Lessons Learned from the ShrinkLocker Ransomware: From Response to Detection

Cristian H. M. Souza[1,2] **, Eduardo O. Chavarro**[1]

[1]Global Emergency Response Team (GERT), Kaspersky
[2]Department of Computer Science, University of São Paulo

{cristian.souza, eduardo.ovalle}@kaspersky.com

Abstract. *ShrinkLocker is a ransomware threat that abuses the native BitLocker utility to encrypt entire volumes, using advanced VBScript and PowerShell techniques to exfiltrate keys, disable key recovery, and perform drive-level encryption. In this paper, we provide a detailed technical analysis of the malware, review the adversary's tactics, techniques and procedures (TTPs), and highlight key lessons learned from an incident response case. We also propose detection and mitigation strategies and emphasize the importance of behavioral-based monitoring over static signature detection in such advanced attacks.*

1. Introduction

Ransomware has evolved dramatically over the past decade, from rudimentary screen lockers to highly sophisticated families capable of encrypting entire infrastructures and exfiltrating sensitive data [McIntosh et al. 2024]. While early ransomware was typically written in compiled languages such as C or C++, recent trends have revealed a strategic pivot by threat actors toward languages like Go, Rust, and VBScript. This evolution is often motivated by a desire to minimize detection by traditional antivirus engines and to leverage the flexibility and ubiquity of interpreted languages across enterprise environments [Razaulla et al. 2023].

Compounding this trend is the increasing adoption of "living off the land" (LOTL) techniques, in which adversaries rely entirely on native system utilities already present on the target machine. This allows them to reduce their operational footprint, evade application whitelisting policies, and avoid dropping binaries that could trigger antivirus alerts [Geng et al. 2024]. In parallel, ransomware groups have shown growing interest in exploiting legitimate operating system features (such as PowerShell, Windows Management Instrumentation, and BitLocker) for malicious purposes, thereby blending in with routine administrative activities [De Gaspari et al. 2022].

A striking example of this evolution is the ShrinkLocker ransomware campaign [Souza et al. 2024]. Discovered by our team during a real-world incident response, ShrinkLocker illustrates a highly creative and destructive abuse of the native BitLocker disk encryption utility. Rather than relying on proprietary cryptographic functions or custom payloads, the malware uses a plain-text VBScript to manipulate drive partitions, disable recovery mechanisms, and invoke BitLocker to perform full-volume encryption. It then exfiltrates the generated decryption key to an attacker-controlled server via HTTP POST requests, leaving the victim system entirely inaccessible.

What makes ShrinkLocker particularly dangerous is its stealth: originally, the script is not obfuscated, and it performs no suspicious imports or dynamic library load-

ing. It capitalizes on administrative privileges and standard Windows utilities to achieve its goals. In a recent analysis, we found that attackers have embedded the encryption script within other files to exploit vulnerabilities in Windows systems and execute the encryption process by abusing administrative functionalities. This paper presents an in-depth reverse engineering analysis of ShrinkLocker, detailing its operational flow, tactics, and forensic evasion strategies. We also discuss the challenges it poses for traditional detection mechanisms and highlight important defensive lessons that organizations must internalize in order to prepare for future LOTL-style ransomware attacks.

1.1. Threat Background

Similar script components had already been detected in 2022, employing comparable access strategies and registry key modifications. According to Microsoft Threat Intelligence [1] and The DFIR Report [2] these were linked to the Advanced Persistent Threat (APT) actor identified as DEV-0270, also known as Nemesis Kitten. According to the initial analysis, the function calls and encryption key configuration followed a different structure at the time, typically executed separately. They also leveraged additional applications such as BitLocker's `BdeHdCfg.exe` (BitLocker Drive Encryption: Drive Preparation Tool) to shrink partitions and ensure system recovery. In contrast, current versions of the script have evolved significantly, executing the same functionalities natively (without requiring external applications) and avoiding detection mechanisms that rely on monitoring the installation or usage of legitimate system tools.

