Preventing Disruption of System Backup against Ransomware Attacks

YIWEI HOU, Tsinghua University, China
LIHUA GUO, Tsinghua University, China
CHIJIN ZHOU, Tsinghua University, China
QUAN ZHANG, Tsinghua University, China
WENHUAN LIU, Tsinghua University, China and UnionTech, China
CHENGNIAN SUN, University of Waterloo, Canada
YU JIANG*, Tsinghua University, China

The ransomware threat to the software ecosystem has grown rapidly in recent years. Despite being well-studied, new ransomware variants continually emerge, designed to evade existing encryption-based detection mechanisms. This paper introduces Remembrall, a new perspective to defend against ransomware by monitoring and preventing system backup disruptions. Focusing on deletion actions of volume shadow copies (VSC) in Windows, Remembrall captures related malicious events and identifies all ransomware traces as a real-time defense tool. To ensure no ransomware is missing, we conduct a comprehensive investigation to classify all potential attack actions that can be used to delete VSCs throughout the application layer, OS layer, and hardware layer. Based on the analysis, Remembrall is designed to retrieve system event information and accurately identify ransomware without false negatives. We evaluate Remembrall on recent ransomware samples. Remembrall achieves 4.31%-87.55% increase in F1-score compared to other state-of-the-art anti-ransomware tools across 60 ransomware families. Remembrall has also detected eight zero-day ransomware samples in the experiment.

CCS Concepts: • **Security and privacy** → **Software and application security**; **Malware and its mitigation**; • **Software and its engineering** → Software functional properties.

Additional Key Words and Phrases: Ransomware, Backup Deletion, Volume Shadow Copy, Malware Mitigation

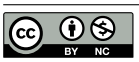
ACM Reference Format:
Yiwei Hou, Lihua Guo, Chijin Zhou, Quan Zhang, Wenhuan Liu, Chengnian Sun, and Yu Jiang. 2025. Preventing Disruption of System Backup against Ransomware Attacks. *Proc. ACM Softw. Eng.* 2, ISSTA, Article ISSTA011 (July 2025), 21 pages. https://doi.org/10.1145/3728880

1 Introduction

Ransomware, a type of malware that unauthorized accesses and encrypts computer files, has rapidly grown in recent decades. It has caused massive financial losses, with billions of dollars extorted from

*Corresponding author

Authors' Contact Information: Yiwei Hou, Tsinghua University, Beijing, China, houyw22@mails.tsinghua.edu.cn; Lihua Guo, Tsinghua University, Beijing, China, glh22@mails.tsinghua.edu.cn; Chijin Zhou, Tsinghua University, Beijing, China, tlock.chijin@gmail.com; Quan Zhang, Tsinghua University, Beijing, China, zhangq20@mails.tsinghua.edu.cn; Wenhuan Liu, Tsinghua University, Beijing, China and UnionTech, Beijing, China, liuwenhuan@uniontech.com; Chengnian Sun, University of Waterloo, Waterloo, Canada, cnsun@uwaterloo.ca; Yu Jiang, Tsinghua University, Beijing, China, jiangyu198964@126.com.



ACM 2994-970X/2025/7-ARTISSTA011
https://doi.org/10.1145/3728880

victims [10] and even reported loss of life [16]. Consequently, implementing defense mechanisms to combat ransomware has attracted huge attention in academia, industry, and government [14, 15].

Efforts to defend against ransomware have often focused on its file encryption behavior, which is the characteristic nature of cryptographic ransomware. As a result, various indicators have been proposed by security researchers to detect the encryption process of ransomware, including file I/O behavior [12, 25, 26, 58], malicious/benign encryption differences [34], encryption key generation [11], and encrypted data buffering [17]. However, both direct and indirect detection methods for these encryption behaviors can be easily bypassed - ransomware has evolved to increase the stealthiness of its encryption behaviors, such as imitating normal program I/O [59] and independently implementing encryption algorithms to avoid using detectable API and libraries [29]. In essence, defending against ransomware based on encryption is difficult as attackers can continually devise a new strategy to evade these indicators.

This paper shifts focus to safeguarding system backups as a sound defense strategy to eliminate the impact of ransomware. Most operating systems, by default, have backup functionalities enabled. For example, Volume Shadow Copy Service (VSS) is a built-in feature in Windows that periodically creates incremental backups of files and system settings, i.e., Volume Shadow Copies (VSC). This allows users to restore their systems to a previous state in case of data loss. Hence, to ensure their ransom demands, ransomware must disrupt these system backups; otherwise, victims can simply recover their data without paying a ransom. Consequently, protecting these system backups becomes a critical measure in preventing ransomware attacks.

However, completely preventing ransomware from disrupting system backups is challenging. First, it is critical to thoroughly identify all potential methods ransomware might use to target backups. Any misidentification could allow attackers to exploit gaps in the defense, leading to false negatives. Therefore, a comprehensive model of malicious backup deletion actions is required. Second, minimizing false positives when detecting malicious actions is equally important. Not all backup deletion actions are malicious; some may result from routine system maintenance or legitimate user activity. Therefore, accurately determining the origin of the deletion action is vital for effective ransomware defense.

We propose Remembrall, which presents a new anti-ransomware perspective that focuses on defending against ransomware by monitoring and preventing system backup disruptions. Focusing on VSC deletion actions, Remembrall captures related malicious events and identifies all ransomware traces as a real-time defending tool. We comprehensively investigate the VSS mechanism and classify all attack actions that one can use to delete VSC backups throughout the application layer, OS layer, and hardware layer. The attack actions include the use of native living-off-the-land binaries, the use of objects in scripting languages, COM object interactions, and direct device access. We will detail them in Section 3.

Based on the above investigation, we design and implement Remembrall to identify VSC deletions and further verify the presence of ransomware. It consists of three modules: *Session Controlling*, *Deletion Monitoring*, and *Process Judging*. The *Session Controlling* module operates quietly in the back-end. It retrieves system events from different layers of the computer system and manages logs in a structured format for future analysis. Next, the *Deletion Monitoring* module filters events related to the four types of VSC deletions. This module ensures all deletion actions are captured. Finally, the *Process Judging* module analyzes these actions, traces the originator of the deletion actions, and determines whether the actions are associated with ransomware or benign programs. Once ransomware is detected, all related processes are terminated forcefully.

With the constructed ransomware dataset, we evaluate the performance of Remembrall against seven other state-of-the-art anti-ransomware tools. Experimental results show that its F1-score of 98.61% is 4.31%-87.55% higher than the other seven peers. The number of files lost under the

protection of Remembrall is 8 on average, lower than 131 of the latest real-time defense tool. Moreover, Remembrall brings an additional 0.07% of CPU usage and 150.27 MiB of memory usage; the runtime overhead is acceptable. Remembrall has also identified eight zero-day ransomware samples in the experiment.

In summary, this paper makes the following contributions:

- We investigate all entrances to disrupt system backup from various computer system layers and report the classification of four types of system backup disruption.
- We design and implement Remembrall, which presents a new anti-ransomware perspective. This novel approach can detect the disruption of system backups and further identify and defend ransomware in real-time. We release the artifact at https://github.com/m1-llie/Remembrall.
- We evaluate Remembrall against the latest ransomware samples. Experimental results show that it outperforms peers by up to 98.61% F1-score across 60 ransomware families, with low detection delay and runtime overhead. It has also identified eight zero-day ransomware samples in the experiment.

2 Background

Volume Shadow Copy Service. Volume Shadow Copy Service (VSS) is designed to provide consistent data protection during Windows file system operations. By capturing snapshots of storage volumes at the block level, VSS creates *volume shadow copies (VSC)* that represent the exact state of the file system at a given point in time. VSS operates as an underlying back-end support. On top of VSS, System Restore focuses on recovering operating system configuration and application state, providing a user interface through the control panel that allows users to select the recovery point, while WinRE operates through a unique recovery environment interface at system startup, providing a comprehensive system failure response solution.

Ransomware Attacks. Ransomware seeks to extort ransom by compromising the accessibility of a victim's data, causing Denial-of-Resources attacks [23]. The lifecycle of ransomware attacks typically contains five steps [5, 8, 41]: Distribution & Infection, Command & Control, Discovery & Movement, Disruption & Leakage, and Extortion. The techniques used in the first two steps are similar to those used for ransomware and other traditional malware, which have been deeply studied by previous attacks [6, 7, 57]. During the rest of the steps in which actual data disruption is carried out, ransomware scans data resources on the compromised host, deletes backup files to inhibit system recovery, encrypts valuable files, etc.

