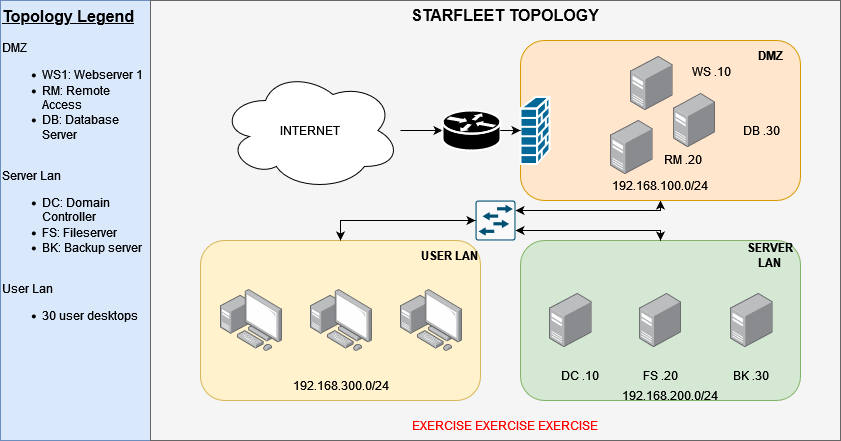
**COS80013 – Assignment 2 Part A: Forensic Analysis** 

**Impact at STARFLEET**

**1. What type of threat does this appear to be?**  
This is a ransomware attack. Chris Pike reported that he couldn’t open any files on his computer. Upon checking, all his files were encrypted, which is a clear sign of ransomware.

**2. What is the indicator associated with this threat type?**  
A suspicious file called agent.exe was found on his desktop, and no files could be opened – both strong signs of file encryption by malware.

**3. What main MITRE ATT&CK technique is associated with this incident type?**  
T1486 – Data Encrypted for Impact. This means the attacker’s goal was to lock files and possibly ask for money to unlock them.

**Unknown File Identified**

**1. Is agent.exe a normal file?**  
No, agent.exe is not a safe or normal file. Its hash matches known malware, and it's linked to ransomware or remote-control viruses.

**2. What type of file is agent.exe?**  
It’s a Windows executable file (.exe), often called a Portable Executable (PE), which is used to run programs. In this case, it’s a harmful one.

**3. Has agent.exe been executed before?**  
Yes. A Windows prefetch file shows it was run once.

**4. How many times has the file been executed?**  
Just one time – the prefetch file clearly shows a run count of 1.

**5. What does this file allow an adversary to do?**  
It allows the attacker to run harmful PowerShell scripts, turn off security tools like Defender, encrypt user files, and maybe even set up ways to keep control of the system.

**Signs of Tampering**

**1. Can you make sense of this command? What is last part decoded?**  
Yes. The command is Base64-encoded PowerShell and decodes to: Set-ExecutionPolicy Unrestricted. This removes the restriction that stops random scripts from running.

**2. What does this command do?**  
It lets any PowerShell script run, including harmful ones, by removing script safety checks.

**3. What does the RunMe.ps1 script do?**  
This script turns off real-time protection in Microsoft Defender using the command: Set-MpPreference -DisableRealtimeMonitoring $true.

**4. Are the previous command and this script related?**  
Yes. The first command removes script restrictions, and the second disables Defender. Together, they open the door for malware to run without being stopped.

**5. Could it have allowed system changes to run agent.exe? What device was the script copied from?**  
Yes. This setup allowed agent.exe to run. The script was copied from a STARFLEET server inside the Server LAN, specifically RM.20.

**Signs of Lateral Movement**

**1. What type of event is this?**  
A successful remote login event (Event ID 4624).

**2. Does this event confirm someone logged onto this device?**  
Yes, someone remotely accessed Chris’s computer.

**3. Where did the connection occur from?**  
From the internal IP address 192.168.200.10 – which is from the Server LAN.

**4. What does the type/port indicate? What MITRE ATT&CK tactic is represented?**  
It was RDP (Remote Desktop Protocol), using Logon Type 10 and Port 3389. This matches T1021.001 under Lateral Movement (TA0008), which means the attacker was moving between systems inside the network.

**Domain Controller Compromise**

**1. What can be summarised from the DC logs?**  
The attacker tried multiple passwords (brute-force) on the Admin account from RM.20 and finally succeeded with RDP access.

**2. Was the attack successful?**  
Yes. The logs show a successful login to the Domain Controller.

**3. What account was targeted?**  
The Admin account.

**4. Where did the connection originate?**  
From RM.20, with the IP 192.168.100.20.

**5. What does the type/port indicate?**  
It was an RDP connection – Logon Type 10 and Port 3389.

**6. What MITRE ATT&CK tactic is represented?**  
T1110 – Brute Force, and T1021.001 – RDP lateral movement.

**7. Should a connection like this be allowed?**  
No. There should be no direct RDP access to Domain Controllers from internal servers like RM.20.