2. Technical Analysis

Malware analysis is of fundamental importance for information security because, through understanding the malware behavior and code, efficient mitigations can be developed [Souza et al. 2025]. In the following subsections, we provide a detailed analysis of the ShrinkLocker ransomware, focusing on its main operations and capabilities.

2.1. VBScript Behavior Overview

ShrinkLocker's primary payload is a VBScript file named `Disk.vbs`, placed in a hidden directory path. Unlike typical ransomware that encrypts files selectively, this script targets full disk volumes using BitLocker, leaving the system in a completely unusable state without the decryption key. The code is mostly unobfuscated, indicating that the attackers executed it only after gaining full administrative privileges.

2.2. Execution Conditions

As shown in Figure 1, the script begins by collecting information about the target system through WMI queries to the `Win32_OperatingSystem` class. It compares the system domain and checks OS version compatibility. If it identifies unsupported legacy operating systems such as Windows XP or Windows Server 2003, it self-terminates and deletes its traces. This reflects an effort to avoid unnecessary execution on non-viable targets.

[1] https://www.microsoft.com/en-us/security/blog/2022/09/07/profiling-dev-0270-phosphorus-ransomware-operations/

[2] https://thedfirreport.com/2021/11/15/exchange-exploit-leads-to-domain-wide-ransomware/

Figure 1. Conditions for execution

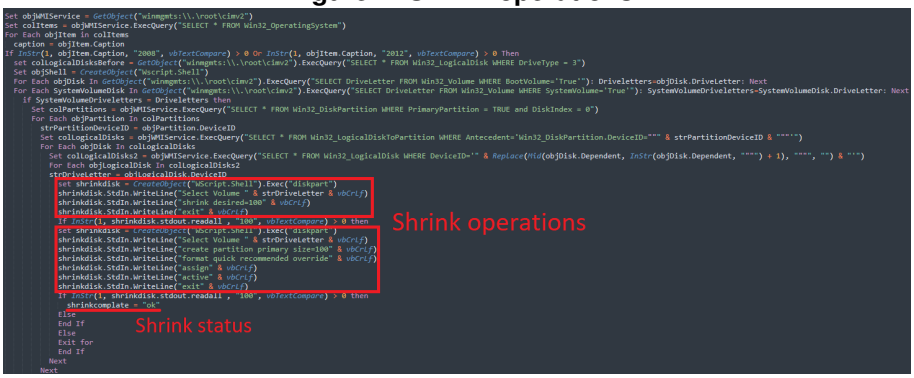
```
main
Sub Main
On Error Resume Next
Set objWMIService = GetObject("winmgmts:\\.\root\cimv2")
Set colItems = objWMIService.ExecQuery("SELECT * FROM Win32_OperatingSystem")

For Each objItem in colItems
  If InStr(1, CreateObject("ADSystemInfo").DomainDNSName, "[redacted]", vbTextCompare) > 0 Then
    else
      If Not condition then Exit Sub
  end if
    If InStr(1, objItem.Caption, "xp", vbTextCompare) > 0 Or InStr(1, objItem.Caption, "2000", vbTextCompare) > 0 Or InStr(1, objItem.Caption
      , "2003", vbTextCompare) > 0 Or InStr(1, objItem.Caption, "Vista", vbTextCompare) > 0 Then
        Set fso = CreateObject("Scripting.FileSystemObject")
        fso.DeleteFile "C:\ProgramData\Microsoft\Windows\Templates\Disk.vbs", True
     If Not condition then Exit Sub
    End If
Next
```

2.3. Partition Shrinking and Setup

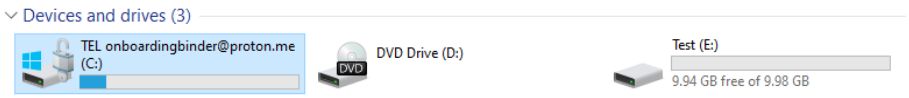
The script proceeds to identify all local fixed drives (DriveType = 3), excluding removable, network, or optical drives. Using the `diskpart` utility, it shrinks each non-boot partition by 100MB and uses the unallocated space to create new partitions. These are formatted with override options and given active status (Figure 2).