ETW for Event Tracing. ETW (Event Tracing for Windows) is a general-purpose, high-speed tracing facility. It provides a tracing mechanism for events raised by user-mode applications and kernel-mode device drivers through buffering and logging mechanisms implemented in the kernel. Its event logging operations are highly optimized to minimize the impact on system performance, and it has the lowest latency logging capability within Windows. This makes ETW an ideal tool for real-time monitoring and problem diagnosis in production environments [31, 33]. Moreover, ETW Providers are components that generate events that provide detailed information about system or application behavior. They register a specific GUID to define the events they log and their structure, allowing consumers to subscribe to and collect these events.

3 Investigating VSC Deletions

In this section, we investigate all possible approaches to deleting VSCs in Windows systems. This is important for employing a comprehensive defense mechanism because ransomware tends to disrupt VSCs to prevent data recovery. For example, CryptoLocker [54] is one of the most popular

Table 1. List of attack classifications to delete VSCs.

Architecture	Component	Attack Classification	Working Example
Application Layer	User Process	(1) Native Living-off-the-land Binaries	vssadmin.exe
		(2) Objects in Scripting Languages	*Remove_WmiObject* in cmdlets
OS Layer	File System / Driver	(3) COM Object Interactions	*IVssBackupComponents::DeleteSnapshots*
Hardware Layer	Device Controller	(4) Direct Device Access	`IOCTL_VOLSNAP_DELETE_SNAPSHOT`
	Hardware Device	SSD Disruption	/

ransomware families, appearing in September 2013 and spreading via the Gameover Zeus trojan and botnet. CryptoLocker did not remove VSCs in its original iteration, so a common recovery solution for victims was to remove the ransomware from the compromised system and then recover from a known-good VSC. After a few months, the new variant of CryptoLocker appeared, which removed all VSCs in victim systems to prevent recovery.

After analyzing the VSS mechanism, we summarize the possible ways to delete VSCs in Table 1. We classify them into four categories: the use of native living-off-the-land binaries, the use of objects in scripting languages, component object model (COM) object interactions, and direct device access. We provide a detailed explanation of each category below.

(1) Use of Native Living-off-the-land Binaries. Living-of-the-land binaries refer to the tools resident in the system, e.g., vssadmin.exe, wmic shadowcopy, wbadmin.exe, and bcdedit.exe. Among them, the most commonly used one related to VSCs is vssadmin.exe. It provides functionality to list, delete, and resize VSCs. Here is one example of deleting VSCs using vssadmin.exe:

```
vssadmin delete shadows /for=<SpecVolume>
    [/oldest | /all | /shadow=<ShadowID>] [/quiet]
```

Another way to abuse vssadmin.exe is to control the size of the volume shadow copy 'diff area' storage to anything smaller than its currently used space, which can cause automatic snapshot dropping. If the existing snapshots exceed the size of the newly resized 'diff area', the provider will drop snapshots to free up space. To implement such resize deletion, one can set the *maxsize* parameter to any value less than the currently used value. Here is an example of resize deletion:

```
vssadmin resize shadowstorage /for=<SpecVolume>
            /on=<SpecVolume> /maxsize=<newSize>
```

In addition to vssadmin.exe, utilities such as wmic.exe, wbadmin.exe, and bcdedit.exe can also be used to delete VSCs. For example, the wmic shadowcopy utility can be used to delete all VSC backups without any user interaction:

```
wmic shadowcopy delete /nointeractive
```

Therefore, if an attacker wants to delete VSCs, they can use various native Windows utilities with custom parameters to achieve this goal.

A protection mechanism must account for the syntactically diverse nature of command parameters in the user process and be capable of handling obfuscated forms of multiple parameter pairings.

(2) Use of Objects in Scripting Languages. In addition to living-off-the-land binaries, some objects also allow users to manipulate VSCs. For example, one can use one of the following PowerShell scripts to remove all VSCs in the host:

```
1. Get-WmiObject Win32_ShadowCopy | %{$_.Delete}
2. Get-WmiObject -Class Win32 ShadowCopy | ForEachObject {$ .Delete()}
3. Get-WmiObject -Class Win32 ShadowCopy | RemoveWmiObject
4. Get-CimInstance -ClassName Win32 ShadowCopy | Remove-CimInstance
```

Unlike PowerShell, VBScript does not have a built-in object for interacting directly with VSCs, but it can use query language to achieve the same goal. In the example of Listing 1, VBScript retrieves a handle to WMI and then uses that handle to invoke the WQL (WMI Query Language) request, *Select * From Win32_ShadowCopy*. The *For Each* loop will then delete each VSC. Similar scripts can be written in Python or other scripting languages where an interface to WMI is provided.

```
strComputer = "."
Set objWMIService = GetObject("winmgmts:\\" & strComputer & "\root\cimv2")
Set colItems = objWMIService.ExecQuery("Select * From Win32_ShadowCopy")
For Each objItem in colItems
    objItem.Delete_
Next
```

Listing 1. VBScript for the deletion of VSCs.

A protection mechanism must account for unauthorized deletions through object manipulation in various scripting environments, adapting to diverse object interactions and covert script executions.

(3) COM Object Interactions. COM is a standard established by Microsoft Windows to facilitate communication between software components. It provides direct access to the operating system's underlying component objects. Therefore, one can access the Volume Shadow Copy Coordinator (VssCoordinator) to manipulate VSCs.

Users can adopt two approaches to use the VssCoordinator. First, as depicted in Listing 2, the *CoCreateInstance* function provides a handle to the COM interface of the VSS Coordinator. This VssCoordinator interface allows for the secure deletion of all VSCs. Second, as demonstrated in Listing 3, the *IVssBackupComponents* function provides a complete set of VSC manipulations, and one can iteratively delete each one through subsequent calls to *IVssBackupComponents::DeleteSnapshots*.

```
undefined8 init_com_delete_shadow_copies(qword param_1, qword path, qword env){
    uint local_2c; LPVOID local_28; longlong *pIVssCoordinator;
    adjust_SeBackupPrivilege();  // adjust backup privileges
    initialize_COM();  // init COM
    CoCreateInstance(&CLSID_CVssCoordinator, 0x0, CLSCTX_LOCAL_SERVER |
     CLSCTX_REMOTE_SERVER, &IID_IVssCoordinator, &pIVssCoordinator);
    list_shadow_copies(pIVssCoordinator, &local_28, &local_2c);  // list VSCs
    delete_shadow_copies(pIVssCoordinator, local_28, local_2c);  // delete VSCs
    CoUninitialize();  // clear COM
    return 0;
}
```

Listing 2. Using the VSS Coordinator COM object to directly delete VSCs.

```
hResult = (**(*IVssBackupComponents + Query)) (IVssBackupComponents, &vssType, 1, 3, &
     out_IVssEnumObject);
if (hResult == 0 && out_IVssEnumObject != 0x0){
    memset(com_proxy_mem, 0, 0x88);
    // loop through each VSC
    while (VSCs[0] != 0){
        //delete VSCs
        (**(*IVssBackupComponents + DeleteSnapshots)) (IVssBackupComponents, FORCE, 1,
     out_pNondeletedSnapshotID);
        memset(com_proxy_mem, 0, 0x88);
        //prepare for processing next VSC
        VSCs[0] = 0;
    }
}
```

Listing 3. Using *VssBackupComponents* to iteratively delete VSCs.

A protection mechanism must account for enforcing the validation of COM method interactions and interface transactions and be aware of unauthorized access and manipulation across various system components.

(4) Direct Device Access. One can directly control corresponding devices through IOCTLs to manipulate VSCs. Control code `FSCTL_DISMOUNT_VOLUME` and `IOCTL_VOLUME_OFFLINE` can be used to exhibit VSCs, and finally `IOCTL_VOLSNAP_DELETE_SNAPSHOT` to delete certain VSC. In addition, control code `IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE` can be used to control the size of the 'diff area', causing automatic snapshot dropping. They should also be protected against unauthorized access and manipulation of backup resources.

A protection mechanism must account for ensuring rigorous authorization and verification of IOCTL commands to safeguard against malicious direct hardware interactions and manipulation of backup resources.

Empirical Study on Ransomware Dataset. In our analysis of a real-world ransomware dataset introduced in Section 6, we observe distinct patterns in attack classifications of four backup deletion methods. Results show that 86.0% of ransomware uses native living-off-the-land binaries to complete malicious rollback prevention actions, 12.9% uses scripting languages objects, 2.8% manipulates COM object interactions, and 1.7% completes direct device access. Some ransomware uses a combination of techniques to ensure complete removal. These findings help us realize the current threat landscape and point to emerging trends in ransomware tactics. Remembrall should protect against all of the previously listed attack vectors with a unified architectural design.