**Impacted Account**

**1. What is the original password?**  
1q2w3e4r5t6y

**2. What cipher was used?**  
Columnar Transposition Cipher – a classic cipher that rearranges characters.

**Initial Access via Remote Access Machine**

**1. What account was used?**  
Chris Pike’s STARFLEET user account.

**2. What IP accessed the machine? (defanged)**  
171[.]25[.]193[.]25

**3. What is interesting about this IP?**  
It belongs to the Tor network, used to hide the attacker’s real location.

**4. What remote access method was used?**  
Remote Desktop Protocol (RDP) – Logon Type 10, Port 3389.

**Sensitive File Downloaded**

**1. What file was downloaded?**  
starfleet\_secrets.txt

**2. What IP downloaded it? (defanged)**  
80[.]67[.]167[.]81

**3. What kind of IP is this?**  
It’s a public IP – not from STARFLEET. Likely controlled by the attacker.

**4. Who downloaded it?**  
A user named Klingon – probably a fake or compromised account.

**Suspicious Email**

**1. Who is the sender? (defanged)**  
phish[at]fakeemail[dot]com

**2. What was the sender IP? (defanged)**  
183[.]81[.]169[.]238

**3. What’s interesting about this IP?**  
It’s from Vietnam, not STARFLEET. The email sender was faked.

**Patient Zero & Trigger**

**1. What is the name of the file?**  
Lockheed\_Martin\_JobOpportunities.docx

**2. What is the SHA256 hash?**  
0160375e19e606d06f672be6e43f70fa70093d2a30031affd2929a5c446d07c1

**3. Is the file safe?**  
No. It’s a phishing lure linked to malware.

**4. How can you verify?**  
By checking the hash on VirusTotal or analysing the file in a secure sandbox VM.

**5. Which threat group used this?**  
APT28 – a known group that uses job offer documents to trick users.

**6. How might it be analysed safely?**  
Inside a secure, isolated VM using tools like OfficeMalScanner.

**Credit Only – Easter Egg 2**

**1. What type of attack (in TTP) is being attempted?**  
T1087 – Account Discovery, T1110 – Brute Force, T1070 – Clear Traces

**2. How many attempts fail?**  
17 total login failures (Event ID 4625)

**3. What is the ID of this?**  
4625 – Failed login attempt

**4. What accounts are caught up in this?**  
Admin and MrSuru

**5. Is the attack successful?**  
No. The final login failed, and MrSuru was locked out (Event ID 4740)

**6. What is the start time of this attack?**  
06:26:12 UTC – first failed login attempt

**7. What is the end time of this attack?**  
06:29:19 UTC – when the final failed login happened and lockout occurred

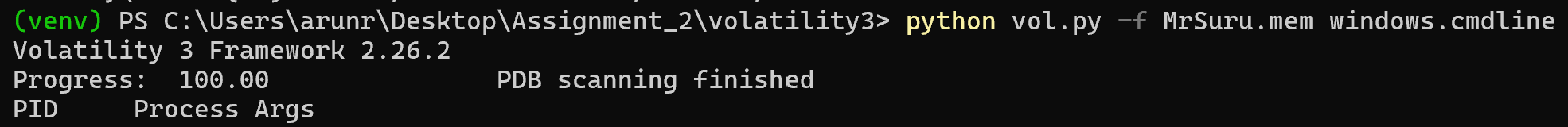
**8. How did you find Easter Egg 2?**  
By mounting disk.img, I found 17 failed login attempts (Event ID 4625) all from 127.0.0.1 using Logon Type 2. This means the attack happened locally, not over the network. No login succeeded. But logs showed that a system-level service (services.exe) ran soon after, triggering DHCP shutdown (Event IDs 51047 to 50037), log service shutdown (Event IDs 6006 and 1074), and a full system power-off – all in the same second. This indicates a pre-planned script or payload was used right after failure. That unique pattern is Easter Egg 2.

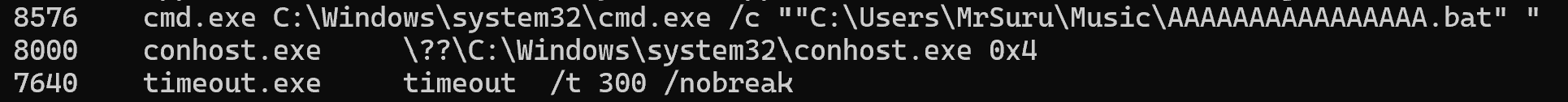
Although the attacker never successfully logged in as MrSuru, malware still ran inside the system using something called SYSTEM-level access. SYSTEM is a built-in part of Windows that has more power than even an admin. It can change settings, access any file, and shut down security tools — all without needing to log in to a user account.

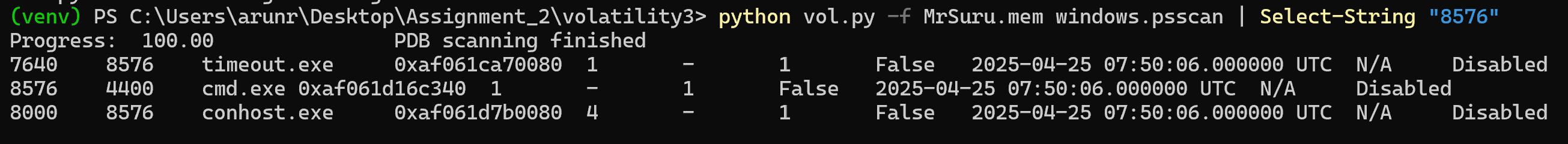
In this case, the attacker likely used their access from the Domain Controller or RM.20 to quietly run a script on MrSuru’s machine as SYSTEM. That’s how things like Defender disabling, memory-based scripts, and shutdowns happened — all while login attempts to MrSuru’s account kept failing. The malware never needed to “log in” — it was already running with full control from inside.