Figure 2. Shrink operations



It then reinstalls the bootloader files using `bcdboot`, ensuring the partitions are bootable and pointing to the modified configuration. The partition labels are replaced with the attacker's email, typically to prompt victims to negotiate ransom payments (Figure 3).

Figure 3. Contact information

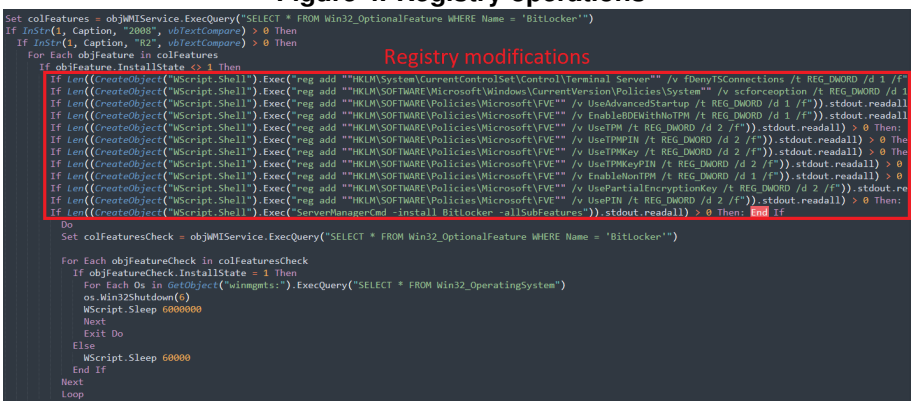


2.4. BitLocker Configuration and Encryption

The malware modifies multiple registry entries to ensure BitLocker can be activated without requiring TPM or smart card support. It disables RDP, forces advanced startup authentication, and ensures BitLocker accepts passwords or USB keys. All protectors (mechanisms for decrypting the drive) are removed, making recovery through normal methods impossible, as depicted in Figure 4.

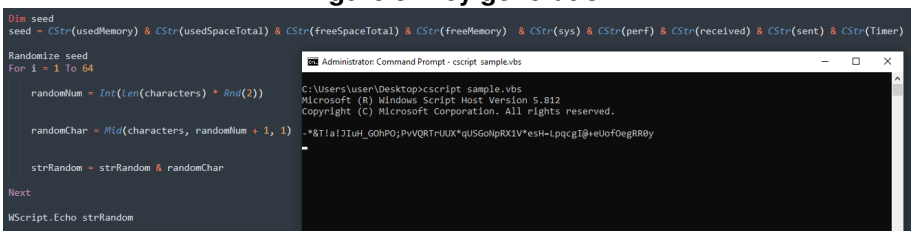
As illustrated in Figure 5, a new 64-character password is generated using a mix of system metrics and randomized input, including the pangram "The quick brown fox

Figure 4. Registry operations



jumps over the lazy dog". This password is stored only temporarily and is exfiltrated to an external C2 server hosted behind a Cloudflare tunnel. The script then enables BitLocker and applies the generated password as the only protector.

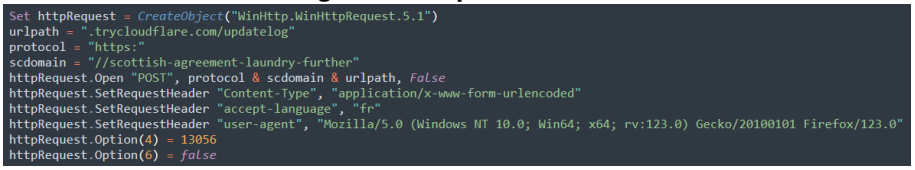
Figure 5. Key generation



2.5. Exfiltration and Cleanup

Once encryption is enabled, the script prepares an HTTP POST request containing the generated key, hostname, OS version, and affected drives (Figure 6). The Figure 7 shows an example of encoded data that is sent to the attacker's server. The malware also retries the operation multiple times upon failure. To cover its tracks, the script clears Windows PowerShell logs, deletes scheduled tasks, removes firewall rules, and initiates a forced shutdown. Upon reboot, victims are greeted with the BitLocker recovery screen but have no recovery key available.