4 Remembrall Design

For ransomware attacks, deleting data backups and system copies to prevent system rollback is a step that cannot be skipped and hidden. Ransomware has to ensure that the victim host's data is completely disrupted and cannot be recovered on its own, i.e., it causes a complete loss of data accessibility and usability to facilitate the success of the ransom. Thus, Remembrall maintains a single, core invariant: *malicious rollback prevention actions signal the existence of ransomware and cannot be bypassed.* Remembrall interrupts the attack during the process, achieving real-time detection and attack interception to prevent further harm to users. Its general workflow is shown in

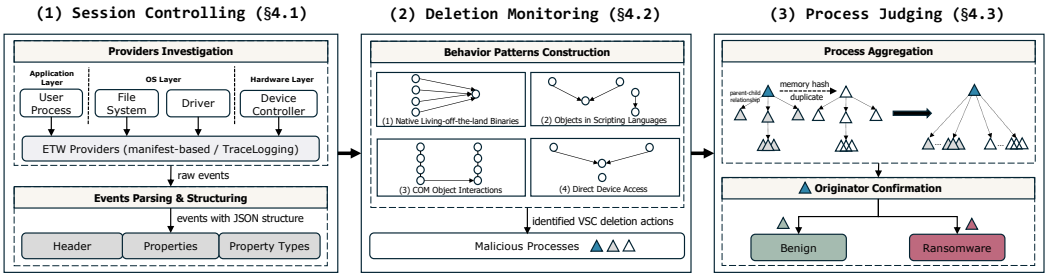


Fig. 1. Overall workflow of REMEMBRALL. It contains three modules: Session Controlling, Deletion Monitoring, and Process Judging. If ransomware is detected, all corresponding processes will be killed, and the system backup disruption attack (i.e., VSCs deletion) will be terminated.

Figure 1. Below, we detail the design of three modules to present how the existence of ransomware is detected and mitigated.

4.1 Session Controlling

The goal of this module is to retrieve data from ETW providers, parse events, and manage logs in a structured format. We first investigate all ETW providers in Windows to identify the ones that can generate ETW events that are potentially related to malicious system backup disruptions. Then, we design the parsing and structuring strategies to parse ETW data and provide structured data to the next module for further analysis.

Providers Investigation. By investigating the ETW providers in Windows, we can find the most suitable providers to generate events potentially related to malicious system backup disruptions. Through accurate and thorough investigation, we narrow down the number of providers to avoid excessive system loads and data noise, while still ensuring that needed data will not be missed.

In detail, there are two kinds of providers, manifest-based providers and TraceLogging providers. Manifest-based providers require an XML manifest file to declare the structure and properties of events, which allows events to be explicitly defined and reused across multiple tools and applications. In contrast, TraceLogging providers do not require an external manifest file; they allow events to be defined directly in the source code, simplifying deployment and maintenance. We consider both types of providers. For the Windows 11 22H2 system, we manually investigate all of the 1,121 providers and focus on the following perspectives: (1) the associated information they provide, for example, VSS interaction, process execution, file I/O, disk I/O, etc; (2) whether they have high events count, which indicates the higher probability to contain interesting events; (3) the coverage of the application layer, the OS layer, and the hardware layer. By following these three strategies, we ultimately select 15 providers to generate ETW events in both the OS's user mode and kernel mode, as shown in Table 2.

Events Parsing and Structuring. In this step, we parse ETW events from the selected ETW providers and structure them into a JSON object format for further analysis. There are two main ways to collect ETW events: interacting with the native ETW API and parsing events, or writing the disk as binary *etl* files using a combination of older built-in Windows tools and then processing them. For a more convenient and low-overhead implementation, we follow the first way and leverage the feature-rich KrabsETW Library [37] to enable detailed filtering and triage of ETW providers and events. To be more specific, we define a *parse_event_to_json* function to convert raw ETW events to a JSON object. This function takes an argument of type `EVENT_RECORD` (which is a Windows API structure used to represent ETW events), and an argument of type *krabs::schema*

Table 2. The ETW providers used in REMEMBRALL, categorized into user and kernel traces. After manually investigating all 1,121 providers, we select 15 for generating events in both user and kernel modes. The provider names and GUIDs are listed for User Traces, and only provider names for Kernel Traces.

User Traces: provider names and their corresponding GUID.	
Provider Name	**Provider GUID**
Microsoft-Windows-Powershell	a0c1853b-5c40-4b15-8766-3cf1c58f985a
VSS-tracing-provider	9138500e-3648-4edb-aa4c-859e9f7b7c38
Microsoft-Windows-WMI	1edeee53-0afe-4609-b846-d8c0b2075b1f
Microsoft-Windows-WMI-Activity	1418ef04-b0b4-4623-bf7e-d74ab47bbdaa
Microsoft-Windows-Kernel-Process	22fb2cd6-0e7b-422b-a0c7-2fad1fd0e716
Microsoft-Windows-Backup	1db28f2e-8f80-4027-8c5a-a11f7f10f62d
shell32-trace	382b5e24-181e-417f-a8d6-2155f749e724
Kernel Traces: provider names.	
image_load	process
disk_io	file_io
disk_file_io	file_init_io
disk_init_io	system_call

Table 3. The components of a JSON object are the Header Field, Properties Field, and Property Types Field. The entries in the Property Types Field are the same as those in the Properties Field and are therefore displayed together. Each category lists the respective fields and types as defined within the JSON structure, providing an overview of the hierarchical organization of the data elements.

Header Field:		
activity_id	event_flags	event_id
event_name	event_opcode	event_version
process_id	provider_name	task_name
thread_id	timestamp	trace_name
Properties Field, Property Types Field:		
ApplicationId	CommandLine	DirectoryTableBase
ExitStatus	Flags	ImageName
ImageChecksum	PackageFullName	ParentProcessId
ProcessId	SessionId	UniqueProcessKey
UserSID	SysCallAddress	SyscallNtStatus
ClientMachine	ClientProcessId	GroupOperationId
Operation	Filekey	FileObject

(which is a class in the KrabsETW library used for parsing and accessing metadata about ETW events). The output JSON format of one event is shown in Table 3, providing an overview of the hierarchical organization of the data elements. To efficiently manage the amount of data generated during event logging, the event data is first stored in memory buffers, then written to the log file and flushed at regular intervals. This reduces performance impact and supports efficient data processing for the following Deletion Monitoring module.

After this module, the structured events are generated, which contain rich system status information. They also become the input of the next module by real-time delivery.

4.2 Deletion Monitoring

The goal of this module is to filter all possible events to obtain ones related to system backup disruptions, i.e., the deletion actions of VSCs. As introduced in Section 3, there are four types

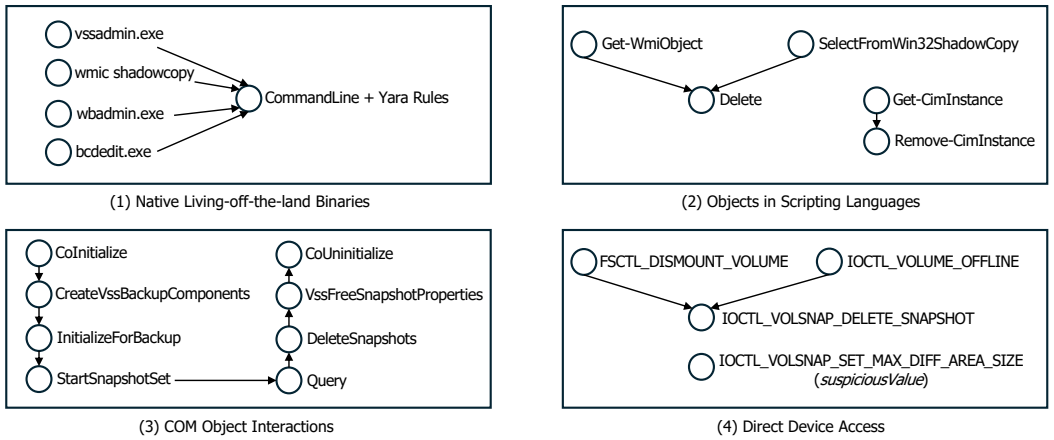


Fig. 2. The high-level behavior patterns for four types of system backup disruptions that we proposed in Section 3. We follow these patterns to monitor the VSC deletion attacks from native living-off-the-land binaries, objects in scripting languages, COM object interactions, and direct device access, respectively.

of attacks to delete VSCs, including the use of living-off-the-land binaries, the use of objects in scripting languages, COM object interactions, and direct device access. Based on these types, we summarize four rules for deletion monitoring to identify VSC deletion actions.

Behavior Patterns Construction. We studied different system backup disruptions across various scenarios. Our analysis shows that although these attacks have very different attack strategies (e.g., OS layer, component manipulation, triggering manner), each can be categorized into an attack pattern based on their behavior requests. Figure 2 shows the high-level behavior patterns for multiple system backup disruptions we studied during our investigation and experiment.