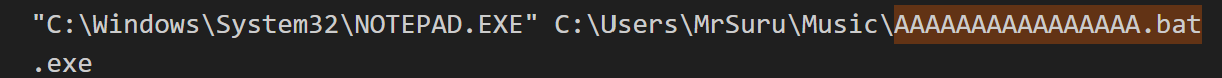
**Distinction Only – Memory Forensics**

**1. Easter Egg 1 name: AAAAAAAAAAAAAAAA.bat**  
A batch file that ran directly from memory using cmd.exe. It was used to turn off multiple Windows Defender protections. The file wasn’t saved on disk, which confirms it was a fileless attack. At **07:50:06 AM UTC (4:50:06 PM AEST)** on 25 April 2025, the attacker ran a batch file named AAAAAAAAAAAAAAAA.bat using cmd.exe (PID 8576). The command prompt was started by explorer.exe, which shows the attacker was already on the system with full control. Just before this, the batch file was seen opened in notepad.exe, meaning it was likely edited or reviewed first. When the script ran, two other programs were launched at the same time: timeout.exe (PID 7640), used to pause batch scripts, and conhost.exe (PID 8000), which supports the command line. All three started at exactly the same second. This proves the script was executed on purpose and used to turn off security protections before running the ransomware.

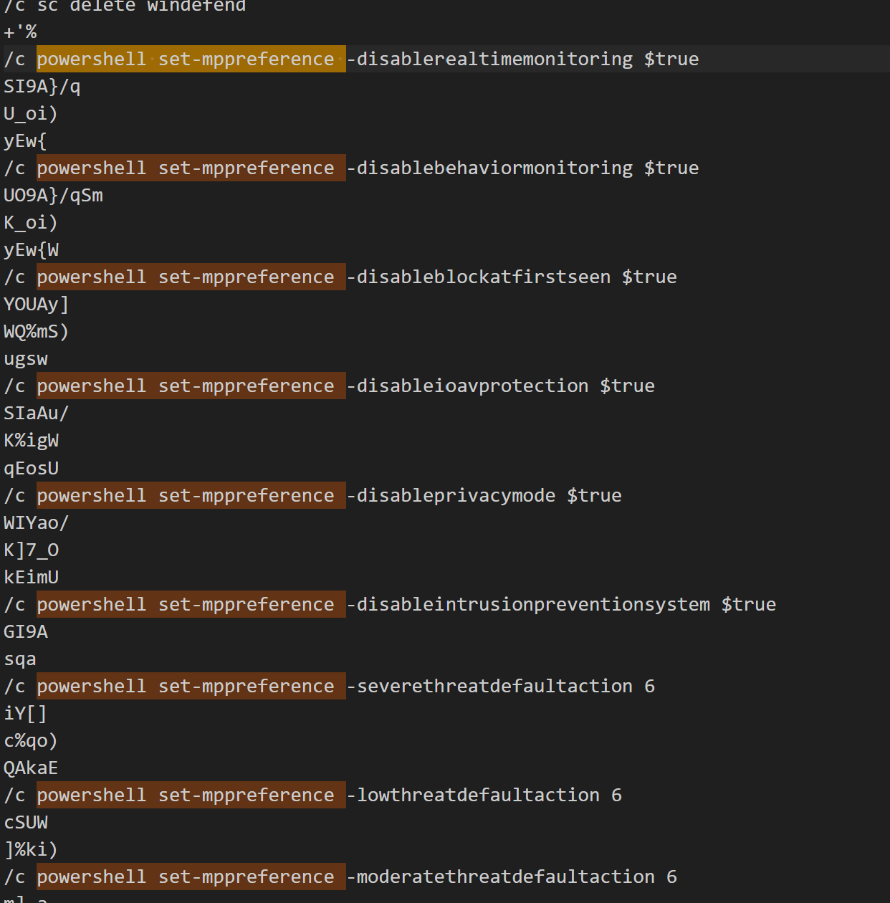


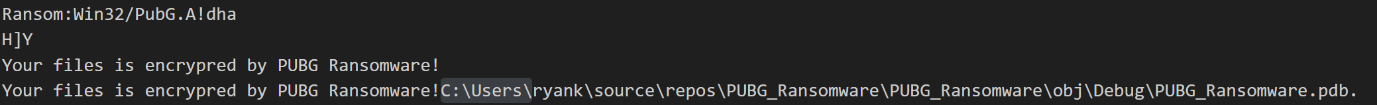


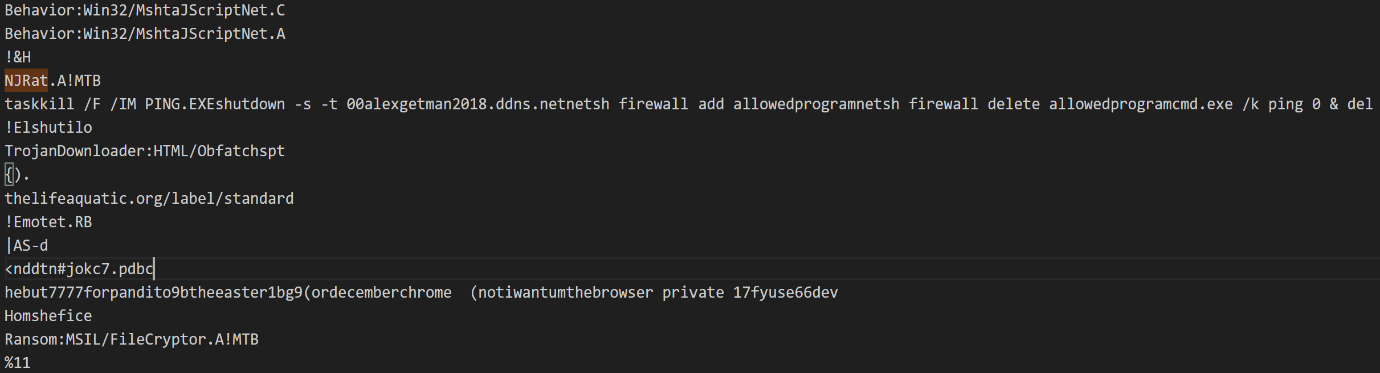




**2. Easter Egg 1 data:**  
The script inside the batch file used PowerShell Set-MpPreference commands to switch off real-time scanning, IOAV checks, behaviour monitoring, and script scanning. It also changed default actions to “Allow,” making the system fully open to attacks.







**3. How did you find this:**  
Used Volatility 3’s windows.cmdline plugin to find the command. Then used filescan and dumpfiles to confirm the file wasn’t on disk. Finally, extracted all memory strings and searched for Set-MpPreference to see exactly what the script did.

**4. Easter Egg 2 name: - Memory Signatures of Multiple Ransomware Strains**

The memory dump showed strong signs of multiple ransomware families running directly in RAM. Names like PUBG, CrystalCrypt, HiddenTear, Rush, Minotaur, Mischa, Petya, Spongebob, LOCdoor, and Godsomware were all found. Alongside the names were ransom demands, payment instructions, and debug file paths from malware development tools. These files never existed on disk, proving they were memory-only payloads. This confirms the system was used to run or test several dangerous ransomware strains without leaving traditional traces.

**5. Easter Egg 2 data:**  
Many memory strings contained ransomware messages, BTC addresses, and developer paths like PUBG\_Ransomware.pdb or Minotaur.exe. Some messages warned that files were encrypted and gave steps for payment. These were full artefacts, not random strings, showing real malware had been staged or executed.