Figure 6. Request creation



3. Evolution of Threat Techniques: Embedding Encryption Scripts into Diverse Exploitation Vectors

At the end of 2024 ShrinkLocker re-emerged with an optimized script variant and novel delivery technique. Our findings indicate the resurgence of the ShrinkLocker malware,

Figure 7. Data to be sent

this time using a streamlined version of its original script. This variant is embedded within an MSC (Microsoft Management Console) file, employing a technique referred to as GrimResource [3], a method designed to achieve initial access and evade detection. Key observations from the analyzed sample include the following:

- The script consists of approximately 155 lines of code, indicating a more compact and efficient implementation.
- Internal command-and-control (C2) communication is established via Web protocols using internal IP addresses that confirm active intrusion.
- There are no ransom notes or predefined communication channels with the threat actors.

4. Tactics, Techniques and Procedures (TTPs)

The attacker demonstrated in-depth knowledge of Windows internals and scripting. Table 1 lists observed TTPs aligned with the MITRE ATT&CK framework.

Table 1. MITRE ATT&CK Techniques used in ShrinkLocker

Technique ID	Description
T1059.005	Command and Scripting Interpreter: VBScript
T1059.001	Command and Scripting Interpreter: PowerShell
T1047	Windows Management Instrumentation
T1486	Data Encrypted for Impact
T1529	System Shutdown/Reboot
T1070.001	Clear Windows Event Logs
T1112	Modify Registry
T1562.004	Disable or Modify System Firewall
T1041	Exfiltration Over Web Service

5. Forensics and Artifact Recovery

One of the key challenges faced during incident response was the lack of forensic evidence on compromised systems. The malware deleted its own script, removed scheduled tasks, and wiped logs. Nevertheless, indirect evidence such as registry remnants, failed service startups, and externally logged PowerShell execution traces were useful in reconstructing the attack. Network monitoring tools that retained POST requests were instrumental in capturing exfiltrated payloads. On a few unaffected systems where BitLocker was not installed, analysts found copies of the script and log output, allowing for detailed reconstruction of the malware's logic.

[3]https://www.elastic.co/security-labs/grimresource

6. Mitigation Strategies

ShrinkLocker demonstrates the limitations of traditional antivirus tools against threats that abuse native Windows features like BitLocker, WMI, and PowerShell. Since it operates without deploying custom binaries, it can evade signature-based detection. To counter this, organizations should adopt a defense-in-depth approach combining prevention, detection, and recovery.

Endpoints should be hardened by configuring BitLocker with TPM and multifactor protectors, and by enforcing least privilege for administrative access. Unused scripting tools such as VBScript and legacy PowerShell should be disabled using AppLocker or Windows Defender Application Control. Critical registry paths (e.g., `HKLM\SOFTWARE\Policies\Microsoft\FVE`) should be audited regularly.

Detection should rely on behavioral analytics. EDRs must flag unusual use of disk utilities or PowerShell activity. Script block logging and transcription should be enabled and sent to centralized logging systems, since ShrinkLocker deletes local logs and tampers with scheduled tasks and firewall settings.

At the network level, outbound HTTP traffic should be fully logged, especially POST requests. TLS inspection can provide additional visibility where permitted, as ShrinkLocker uses encrypted tunnels to exfiltrate data. From an academic perspective, paradigms such as Software-Defined Networking (SDN) could improve malware detection through traffic analysis, especially by using machine learning, as demonstrated by [Souza and Arima 2024]. Additionally, Moving Target Defense (MTD) approaches can help build infrastructure resilient to the spread of cyber attacks, as described by [Dantas Silva et al. 2023].