(1) As for the use of native living-off-the-land binaries, the `task_name`, `trace_name` Header Field and the `CommandLine`, `ImageFileName` Properties Field can record clear traces of command execution. Nevertheless, the `CommandLine` Properties Field demonstrates a syntactically diverse nature of command parameters in the user process and obfuscated forms of multiple parameter pairings. To deal with this, we write Yara rules [52] to capture all kinds of deletion actions using vssadmin.exe, wmic shadowcopy, wbadmin.exe and bcdedit.exe.

(2) As for the use of objects in scripting languages, the `activity_id`, `provider_name`, `trace_name` Header Field and the `ClientMachine`, `ClientProcessId`, `GroupOperationId`, `Operation` Properties Field record information that form a behavior pattern of (Get-WmiObject, Delete), (SelectFromWin32ShadowCopy, Delete), and (Get-CimInstance, Remove-CimInstance).

(3) As for COM object interactions, the `provider_name`, `task_name`, `trace_name` Header Field and the `SysCallAddress` Properties Field record the access of various system components. If the behavior pattern series of (CoInitialize, CreateVssBackupComponents, InitializeForBackup, StartSnapshotSet, Query, DeleteSnapshots, VssFreeSnapshotProperties, CoUninitialize) are detected, this indicates the existence of malicious COM object interactions to prevent rollback.

(4) As for direct device access, the patterns of (`FSCTL_DISMOUNT_VOLUME` |`IOCTL_VOLUME_OFFLINE`, `IOCTL_VOLSNAP_DELETE_SNAPSHOT`) and (`IOCTL_VOLSNAP_SET_MAX_DIFF_AREA_SIZE`, *suspiciousValue*) are constructed to detect this kind of system backup disruptions.

Specifically, in the first type of system backup disruptions, i.e., the use of native living-off-the-land binaries, the execution of binaries contains various parameters. The binary itself is benign,

but with various well-curated parameter combinations, it can be manipulated to conduct malicious actions. To detect such attacks, we write Yara rules to cover different combinations.

After this module, any VSC deletion actions are identified in time. Whether they are caused by ransomware will be judged in the next module.

4.3 Process Judging

The goal of this module is to determine whether the detected VSC deletion actions are initialized by ransomware or benign programs of the end-user, to avoid false positives. Also, searching for the analogous processes and aggregating them can help identify all sources of ransomware, avoiding false negatives. By process judging, we will track and terminate all ransomware-related processes.

Many seemingly unrelated processes can originate from the same program entity. Like other types of malware [57], ransomware utilizes camouflage strategies, e.g., self-replication, to conceal its presence and distribute its malicious activities across sub-modules. It is crucial to analyze the interrelationships among these ransomware-related processes and consolidate the disparate malicious actions to uncover disguised ransomware activities comprehensively. By synthesizing the behaviors of these analogous processes, we can construct a comprehensive sequence of behavioral data that accurately depicts the suspicious activities of the malware, even when camouflaged.

The homology of a process can be determined based on `UniqueProcessKey`, `ParentProcessID`, `ProcessID`, `ImageName`, `ImageChecksum` and `UserSID`, which are recorded into Properties Field of the event JSON object. The `UniqueProcessKey`, `ParentProcessID`, and `ProcessID` refer to the ID of the process that performs the operation that triggers Deletion Monitoring. Based on the parent-child relationship, it is possible to preliminarily determine the process that is spawned by the program through *fork()*, *execve()*, or *system()*. However, it ignores processes spawned by re-executing the same ransomware binary. Therefore, we address this issue by introducing path metrics to compare the exact paths of binaries, i.e., `ImageName`. However, although re-executed processes can be recognized by the same paths of the binaries even after self-deletion, ransomware can copy or rename its binaries and change the paths for camouflage purposes. Therefore, we further consider the hash value of the memory-mapped file, i.e., `ImageChecksum`, to counter other camouflage techniques. `UserSID` is the user security identifier to determine which user account triggered the event.

Specifically, Algorithm 1 presents **Process Aggregation** and **Originator Confirmation**. This algorithm receives a list of ETW events E as input and outputs a mapping from these events' process to their originator. Section (1) initializes the array P used to store the final results and two maps - OMap (origin program map) and PMap (process origin map). OMap is used to associate each unique origin identifier with the corresponding program information. PMap is used to associate a process ID with its origin information, including the origin ID and a list of associated events. Section (2) creates an origin relationship for each process based on ETW events and updates the event information. It first checks whether each process already has an origin mapping. If not, it looks to see if the process is a child of a known process. If it is, it categorizes it under the origin of the parent process. If it is not a child process, the algorithm creates a new origin identifier for it and records its program information, including program user SID, program name, and image checksum. All events are added to the list of events corresponding to the origin. Section (3) determines the final originator of each originating program after all the data has been aggregated. By traversing each origin and calling the *get_final_originator* function to analyze all of the events associated with it, we can analyze which events best represent the core behaviors or attributes of that origin and determine the final originating program.

Algorithm 1: Aggregating Process Origins and Determining Originators

```
Input  : E, ETW events for analysis
Output : P, Mapping from processes to their originator
P ← ∅, OMap ← new Map(), PMap ← new Map()                          // (1)
for each event in E do
    pid ← event.ProcessId
    ppid ← event.ParentProcessId
    if event.UniqueProcessKey ∉ PMap then                          // (2)
        if ppid ∈ PMap then
            PMap[pid].origin ← PMap[ppid].origin
        else
            originId ← generate_new_origin_id()
            OMap[originId].proInfo ← event.proInfo
            PMap[pid].origin = originId
            PMap[originId].events.add(event)
            P.add(originId, OMap[originId])
    else
        origin ← PMap[pid].origin
        OMap[origin].events.append(event)
end
for each origin in OMap do                                         // (3)
    finalOriginator ← get_final_originator(origin.events)
    origin.finalOriginator ← finalOriginator
end
```

After this module, benign processes that perform VSC deletion actions continue, while any ransomware-related processes are identified and terminated. The compromised system can then proceed to recovery.

5 Implementation

Our Remembrall prototype implements the *Session Controlling* module, the *Deletion Monitoring* module, and the *Process Judging* module described in Section 4 on Windows 11 22H2 version, targeting the x86-64 architecture. It can run as an underlying back-end program to sense the existence of ransomware in real time.

In fact, Remembrall can be implemented on any operating system that (1) is part of the Windows series and (2) the version is higher than Windows 8. The first requirement ensures that our attack classification is reasonable and accurate because VSS is a rollback support mechanism specifically for Windows systems. Other operating systems, such as Linux, macOS, and Android, have different mechanisms and system services for backups and, therefore, require different implementations according to Remembrall's *invariant* and modular design. The second requirement enables Remembrall to receive event logs from multiple ETW kernel providers. The reason is that prior to Windows 8, there could only be one kernel session; in this case, using kernel providers is problematic because any other malicious programs that use kernel sessions could potentially interfere by overriding the single kernel model event logging session. Our prototype customizes Sealighter [46] for part of event parsing. Overall, we implement our prototype using 3,409 lines of C++ and Python.

6 Evaluation

In this section, we evaluate the effectiveness and performance of Remembrall. The evaluation aims to answer the following questions:

- RQ1. Can Remembrall defend ransomware attacks with better performance than state-of-the-art approaches? (Section 6.1)
- RQ2. Can Remembrall detect and block the ransomware in a timely manner with low file loss in real-time detection? (Section 6.2)
- RQ3. What is the system overhead of Remembrall? (Section 6.3)
- RQ4. Can Remembrall combat zero-day ransomware samples? (Section 6.4)

Experimental Setup. In our experiments, we run real-world ransomware samples in the 64-bit Microsoft Windows 11 22H2 system. Experiments are undertaken on a server with an Intel(R) Core(TM) i7-10510U CPU with 4 cores and 16 GiB of memory. During sample execution, we launched two running guests in parallel. Each running guest operates within an isolated environment that does not affect or interfere with the others and will not do harm to the actual physical world.

Ransomware Dataset. To evaluate Remembrall, we use various real-world ransomware samples. We initially collect ransomware samples from VirusTotal [3], MalwareBazaar [1], tutorialjinni [2], and MarauderMap [23]. we perform deduplication by calculating their SHA-256 hash values. Then, we build a testbed to run all samples we collect, and keep them in our ransomware dataset only if (1) it is flagged as ransomware by more than five vendors on VirusTotal; (2) we observe that it encrypts files in the victim machines; (3) a ransom note is shown, or the wallpaper is changed after disruption occurred. The standard of measurement is consistent with the observation in the previous work [4, 23]. To be more specific, we put each sample in a clean, running guest with the aforementioned system files and user files as the testbed and execute it for ten minutes, observing its behaviors within the running guest machine. After each round of the experiment, we reset the running guest to a clean one without being compromised and then run another sample for the next round. We found out that some samples are not actual ransomware in our definition, though they are flagged as ransomware by these vendors, or key behaviors are not observed in our testbed, or they are not active due to their corresponding C&C servers being down in May 2024.