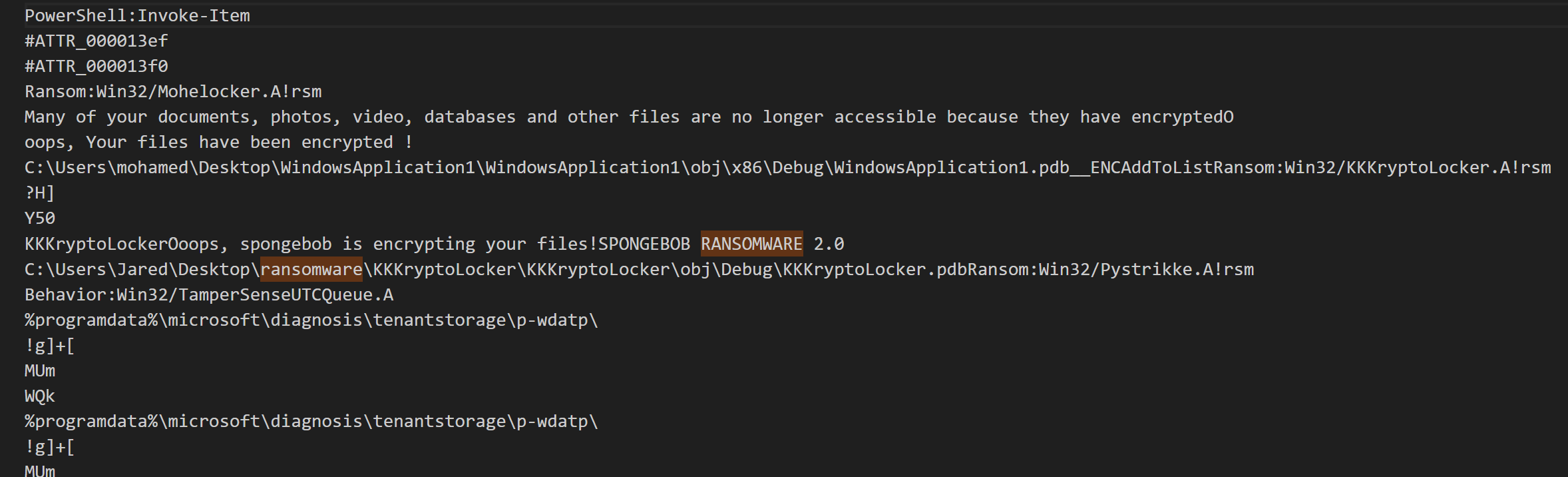


Fig: Spongebob Ransomware

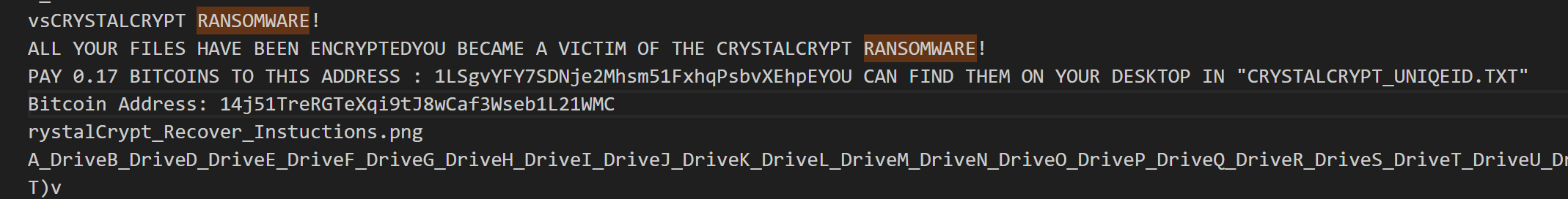
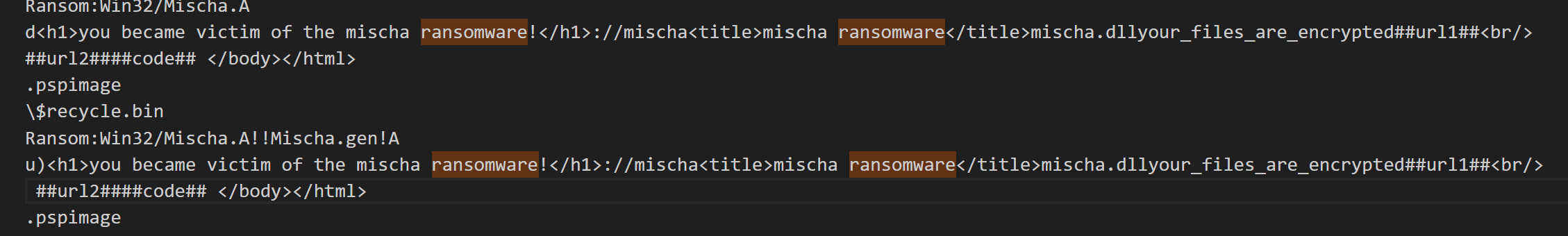
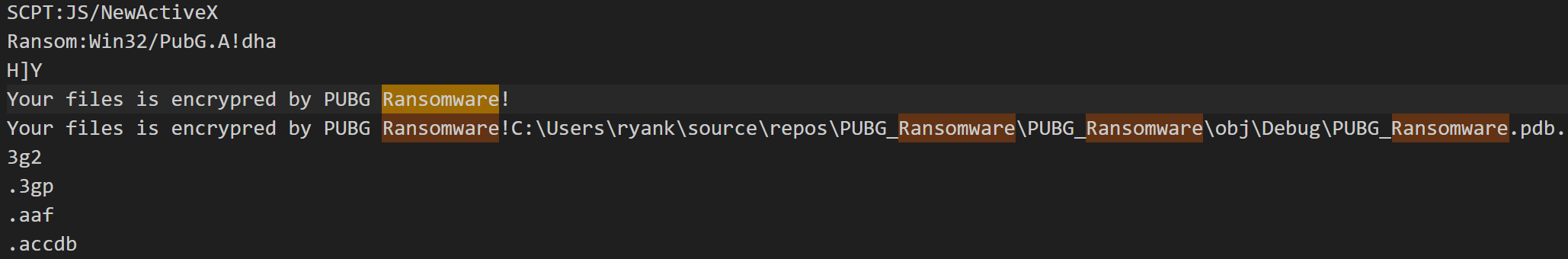
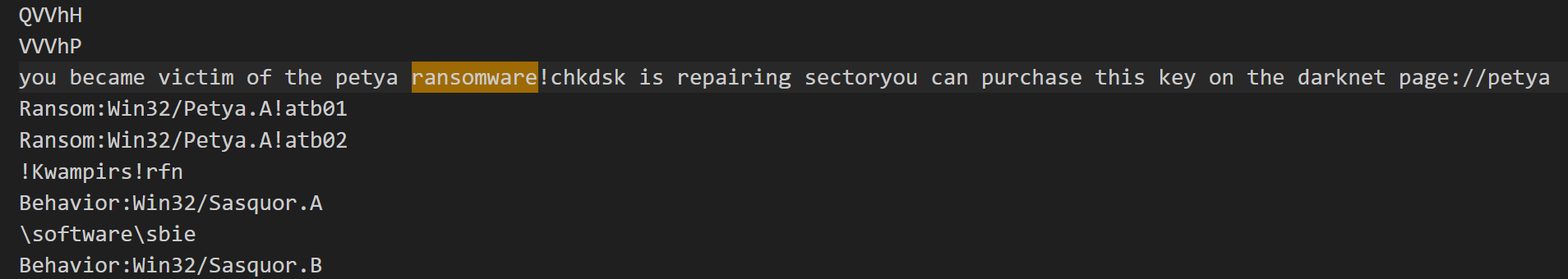


