaro Kobayashi. 2022. RanSAP: An open dataset of ransomware storage access patterns for training machine learning models. *Forensic Science International: Digital Investigation* 40 (2022), 301314.
[12] Yang Hu, Hong Jiang, Dan Feng, Lei Tian, Hao Luo, and Chao Ren. 2012. Exploring and exploiting the multilevel parallelism inside SSDs for improved performance and endurance. *IEEE Trans. Comput.* 62, 6 (2012), 1141–1155.
[13] Jian Huang, Jun Xu, Xinyu Xing, Peng Liu, and Moinuddin K Qureshi. 2017. FlashGuard: Leveraging intrinsic flash properties to defend against encryption ransomware. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2231–2244.
[14] Jamil Ispahany, MD Rafiqul Islam, Md Zahidul Islam, and M Arif Khan. 2024. Ransomware detection using machine learning: A review, research limitations and future directions. *IEEE Access* (2024).
[15] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. 2016. {UNVEIL}: A {Large-Scale}, automated approach to detecting ransomware. In *25th USENIX security symposium (USENIX Security 16)*. 757–772.
[16] Bryan S Kim, Jongmoo Choi, and Sang Lyul Min. 2019. Design tradeoffs for {SSD} reliability. In *17th USENIX Conference on File and Storage Technologies (FAST 19)*. 281–294.
[17] Sandeep Kumar and Eugene H Spafford. 1992. A generic virus scanner in C++. (1992).
[18] Donghyun Min, Yungwoo Ko, Ryan Walker, Junghee Lee, and Youngjae Kim. 2021. A content-based ransomware detection and backup solid-state drive for ransomware defense. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 7 (2021), 2038–2051.
[19] Donghyun Min, Donggyu Park, Jinwoo Ahn, Ryan Walker, Junghee Lee, Sungyong Park, and Youngjae Kim. 2018. Amoeba: An autonomous backup and recovery SSD for ransomware attack defense. *IEEE Computer Architecture Letters* 17, 2 (2018), 245–248.
[20] Jisung Park, Youngdon Jung, Jonghoon Won, Minji Kang, Sungjin Lee, and Jihong Kim. 2019. RansomBlocker: A low-overhead ransomware-proof SSD. In *Proceedings of the 56th Annual Design Automation Conference 2019*. 1–6.
[21] Nolen Scaife, Henry Carter, Patrick Traynor, and Kevin RB Butler. 2016. Cryptolock (and drop it): stopping ransomware attacks on user data. In *2016 IEEE 36th international conference on distributed computing systems (ICDCS)*. IEEE, 303–312.
[22] Gokul Soundararajan, Vijayan Prabhakaran, Mahesh Balakrishnan, and Ted Wobber. 2010. Extending SSD Lifetimes with Disk-Based Write Caches.. In *FAST*, Vol. 10. 101–114.
[23] Hui Sun, Shangshang Dai, Jianzhong Huang, Yinliang Yue, and Xiao Qin. 2023. DAC: A dynamic active and collaborative cache management scheme for solid state disks. *Journal of Systems Architecture* 140 (2023), 102896.
[24] Sangfor Technologies. 2023. List of Top Ransomware Attacks in 2023. https://www.sangfor.com/blog/cybersecurity/list-of-top-ransomware-

attacks-in-2023 Accessed: 2024-05-25.

[25] Patrick Traynor, Michael Chien, Scott Weaver, Boniface Hicks, and Patrick McDaniel. 2008. Noninvasive methods for host certification. *ACM Transactions on Information and System Security (TISSEC)* 11, 3 (2008), 1–23.

[26] Shivani Tripathy and Manoranjan Satpathy. 2022. SSD internal cache management policies: A survey. *Journal of Systems Architecture* 122 (2022), 102334.

[27] Varonis. 2024. PeStudio: A Comprehensive Tool for Analyzing Windows Executables. https://www.varonis.com/blog/pestudio

[28] Cybersecurity Ventures. 2019. *Ransomware Attack Every 14 Seconds*. https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/ URL: https://cybersecurityventures.com/global-ransomware-damage-costs-predicted-to-reach-20-billion-usd-by-2021/.

[29] VirusTotal. 2024. VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/gui/home/upload Accessed: 2024-05-25.

[30] Dongpeng Xu, Jiang Ming, and Dinghao Wu. 2017. Cryptographic function detection in obfuscated binaries via bit-precise symbolic loop mapping. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 921–937.

[31] Lingchen Zhang, Sachin Shetty, Peng Liu, and Jiwu Jing. 2014. Rootkitdet: Practical end-to-end defense against kernel rootkits in a cloud environment. In *Computer Security-ESORICS 2014: 19th European Symposium on Research in Computer Security, Wroclaw, Poland, September 7-11, 2014. Proceedings, Part II 19*. Springer, 475–493.

[32] Leah Zhang-Kennedy, Hala Assal, Jessica Rocheleau, Reham Mohamed, Khadija Baig, and Sonia Chiasson. 2018. The aftermath of a crypto-ransomware attack at a large academic institution. In *27th USENIX Security Symposium (USENIX Security 18)*. 1061–1078.