Table 4. Ransomware families and corresponding samples count. Our dataset contains 178 active and unique samples from 60 families.

No.	Family	Count	No.	Family	Count	No.	Family	Count	No.	Family	Count	No.	Family	Count
1	Blocker	15	**13**	Kryptik	4	**25**	Targeted	2	**37**	CTBLocker	1	**49**	Magniber	1
2	Phobos	15	**14**	Regrun	4	**26**	Vohuk	2	**38**	Dalexis	1	**50**	Mallox	1
3	Chaos	14	**15**	Teslacrypt	4	**27**	Abyss	1	**39**	Dorpal	1	**51**	Mydoom	1
4	Dharma	12	**16**	MoneyMessage	3	**28**	ALPHV	1	**40**	Dropper	1	**52**	Paradise	1
5	Sivis	11	**17**	RedLineStealer	3	**29**	AvosLocker	1	**41**	Elbie	1	**53**	Prestige	1
6	GenericKD	9	**18**	Avaddon	2	**30**	BlackCat	1	**42**	FileCryptor	1	**54**	SageCrypt	1
7	Babuk	7	**19**	Filecoder	2	**31**	BlueCarb	1	**43**	GandCrab	1	**55**	Sality	1
8	MedusaLocker	6	**20**	Generickdz	2	**32**	Conti	1	**44**	Hardbit	1	**56**	Saturn	1
9	BlackBasta	5	**21**	Neshta	2	**33**	Critroni	1	**45**	Hive	1	**57**	Shade	1
10	Makop	5	**22**	Pony	2	**34**	CryLock	1	**46**	Kuiper	1	**58**	Surtr	1
11	Venus	5	**23**	Royal	2	**35**	CryptFile2	1	**47**	LockBit	1	**59**	Voidcrypt	1
12	Cerber	4	**24**	Spora	2	**36**	CryptoLocker	1	**48**	Loki	1	**60**	WannaCry	1

Total samples count: 178.

Through the test running and filtering process, we build a real-world dataset with 178 active samples from 60 ransomware families, as shown in Table 4. These samples were collected over a period ranging from November 2023 to March 2024. Regarding the first-seen timestamp distribution of the samples, 7 samples (3.9%) were reported before 2021, while the remaining 171 samples (96.1%) were first reported between 2021 and 2024. This distribution can reflect the recent evolution trends

in ransomware. In addition, we would like to note that, the use of diverse families is more important than the number of ransomware samples from a few families for evaluating the performance of anti-ransomware tools. This is because the core behavioral traits remain almost identical from one variant to another in the same family [59]. Thus, we believe our dataset achieves a satisfying diversity of family counts while containing enough samples at the same time. A full round of executions of all the ransomware samples in this dataset will take nearly 30 hours in total.

Benign Dataset. Similar to other works [25, 58], we consider two kinds of benign programs for comparison, ones with similar functions to ransomware and other common office software in daily use scenarios. As shown in Table 5, we select a list of 12 benign applications of their latest versions. Some of them contain similar functions to ransomware, like 7-Zip [47] and WinRAR [56] as the archiver and extractor, DiskCryptor [45] and AES Crypt [42] for encryption, Eraser [18] and SDelete [38] as the shredder, and DiskGenius [30] and DiskPart [35] as the disk manager. The others are common office software in the end-user's daily use scenario, like Edge [36] and Chrome [21] as the browser, Notepad++ [22] as the text editor, and WMPlayer [39] as the video player. In the experiments, we run each of these applications for ten minutes in one round, the same time period with one ransomware sample. After 15 rounds of different user actions, we execute these benign applications 178 times in total, achieving a balanced number with ransomware samples.

Table 5. The list of benign applications with similar behaviors to ransomware and other common office software in the end-user's daily use scenario. We use their latest versions for experiments.

Application	Main Capability	Version	Application	Main Capability	Version
7-zip	Archiver, Extractor	23.01	Eraser	Shredder	6.2.0.2993
WinRAR	Archiver, Extractor	6.22	SDelete	Shredder	2.05
DiskGenius	Disk manager	5.6.0.1565	DiskCryptor	Encryption	1.3.0b
DiskPart	Disk manager	10.0.22621.1	AES Crypt	Encryption	v310
Edge	Browser	126.0.2592.87	Notepad++	Text editor	8.6.8
Chrome	Browser	126.0.6478.127	WMPlayer	Video player	12.0.19041.1288

6.1 Effectiveness of Detection

Compared Tools. We choose seven state-of-the-art anti-ransomware tools for effectiveness comparison considering their popularity and detection perspective: Unveil [25], ShieldFS [12], Redemption [26], PayBreak [27], Peeler [4], DeepWare [19], and RTrap [20]. We contacted the authors, but none provided original artifacts except for ShieldFS (an old 2015 version of win10 1511.10586.0); thus, we obtained the implementation code of some tools from [23, 59], and implemented the others from scratch by strictly following their papers. In addition, some of them are offline detection based on behavior logs, so we also collect corresponding data such as I/O request packets (IRP), API calls, and HPC values when running our ransomware samples.

Effectiveness Results. Experimental results are shown in Table 6. Remembrall achieves 100.00% detection rate, 2.81% false positive rate, 0.00% false negative rate, 97.27% precision rate and 98.61% F1-score. Compared to other tools, PayBreak monitors the encryption API usage, but many ransomware samples use their own encryption algorithm implementation, which results in significant false negatives; their pre-defined rules are too tight and cause false positives. Redemption, Unveil, and ShieldFS all focus on I/O behavior, monitoring whether high-entropy buffers are written to files, whether specific I/O sequence patterns indicative of ransomware are present, or assign weights to several I/O-related features and then calculate the final score to determine whether the program exceeds the threshold. They ignore other key perspectives and thus lead to unsatisfied detection results. RTrap uses a data-driven decoy file strategy to detect and contain ransomware, but

Table 6. Effectiveness comparison between Remembrall and other seven state-of-the-art anti-ransomware tools. Remembrall achieves a detection rate of 100.00% with 2.81% false positives, 4.31%-87.55% increase in F1-score compared to others.

Tool	Recall	FPR	FNR	Precision	F1-score
PayBreak	6.22	88.94	93.78	50.00	11.06
Unveil	36.70	0.35	63.30	95.24	52.98
Redemption	42.71	0.70	57.29	91.11	58.16
RTrap	49.44	3.37	50.56	93.62	64.71
Peeler	70.05	5.95	29.94	78.48	74.03
ShieldFS	100.00	10.43	0.00	81.10	89.57
DeepWare	97.90	5.70	2.10	90.95	94.30
Remembrall	**100.00**	**2.81**	**0.00**	**97.27**	**98.61**

ransomware may also adapt its tactics to avoid or identify decoy files, thus resulting in significant false negatives; legitimate software or users may also accidentally trigger the decoy monitor by interacting with these decoys, leading to false positives. Peeler collects limited malicious commands from the single ETW Provider *Process*, which misses many ransomware traces, so its Recall and FNR performances are limited. DeepWare collects HPC values and turns them into greyscale images for CNN model construction. As a side-channel sensing method, it is limited by the reliability of HPC data as indicators and the interpretability of its model. To conclude, experimental results show that Remembrall is an overall better-performed tool than state-of-the-art anti-ransomware methods.

Actual Realistic Evaluation. To ensure that Remembrall does not impact daily user operations, we also conducted a realistic evaluation. We recruited three volunteers and observed their regular activities on computers, including daily office tasks and leisure pursuits, over a one-week period on three computers equipped with Remembrall. The results demonstrate that our tool neither generates false positives nor disrupts benign processes during normal usage.

6.2 Detection Delay

In this section, we evaluate the speed of Remembrall in ransomware detection and quantify the extent of detection delay. Most ransomware delays launching an attack after it starts executing, and often remains inactive for a period of time before encrypting a file [58]. Therefore, it may not be appropriate to evaluate the efficiency of a detector using the detection time (i.e., the time interval between the start of ransomware execution and the detection of ransomware in the testbed). Given that ransomware's goal is to encrypt as many files as possible, we believe that the anti-ransomware tool that can detect ransomware with a small number of encrypted files can also prove to be efficient in terms of detection speed, thus controlling the detection delay. Therefore, we use the number of files lost as a criterion for evaluating the detection delay.