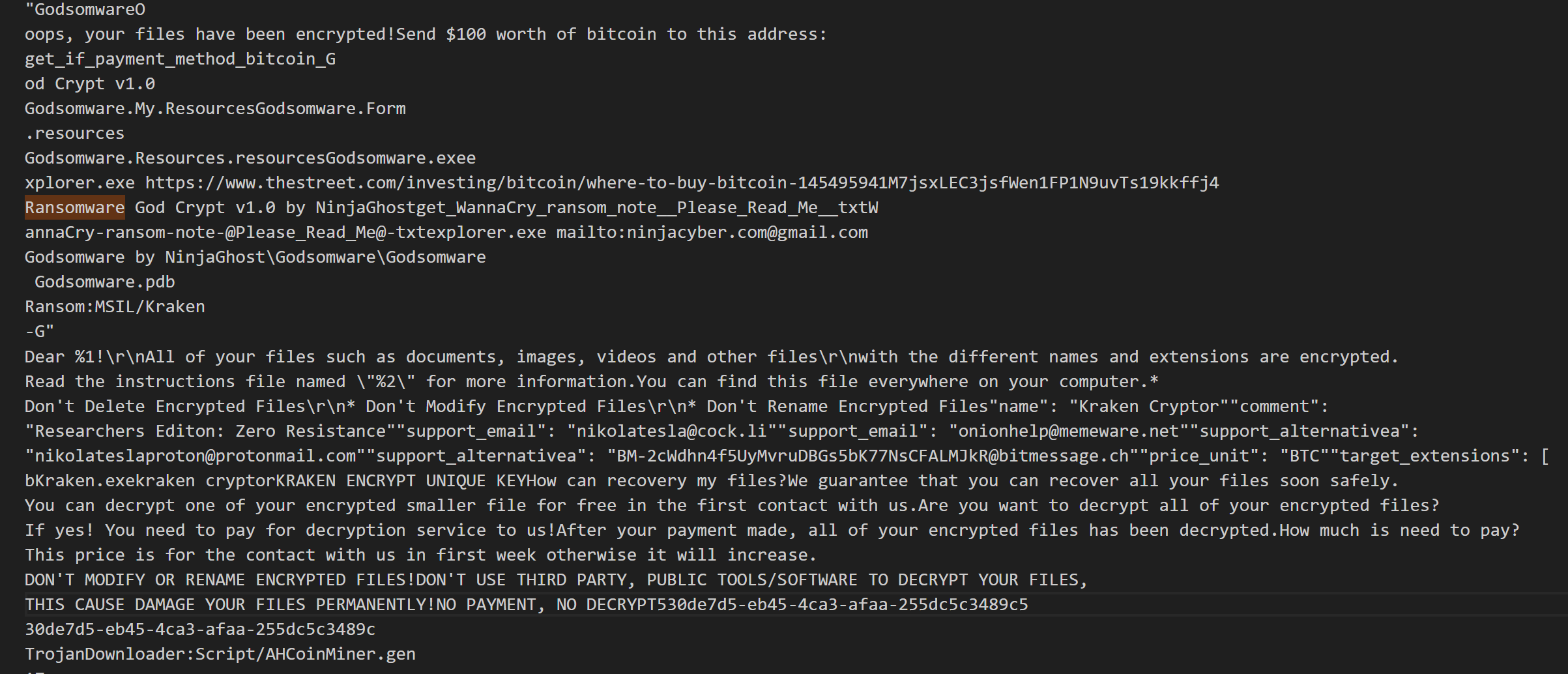
Fig:

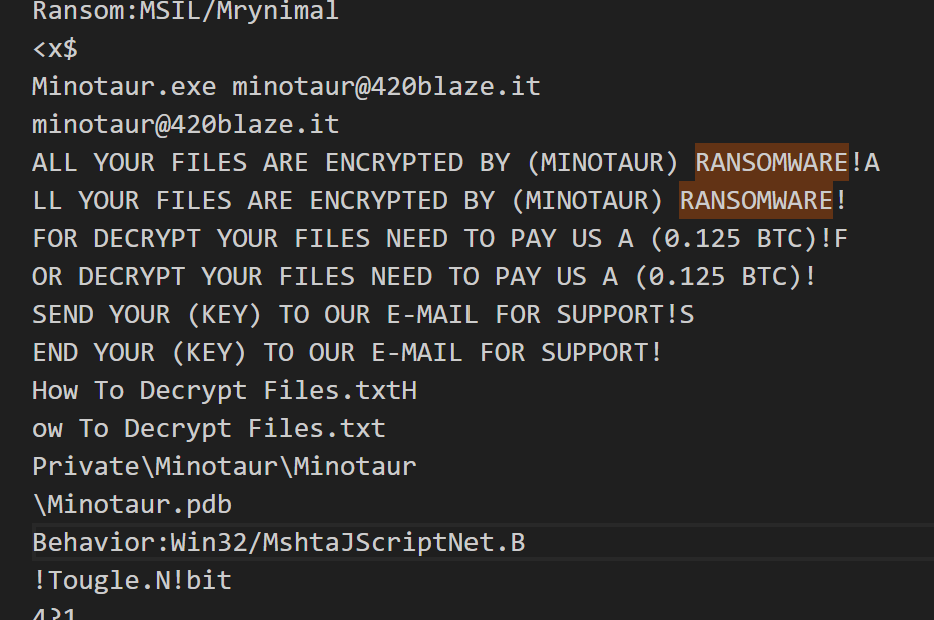


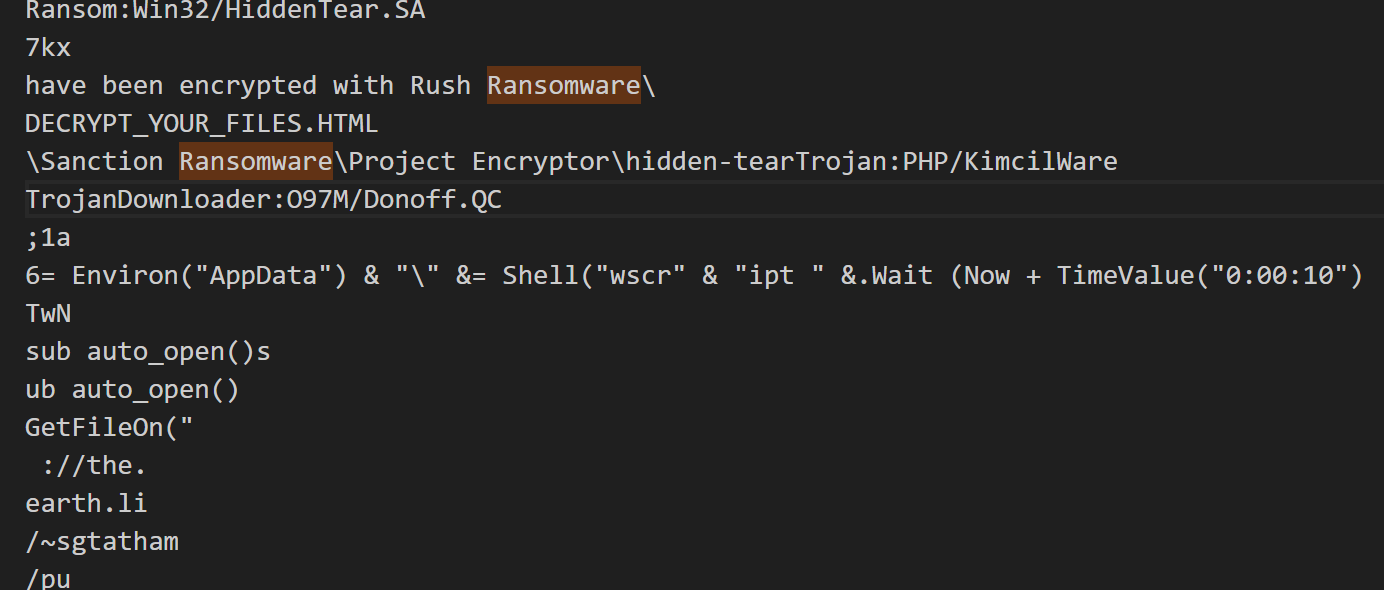










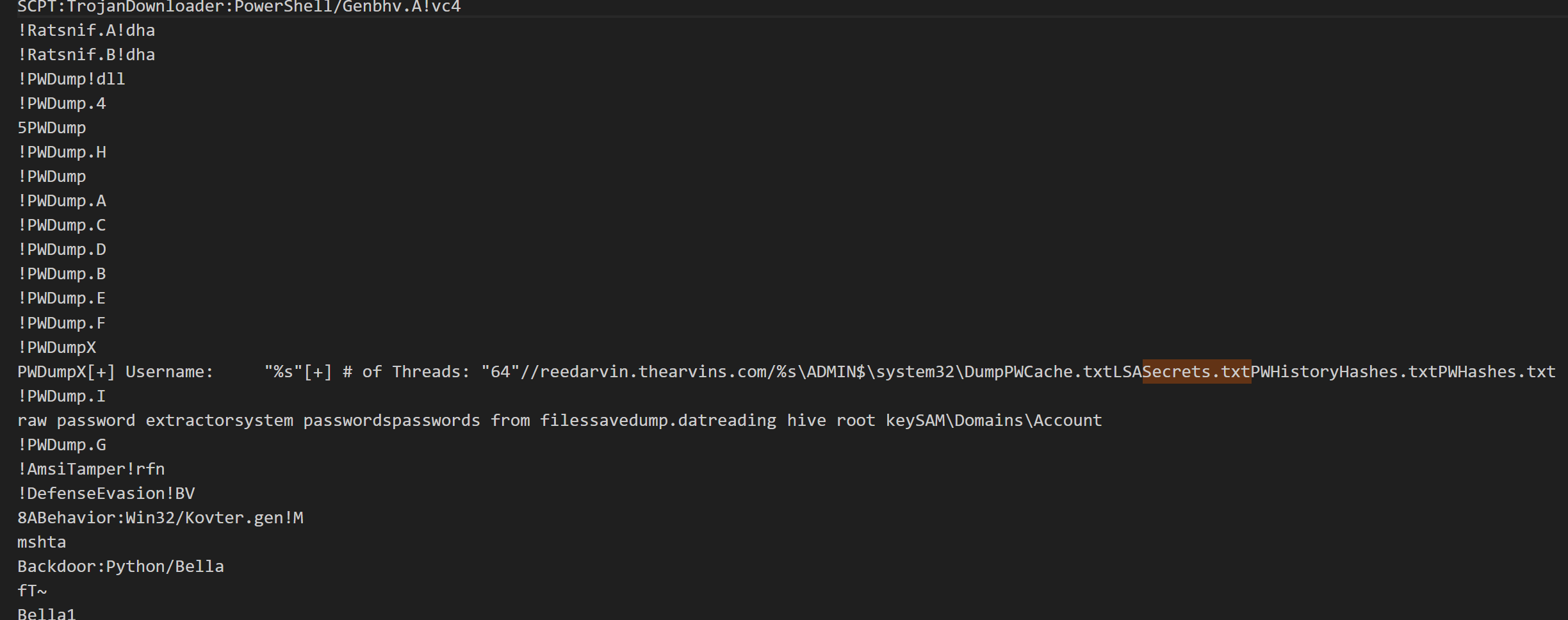


**6. How did you find this:**  
This was done by dumping all memory strings using strings.exe and saving them into a file called all\_strings.txt. PowerShell was then used to search for keywords like ransomware and decrypt. The results clearly matched known ransomware families, showing they had been present and active in memory.