Reliable recovery is critical. Backups must be offline, tested regularly, and inaccessible from production systems. BitLocker recovery keys should be centrally managed. Incident response plans must simulate ransomware scenarios to ensure organizational readiness.

7. Lessons Learned

ShrinkLocker reveals key weaknesses in conventional security models. Major takeaways include:

1. **LotL techniques:** The malware used only native tools (e.g., PowerShell, DiskPart, BitLocker), bypassing traditional AV and EDR detection.
2. **Behavioral detection:** Only behavioral monitoring caught early-stage anomalies; static rules and signatures failed.
3. **Centralized logging:** ShrinkLocker deleted local logs. Only organizations with external log aggregation retained critical forensic data.
4. **Network visibility:** Key exfiltration occurred via HTTP POST to Cloudflare. Lack of POST logging and TLS inspection hindered detection.
5. **Privilege misuse:** The attack required full admin rights, which enabled reconfiguration and encryption without user awareness.
6. **Recovery dependence:** Recovery was only possible where BitLocker was not enforced or backups were properly maintained.

These findings highlight the need for behavioral defenses, strong identity management, outbound traffic inspection, and comprehensive recovery plans.

8. Conclusion

ShrinkLocker exemplifies how attackers can weaponize built-in OS features to conduct stealthy and destructive operations. By leveraging native tools like BitLocker and minimizing binary dependencies, the threat bypasses conventional detection mechanisms. Our investigation underscores the importance of behavioral analysis, robust logging, and privilege management in mitigating such threats. We also emphasize the value of proactive logging, especially of outbound web requests, and the significance of securely storing and testing backup procedures.

After completing the incident response and investigation processes, we developed a YARA rule capable of generically detecting ShrinkLocker. This rule now helps protect approximately one billion devices worldwide. Additionally, ShrinkLocker has been included in version 17 of the MITRE ATT&CK framework[4].

References

Dantas Silva, F. S., Neto, E. P., Nunes, R. S., Souza, C. H., Neto, A. J., and Pascoal, T. (2023). Securing software-defined networks through adaptive moving target defense capabilities. *Journal of Network and Systems Management*, 31(3):61.

De Gaspari, F., Hitaj, D., Pagnotta, G., De Carli, L., and Mancini, L. V. (2022). Evading behavioral classifiers: a comprehensive analysis on evading ransomware detection techniques. *Neural Computing and Applications*, 34(14):12077–12096.

Geng, J., Wang, J., Fang, Z., Zhou, Y., Wu, D., and Ge, W. (2024). A survey of strategy-driven evasion methods for pe malware: Transformation, concealment, and attack. *Computers & Security*, 137:103595.

McIntosh, T., Susnjak, T., Liu, T., Xu, D., Watters, P., Liu, D., Hao, Y., Ng, A., and Halgamuge, M. (2024). Ransomware reloaded: Re-examining its trend, research and mitigation in the era of data exfiltration. *ACM Computing Surveys*, 57(1):1–40.

Razaulla, S., Fachkha, C., Markarian, C., Gawanmeh, A., Mansoor, W., Fung, B. C., and Assi, C. (2023). The age of ransomware: A survey on the evolution, taxonomy, and research directions. *IEEE Access*, 11:40698–40723.

Souza, C., Ovalle, E., Muñoz, A., and Zachor, C. (2024). Shrinklocker: Turning bitlocker into ransomware. Technical report, Kaspersky Securelist.

Souza, C. H., Pascoal, T., Neto, E. P., Sousa, G. B., Filho, F. S., Batista, D. M., and Dantas Silva, F. S. (2025). Sdn-based solutions for malware analysis and detection: State-of-the-art, open issues and research challenges. *Journal of Information Security and Applications*, 93:104145.

Souza, C. H. M. and Arima, C. H. (2024). A hybrid approach for malware detection in sdn-enabled iot scenarios. *Internet Technology Letters*, 7(6):e534.

[4]https://attack.mitre.org/software/S1178/