We select the latest real-time anti-ransomware tool, RTrap, for comparison. Experimental results are shown in Figure 3. To account for the varying encryption speeds of different ransomware samples, we report results based on ransomware samples that were accurately detected by both tools. RTrap loses a minimum of 46 files and a maximum of 239 files, averaging a loss of 131 files, with the medium number 124. In contrast, Remembrall experiences a minimum loss of 3 files and a maximum of 20 files; the medium number is 5. On average, Remembrall records a loss of 8 files due to ransomware attacks. These findings present Remembrall's capacity to rapidly detect ransomware with minimal file loss, showing its control over detection delay and real-time detection capabilities. To explain further, RTrap relies on file I/O, which needs to accumulate statistics over time, resulting in longer detection latency. Instead, Remembrall focuses on the unique behavioral

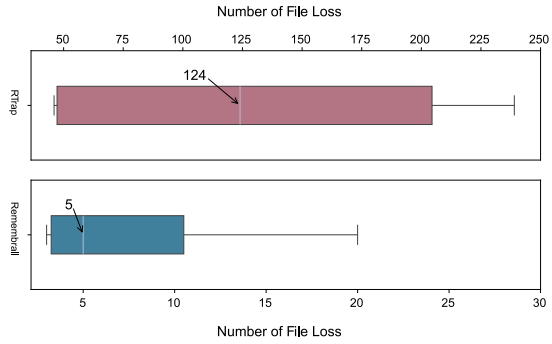


Fig. 3. The number of files that have been encrypted when the anti-ransomware tool detects attacks. Given that ransomware's goal is to encrypt as many files as possible, the file loss metric reflects the detection latency of real-time security tools.

patterns of system backup disruption and monitors clear instantaneous actions, resulting in faster detection times and fewer lost files. Recovery of encrypted files through external backups, such as cloud backups, falls outside the scope of this paper.

6.3 Overhead of Remembrall

It is critical to ensure that security mechanisms are lightweight and do not take up too many system resources. To evaluate the overhead of Remembrall, we measure the CPU and memory usage of the system under the conditions of without and with Remembrall protection. Hardware usage is collected every five seconds for ten consecutive minutes after system startup to avoid additional resource costs due to high sampling frequency. The ten-minute interval is consistent with the previously described experimental setups, and the hardware of the machine used for this experiment has also not changed.

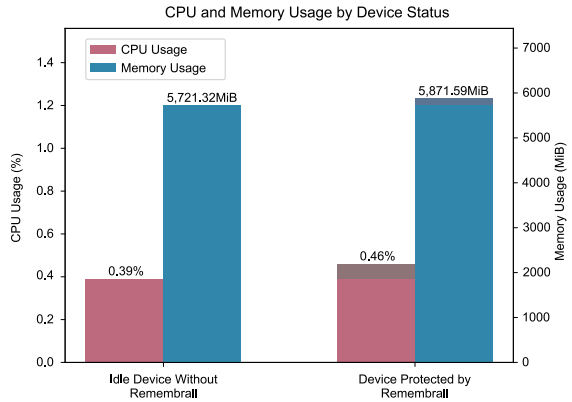


Fig. 4. CPU and memory usage of Remembrall on a freshly booted machine. The first 'Idle Device' group shows hardware usage without Remembrall installed, while the increments in the second group reflect Remembrall's overhead - 0.07% CPU and 150.27 MiB memory.

The average CPU and memory usage is presented in Figure 4, showing that Remembrall brings acceptable overhead. The overall CPU usage is 0.39% after the system is fully booted, with 5,721.32 MiB of memory consumption. After Remembrall is started and runs in the back-end, the CPU usage has increased to 0.46%, and the memory consumption increases to 5,871.59MiB, with an additional 0.07% of CPU usage and 150.27 MiB of memory used by Remembrall. In designing and implementing Remembrall, various mechanisms have been considered to minimize overhead. For example, in the Session Controlling module, we use native ETW API to collect events and deal with them in real-time data flow, rather than write them to the disk as binary *etl* files. In addition, when tracing and aggregating processes in the Process Judging module, we use an incremental strategy, prioritizing lighter-weight metrics.

6.4 Zero-day Ransomware Defense

Given the rapid evolution of ransomware, we further evaluate the effectiveness of Remembrall against eight zero-day ransomware samples. Table 7 provides the details of these samples, including their family classification, SHA-256 hash values, and the date they were first reported on VirusTotal. These samples are not part of our previous ransomware dataset. Note that the anti-ransomware mechanisms used by Windows Defender[13], Kaspersky Total Security [28], and 360 Total Security [50] are not publicly available, as they are commercial products. Therefore, we treat them as black-box tools during testing. As soon as these commercial tools identify ransomware samples, they are uploaded to their malware libraries, meaning that after we conducted this experiment in early July 2024, they can no longer be considered zero-day samples. Experimental results indicate that other tools may fail to detect some zero-day variants. In contrast, Remembrall successfully identifies all of them. This demonstrates that our proposed method effectively mitigates the risks posed by ransomware attacks and can adapt to future threats.

Table 7. Performance comparison between Remembrall and three state-of-the-art production-level detection techniques against the newly emerged ransomware samples that we see as zero-day variants in early July 2024. ✓ refers to successful identification of ransomware attacks, while ✗ refers to a miss.

Family	SHA-256 Value	First Reported	Defender	Kaspersky	360	Remembrall
trojan.bacz	cd642c7b2e6fd20593593e89113988f0a5af0157c5d6f46312d9c51ab25276f7	2024-07-01 UTC	✗	✓	✗	✓
CryptLocker	58c7c16f0679415f37b75658d19a617ce3d471f0d68af2505939fd826907cd40	2024-06-28 UTC	✓	✗	✓	✓
PlayCrypt	bc381dbeff70b5869fa737860c8cd8a8684cc768981beb55543499efcd32bab7	2024-06-27 UTC	✓	✓	✓	✓
phobos	2a8353551d099c78ac100b44718a691142f8cc7879b47e842ee8491426e15c08	2024-06-26 UTC	✓	✓	✓	✓
Rapidstop	e67260804526323484f564eebeb6c99ed021b960b899ff788aed85bb7a9d75c3	2024-06-06 UTC	✗	✓	✗	✓
LockBit	311edf744c2e90d7bfc550c893478f43d1d7977694d5dcecf219795f3eb99b86	2024-05-30 UTC	✓	✓	✗	✓
Mallox	45a236e7aa80515aafb6c656c758faad6e77fb435b35bfa407aef3918212078d	2024-05-21 UTC	✗	✓	✓	✓
BlackLockbit	479d0947816467d562bf6d24b295bf50512176a2d3d955b8f4d932aea2378227	2024-04-28 UTC	✓	✓	✓	✓

Case Study. The newly-appeared variant of the CryptLocker family was first seen in the wild on June 28, 2024. Without Remembrall serving as a security tool, the compromised machine lost all of its user files within ten minutes, which means the ransomware sample has done a complete encryption task. However, after the machine was equipped with Remembrall, when the sample was executed, we found that Remembrall killed it at the time node of 23s, with only six files being encrypted. No significant system load fluctuations were observed during this process. By analyzing the tool logs, we found out that this ransomware sample tried to use the native binary of vssadmin.exe to delete the system backup, which matched the first behavior pattern defined by Remembrall. One notable thing is that the command line parameters combination in this sample is quite different from that in the past samples when calling vssadmin.exe to delete backups. However, Remembrall utilized Yara rules to flexibly detect the malicious use of the binary, and then verified the existence of ransomware and stopped its following attack actions.

7 Discussion

Ethic Considerations. Ethics is a top priority when collecting ransomware samples and conducting experiments. First, we chose legitimate and credible sources to collect the samples and disclosed the source of the samples in our paper. This ensures that our research is ethical and increases the credibility of our work. In addition, all experiments were conducted in a self-built testbed, isolated from external devices to ensure that real-world computer equipment or end-users were not affected. We believe that the experimental setup was not ethically questionable and is sufficient to investigate the link between rollback prevention attacks and the existence of ransomware.

Application Scenarios. In a real-time defense scenario, Remembrall can execute silently in the computer as a background program with low resource consumption, without affecting daily use and other functions. If ransomware is detected, Remembrall will block its execution. The part of encrypted files can be recovered by external backups, such as the cloud backup, which is out of the scope of this paper. In a sandbox scenario, Remembrall can report the presence of ransomware without affecting the data integrity on real devices. Moreover, our detection algorithm can be integrated into other intrusion detection tools and systems to improve their performance.

Potential Adversarial Evasion and Its Mitigation. Ransomware may evade detection by fooling the ETW, for example, by stopping a running ETW session associated with Remembrall and starting a fake session. This may cause Remembrall to receive no real system activities. We have not observed such advanced ransomware samples at this time, but adding a new module to prevent and recognize masking attacks against ETW is worth considering.