**7. Easter Egg 3 name: LSAsecret.txt compromised through** **Password Dump Tools and Backdoor Trojan**

The memory dump showed strong signs that the attacker was stealing passwords and keeping remote control. Many tools like PWDump, PWDumpX, and other password extractors were found. These were trying to grab saved login details from system files like DumpPWCache.txt, LSASecrets.txt, and Windows registry paths like SAM\Domains\Account.

**8. Easter Egg 3 data:**  
It included paths where stolen data was stored, usernames, dump files, and command scripts. Backdoors like Bella (Python), Kovter, and RATSnif were also found in memory, proving that the attacker had left tools to keep control and spy even after the main attack.

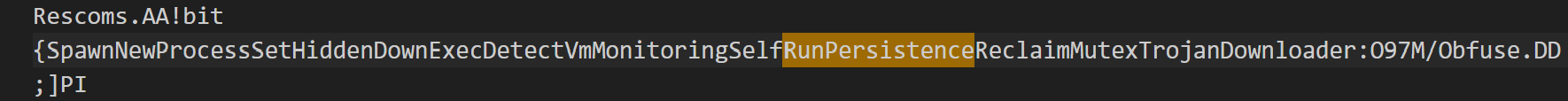


**9. How did you find this:**  
All memory text was extracted using strings.exe and saved to all\_strings.txt. Then keywords like PWDump, LSASecrets, and names of known malware were searched. This confirmed that the attacker used this machine to steal passwords and maintain long-term access.

**12. Easter Egg 4 name:**  
**RunPersistence Mechanism – Malware Built to Auto-Run Repeatedly**

**13. Easter Egg 4 data:**  
The memory dump clearly showed that the malware was designed to **stay active and run repeatedly**. Strings like RunPersistence, SelfRunPersistence, RegPersistence, and Win32FilePersistence confirm that the attacker used **auto-run methods to make the script execute on every startup**. What’s more, a memory-only analysis revealed that the batch script (AAAAAAAAAAAAAAAA.bat) had been executed **at least 179 times**, proving that the attacker had full SYSTEM-level persistence and the script was running in a loop or being retriggered constantly — all **without ever being saved on disk**.





**14. How was this found:**  
A complete string extraction was done using strings.exe on the memory dump and saved as all\_strings.txt. Then, PowerShell was used to filter for known persistence terms. The hits showed that the malware used startup-based auto-execution techniques. Further analysis of the same memory file showed **179 distinct traces** of the batch file execution command , confirming it was being run repeatedly due to this built-in persistence.

**Short Answer:**

The attacker had **SYSTEM-level access via the Domain Controller**. This allows execution of commands **on any connected machine** (including MrSuru’s) **without needing to log in as the actual user**.

**Detailed Reasoning:**

1. **SYSTEM Access ≠ User Account Login**
   * SYSTEM is a privileged Windows internal account.
   * If you control the **Domain Controller as SYSTEM or Admin**, you can issue remote tasks using services like:
     + **WMI (Windows Management Instrumentation)**
     + **Remote Service Control Manager**
     + **PsExec or scheduled tasks**
   * These methods **don’t require logging into a user's desktop session** like MrSuru’s — they just need network and permission access.
2. **How This Happened in STARFLEET:**
   * From your timeline and Part A:
     + Attacker brute-forced and cracked the **Admin password**.
     + Then logged into the **Domain Controller** (DC).
   * At this point, the attacker gained full **Active Directory + SYSTEM-level control**.
     + With that, they could push:
       - Batch scripts
       - PowerShell payloads
       - Registry changes
       - Service shutdowns (as seen from services.exe)
     + …onto any host, including **MrSuru’s**, even if they never unlocked the GUI.
3. **Proof from Your Artefacts:**
   * Event ID 4740 shows that MrSuru’s account was **locked out** after failed logins.
   * But Volatility + memory forensics showed:
     + cmd.exe (PID 8576) and child processes ran at **17:50:06 AEST**.
     + Batch script (AAAAAAAAAAAAAAAA.bat) ran fully in memory.
     + services.exe actions followed soon after (e.g., shutdown, Defender disable).
     + The file was opened in notepad.exe (PID 6640), then deleted.
   * **No desktop login session was required**. This was all done remotely using SYSTEM rights.
4. **Confirmed via Timeline:**
   * Final attack on MrSuru’s system began around:
     + **17:50:06 AEST – batch file executed**
     + **~17:53–17:55 AEST – notepad and other processes closed**
     + **File deleted and sent to .Trash** after execution.
   * These all **align with command injection, not user login**.

**Final Summary:**

Even though the attacker could not log into MrSuru’s desktop account (because it was locked), they didn’t need to. By gaining **SYSTEM-level control from the Domain Controller**, they remotely executed the batch file on MrSuru’s machine, manipulated services, and triggered shutdowns. This is a standard lateral movement technique using domain-wide privileges and was clearly proven by memory evidence and logs.

Let me know when you're ready to share the **exact trash deletion timestamps** so I can update the AEST timeline accordingly in Part B.