ETW Latency. There may be up to 1.8s latency in collecting data through ETW. In our work, this delay is mitigated by several measures, including rationally reducing the number of monitored ETW providers through investigation to optimize performance, experimentally determining the most suitable memory buffer size, and also ensuring sufficient hardware resources to speed up event processing. Experimental results in Section 6.2 further prove that the detection delay and file loss rate are acceptable. To further improve the efficiency of data collection and processing, our follow-up work will include developing a kernel driver, which will directly involve in controlling the collection and processing of system events to improve the overall monitoring performance.

Software Rather Than Hardware Approach. Many prior works [24, 40, 43, 49] have shown that in the hardware layer, the log-structured design of SSDs can be leveraged to sense ransomware activities. The key insight is that valid data will be marked as stale if it is encrypted and overwritten by ransomware; the Flash Translation Layer usually does not reclaim the space immediately but has a non-negligible latency, thus providing an opportunity to recover the user's data before garbage collection [53]. However, these methods are limited by the hardware type, so they are less likely to adapt to a wide range of computer machines. So, in this work, we focus on the software approach and implementation to achieve higher scalability. Our future directions also include the exploration of hardware / software co-design.

Practicality on Other Operating Systems. This paper does not cover the analysis of system backup disruptions and protection mechanism implementation on other operating systems, such as Linux, macOS, and mobile OSes. While the samples of these attacks are currently not too numerous (3.75% of the initial dataset we collected), they are escalating and can be further studied. As presented in Section 4, Remembrall maintains a single, core invariant: *malicious rollback prevention actions signal the existence of ransomware and cannot be bypassed.* Following this invariant and the Remembrall design of three modules, another prototype can be implemented and adapted to different operating systems. For example, for macOS ransomware defense, we can investigate the system backup disruption and its mitigation on Time Machine [55].

8 Related Work

For ransomware mitigation, in the application layer, monitoring process commands is common for detecting suspicious activities, such as the execution of scripts or binaries associated with ransomware. Peeler [4] employs NLP techniques to identify anomalies in ransomware process commands, thereby detecting malicious command execution. Additionally, file I/O operations are closely observed, as ransomware typically encrypts large numbers of files rapidly, changing file extensions and increasing their entropy. Unveil [25] uses pre-defined rules to identify ransomware by analyzing file I/O patterns. Redemption [26] and R-Locker [12] apply ML methods to classify ransomware based on their unusual I/O frequency and data entropy.

In the OS layer, tracking system APIs provide insights into process interactions with the operating system, where ransomware frequently employs specific APIs for encryption. PayBreak [27] monitors the use of system encryption APIs to detect ransomware and assist in the decryption of compromised files. Network traffic analysis is another critical perspective in ransomware detection, as it helps identify communication with C&C servers. RansomSpector [51] combines network activity with file activity to achieve a more precise detection pattern. Berrueta et al. [9] monitor the traffic exchanged between clients and file servers, using machine learning techniques to identify patterns in the traffic that indicate ransomware actions.

In the hardware layer, one approach is to monitor hardware instructions executed by the system. DeepWare [19] and RanStop [48] leverage HPCs embedded in the performance monitoring unit of modern processors to observe micro-architectural event sets, using deep learning methods to detect both known and unknown cryptographic ransomware variants. Another approach focuses on storage behaviors. RansomTag [32], RSSD [49], RansomBlocker [44] and AMOEBA [40] apply logging methods to SSDs and detect ransomware by analyzing the impact on these storage devices.

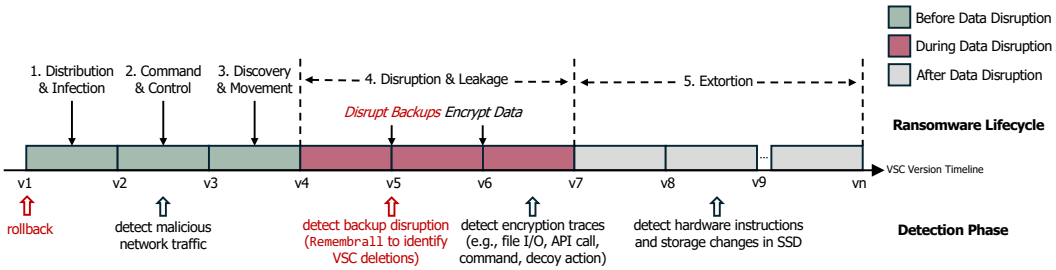


Fig. 5. Ransomware lifecycle stages and corresponding mitigation efforts. Our work targets an unexplored phase - detecting backup disruption - where current approaches fall short, distinct from previous research on network traffic, encryption traces, SSD logs, etc.

Main Differences. As shown in Figure 5, compared to previous work, Remembrall shifts focus on defending ransomware by monitoring its malicious operations on disrupting system backups, which signals an unexplored ransomware attacking phase. To the best of our knowledge, Remembrall is the first to consider potential threats to VSCs from the application layer, OS layer, and hardware layer presented above. The effectiveness of Remembrall is further validated by its high performance and low detection delay and runtime overhead, demonstrating its potential as a practical solution against ransomware.

9 Conclusion

This paper introduces Remembrall, a new anti-ransomware tool that protects Windows systems by monitoring backup disruptions and detecting deletion actions of volume shadow copies. Remembrall performs a comprehensive analysis of system backup disruption behaviors across different system layers to ensure accurate identification of ransomware activities. It achieves an F1-score improvement of 4.31%-87.55% over existing tools, with low detection delay and system overhead. The tool has also identified eight zero-day ransomware samples.

Data Availability

To ensure reproducibility and facilitate future research, the artifacts we used in the experiments are publicly available at: https://github.com/m1-llie/Remembrall.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback and suggestions. This research is sponsored in part by the National Key Research and Development Project (No. 2022YFB3104000), and NSFC Program (No. U2441238, 62021002).

References

[1] 2024. Bazaar. https://bazaar.abuse.ch/browse/. (Online; accessed on October 10, 2024).
[2] 2024. tutorialjinni. https://www.tutorialjinni.com/download-free-malware-samples.htm. (Online; accessed on October 10, 2024).
[3] 2024. VirusTotal. https://www.virustotal.com/gui/contact-us/premium-services. (Online; accessed on October 10, 2024).
[4] Muhammad Ejaz Ahmed, Hyoungshick Kim, Seyit Camtepe, and Surya Nepal. 2021. Peeler: Profiling kernel-level events to detect ransomware. In *ESORICS 2021: 26th European Symposium on Research in Computer Security* (Darmstadt, Germany, October 4–8). Springer, 240–260.
[5] Bander Ali Saleh Al-Rimy, Mohd Aizaini Maarof, and Syed Zainudeen Mohd Shaid. 2018. Ransomware threat success factors, taxonomy, and countermeasures: A survey and research directions. *Computers & Security* 74 (2018), 144–166.
[6] Omar Alrawi, Charles Lever, Kevin Valakuzhy, Kevin Snow, Fabian Monrose, Manos Antonakakis, et al. 2021. The Circle of life: A large-scale study of the IoT malware lifecycle. In *30th USENIX Security Symposium*. 3505–3522.
[7] Erin Avllazagaj, Ziyun Zhu, Leyla Bilge, Davide Balzarotti, and Tudor Dumitraș. 2021. When malware changed its mind: An empirical study of variable program behaviors in the real world. In *30th USENIX Security Symposium*. 3487–3504.
[8] Kenan Begovic, Abdulaziz Al-Ali, and Qutaibah Malluhi. 2023. Cryptographic ransomware encryption detection: Survey. *Computers & Security* 132 (2023), 103349.
[9] Eduardo Berrueta, Daniel Morato, Eduardo Magaña, and Mikel Izal. 2022. Crypto-ransomware detection using machine learning models in file-sharing network scenarios with encrypted traffic. *Expert Systems with Applications* 209 (2022), 118299.
[10] Chainalysis. 2024. Ransomware Payments Exceed $1 Billion in 2023, Hitting Record High After 2022 Decline. https://www.chainalysis.com/blog/ransomware-2024/. (Online; accessed on October 10, 2024).
[11] Fabrizio Cicala and Elisa Bertino. 2020. Analysis of encryption key generation in modern crypto ransomware. *IEEE Transactions on Dependable and Secure Computing* 19, 2 (2020), 1239–1253.
[12] 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.
[13] Windows Defender. 2024. Windows security: Defender, antivirus and more for windows 11. https://www.microsoft.com/en-us/windows/comprehensive-security. (Online; accessed on October 10, 2024).
[14] DoJ. 2023. U.S. Department of Justice Disrupts Hive Ransomware Variant. https://www.justice.gov/opa/pr/us-department-justice-disrupts-hive-ransomware-variant. (Online; accessed on October 10, 2024).
[15] DoJ. 2024. U.S. and U.K. Disrupt LockBit Ransomware Variant. https://www.justice.gov/opa/pr/us-and-uk-disrupt-lockbit-ransomware-variant. (Online; accessed on October 10, 2024).
[16] Melissa Eddy and Nicole Perlroth. 2020. Cyber Attack Suspected in German Woman's Death. https://www.nytimes.com/2020/09/18/world/europe/cyber-attack-germany-ransomeware-death.html. (Online; accessed on October 10, 2024).

[17] Abdulrahman Abu Elkhail, Nada Lachtar, Duha Ibdah, Rustam Aslam, Hamza Khan, Anys Bacha, and Hafiz Malik. 2023. Seamlessly safeguarding data against ransomware attacks. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 1–16.
[18] Eraser. 2024. Eraser. https://eraser.heidi.ie/. (Online; accessed on October 10, 2024).
[19] Gaddisa Olani Ganfure, Chun-Feng Wu, Yuan-Hao Chang, and Wei-Kuan Shih. 2022. Deepware: Imaging performance counters with deep learning to detect ransomware. *IEEE Trans. Comput.* 72, 3 (2022), 600–613.
[20] Gaddisa Olani Ganfure, Chun-Feng Wu, Yuan-Hao Chang, and Wei-Kuan Shih. 2023. RTrap: Trapping and Containing Ransomware With Machine Learning. *IEEE Transactions on Information Forensics and Security* 18 (2023), 1433–1448.
[21] Google. 2024. Chrome. https://www.google.com/chrome/. (Online; accessed on October 10, 2024).
[22] Don Ho. 2024. Notepad++. https://notepad-plus-plus.org/. (Online; accessed on October 10, 2024).
[23] Yiwei Hou, Lihua Guo, Chijin Zhou, Yiwen Xu, Zijing Yin, Shanshan Li, Chengnian Sun, and Yu Jiang. 2024. An Empirical Study of Data Disruption by Ransomware Attacks. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal). Article 161, 12 pages.
[24] 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.
[25] 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*. 757–772.
[26] Amin Kharraz and Engin Kirda. 2017. Redemption: Real-time protection against ransomware at end-hosts. In *Research in Attacks, Intrusions, and Defenses: 20th International Symposium, RAID 2017, Atlanta, GA, USA, September 18–20, 2017, Proceedings*. Springer, 98–119.
[27] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele. 2017. Paybreak: Defense against cryptographic ransomware. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 599–611.
[28] AO Kaspersky Lab. 2024. Kaspersky: Cybersecurity that's always a step ahead. https://usa.kaspersky.com/. (Online; accessed on October 10, 2024).
[29] Seungkwang Lee, Nam-su Jho, Doyoung Chung, Yousung Kang, and Myungchul Kim. 2022. Rcryptect: Real-time detection of cryptographic function in the user-space filesystem. *Computers & Security* 112 (2022), 102512.
[30] Eassos Ltd. 2024. DiskGenius. https://www.diskgenius.com/. (Online; accessed on October 10, 2024).
[31] Liang Luo, Suman Nath, Lenin Ravindranath Sivalingam, Madan Musuvathi, and Luis Ceze. 2018. Troubleshooting {Transiently-Recurring} Errors in Production Systems with {Blame-Proportional} Logging. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. 321–334.
[32] Boyang Ma, Yilin Yang, Jinku Li, Fengwei Zhang, Wenbo Shen, Yajin Zhou, and Jianfeng Ma. 2023. Travelling the hypervisor and ssd: A tag-based approach against crypto ransomware with fine-grained data recovery. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 341–355.
[33] Shiqing Ma, Kyu Hyung Lee, Chung Hwan Kim, Junghwan Rhee, Xiangyu Zhang, and Dongyan Xu. 2015. Accurate, low cost and instrumentation-free security audit logging for windows. In *Proceedings of the 31st Annual Computer Security Applications Conference*. 401–410.
[34] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. 2018. Rwguard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 114–136.
[35] Microsoft. 2024. diskpart. https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/diskpart. (Online; accessed on October 10, 2024).
[36] Microsoft. 2024. Edge. https://www.microsoft.com/en-us/edge/download?form=MA13FJ. (Online; accessed on October 10, 2024).
[37] Microsoft. 2024. Krabsetw. https://github.com/microsoft/krabsetw. (Online; accessed on October 10, 2024).
[38] Microsoft. 2024. SDelete. https://learn.microsoft.com/en-us/sysinternals/downloads/sdelete. (Online; accessed on October 10, 2024).
[39] Microsoft. 2024. Windows Media Player. https://support.microsoft.com/en-us/windows/get-windows-media-player-81718e0d-cfce-25b1-aee3-94596b658287. (Online; accessed on October 10, 2024).
[40] 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.
[41] Harun Oz, Ahmet Aris, Albert Levi, and A Selcuk Uluagac. 2022. A survey on ransomware: Evolution, taxonomy, and defense solutions. *ACM Computing Surveys (CSUR)* 54, 11s (2022), 1–37.
[42] Inc. Packetizer. 2024. AES Crypt. https://www.aescrypt.com/. (Online; accessed on October 10, 2024).

[43] Joon-Young Paik, Joong-Hyun Choi, Rize Jin, Jianming Wang, and Eun-Sun Cho. 2018. A storage-level detection mechanism against crypto-ransomware. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2258–2260.
[44] 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.
[45] pathtofile. 2024. Disk Cryptor. https://diskcryptor.org/. (Online; accessed on October 10, 2024).
[46] pathtofile. 2024. Sealighter. https://github.com/pathtofile/Sealighter. (Online; accessed on October 10, 2024).
[47] Igor Pavlov. 2024. 7-ZIP. https://www.7-zip.org/. (Online; accessed on October 10, 2024).
[48] Nitin Pundir, Mark Tehranipoor, and Fahim Rahman. 2020. RanStop: A hardware-assisted runtime crypto-ransomware detection technique. *arXiv preprint arXiv:2011.12248* (2020).
[49] Benjamin Reidys, Peng Liu, and Jian Huang. 2022. Rssd: Defend against ransomware with hardware-isolated network-storage codesign and post-attack analysis. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 726–739.
[50] 360 Total Security. 2024. 360 Total Security: Free antivirus Protect your security and privacy. https://www.360totalsecurity.com/en. (Online; accessed on October 10, 2024).
[51] Fei Tang, Boyang Ma, Jinku Li, Fengwei Zhang, Jipeng Su, and Jianfeng Ma. 2020. RansomSpector: An introspection-based approach to detect crypto ransomware. *Computers & Security* 97 (2020), 101997.
[52] VirusTotal/Yara. 2024. Yara. https://github.com/VirusTotal/yara. (Online; accessed on October 10, 2024).
[53] Zhongyu Wang, Yaheng Song, Erci Xu, Haonan Wu, Guangxun Tong, Shizhuo Sun, Haoran Li, Jincheng Liu, Lijun Ding, Rong Liu, Jiaji Zhu, and Jiesheng Wu. 2024. Ransom Access Memories: Achieving Practical Ransomware Protection in Cloud with DeftPunk. In *Proceedings of the 18th USENIX Symposium on Operating Systems Design and Implementation*. 1–16.
[54] Wikipedia. 2024. CryptoLocker. https://en.wikipedia.org/wiki/CryptoLocker. (Online; accessed on October 10, 2024).
[55] Wikipedia. 2024. Time Machine (macOS). https://en.wikipedia.org/wiki/Time_Machine_(macOS). (Online; accessed on October 10, 2024).
[56] win.rar GmbH. 2024. WinRAR. https://www.win-rar.com/. (Online; accessed on October 10, 2024).
[57] Yiwen Xu, Zijing Yin, Yiwei Hou, Jianzhong Liu, and Yu Jiang. 2022. MIDAS: safeguarding iot devices against malware via real-time behavior auditing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 41, 11 (2022), 4373–4384.
[58] Huan Zhang, Lixin Zhao, Aimin Yu, Lijun Cai, and Dan Meng. 2024. Ranker: Early Ransomware Detection through Kernel-level Behavioral Analysis. *IEEE Transactions on Information Forensics and Security* (2024).
[59] Chijin Zhou, Lihua Guo, Yiwei Hou, Zhenya Ma, Quan Zhang, Mingzhe Wang, Zhe Liu, and Yu Jiang. 2023. Limits of I/O Based Ransomware Detection: An Imitation Based Attack. In *2023 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2584–2601.

Received 2024-10-22; accepted 2025-03-31