American Journal of Mathematical and Computer Applications

Research Article

Volume 1 | Issue 1

# Solar Winds Compromise Malware Analysis

**Syed Ahmad Maaz**

Department of Mathematics, France.

***Corresponding author:** Syed Ahmad Maaz, Department of Mathematics, France.

**Submitted:** 12 Jan 2025 **Accepted:** 20 Jan 2025 **Published:** 25 Jan 2025

***Citation: Maaz, S A (2025) Solar Winds Compromise Malware Analysis. American J of Math and Comput Applications. 1(1), 01-13.***

## *Abstract*

*The SolarWinds compromise was one of the most significant cyberattacks of the 21st century, not because it breached a single organization, but because it triggered a much larger supply chain incident that affected thousands of organizations globally. Attributed to the Advanced Persistent Threat (APT29) threat group, this attack leveraged sophisticated malware tools to infiltrate high-profile entities. This paper provides a detailed analysis of the four main malware variants used in the attack: SIBOT, Raindrop, GoldMax, and GoldFinder. A controlled environment was established to study the behavior of each malware, focusing on their techniques for achieving persistence, lateral movement, and evading detection. The findings contribute to enhancing threat intelligence and offer insights for improving defenses against similar attacks, highlighting the importance of taking early steps to detect and prevent advanced persistent threats.*

## 1. Introduction

Solarwinds is a software company which provides management tools for network and infrastructure monitoring. It also provides technical services to thousands of organizations around the world. Solarwinds provides network monitoring services like Solarwinds Network Performance Monitor (NPM) that allow organizations to gain real-time insights into the health, performance, and availability of their IT net- works. By ensuring network stability and optimal performance, SolarWinds empowers organizations to deliver uninterrupted services to their users and customers [1].

SolarWinds offers a range of features like intrusion detection, vulnerability assessment, log analysis, and security event correlation to help organizations implement a security infrastructure. SolarWinds helps organizations fortify their cloud deployments, identify vulnerabilities, and safeguard their sensitive data [2].

SolarWinds Orion is a powerful and widely used IT management and monitoring platform that helps organizations effectively manage their network infrastructure, systems, and applications. Designed for both small businesses and large enterprises, Orion provides a comprehensive suite of tools that streamline various IT tasks such as performance monitoring,

configuration management, and real-time issue detection. By collecting and analyzing data from critical components within an organization's IT ecosystem, Orion enables IT administrators to maintain optimal performance across servers, databases, applications, and network devices.

Orion's scalability is one of its key selling points, as it can be tailored to meet the unique needs of diverse environments. Whether it's a small company with a few servers or a large enterprise with complex network architecture, Orion's architecture ensures that it can scale accordingly, providing the same level of in-depth monitoring and control. The platform supports a wide range of monitoring features, which include network traffic analysis, server and application monitoring, database monitoring, and IT asset management. These features give IT teams the ability to track system performance, spot potential problems before they affect operations, and improve the overall health and efficiency of their infrastructure.

A distinguishing factor of SolarWinds Orion is its user-friendly interface and customizable dashboards, which allow IT staff to quickly visualize performance metrics and key data across the entire network. The platform also includes robust reporting capabilities, providing detailed insights into system health, usage trends, and potential security vulnerabilities. These reporting tools make it easier for organizations to make informed decisions, implement performance improvements, and troubleshoot issues effectively.

Another feature of Orion is its integration capabilities, enabling it to unify monitoring tools from various vendors into a single interface. This centralized view allows IT teams to manage all aspects of their infrastructure from one place, reducing the complexity of managing multiple, disparate tools. Orion can integrate with other SolarWinds products and third- party applications, streamlining workflows and ensuring a cohesive approach to network and system management. This integration fosters a proactive approach, allowing IT teams to stay ahead of potential disruptions or performance issues [3].

Given its critical role in an organization's IT ecosystem, SolarWinds Orion is often deeply integrated into the infrastructure of many large organizations, including both public and private sectors. This extensive integration makes it an attractive target for cyberattacks, as compromising Orion provides attackers with access to the networks and systems of numerous high-profile organizations.

## 2. Related Works

Nair et al. [4] provide a detailed guide on static malware analysis. This paper outlines the steps used for statically dissecting a malware. The research paper emphasizes the use of static analysis tools like PEStudio, PEview, strings, UPX (Ultimate Packer for Executables) for static malware analysis and for unpacking the packed malwares.

Guven et al. [5] established a dynamic malware analysis environment and prepared a dataset containing malicious and benign network traffic. They extracted 39 features from the traffic data and used Random Forest and deep learning algorithms to classify the traffic as either malicious or safe. Their comparison of the algorithms determined which approach achieved higher accuracy in identifying malware under such conditions.

Mahmoud et al. [6] proposed a dynamic malware analysis framework that integrates Sysmon and the Elastic Stack to create a custom sandbox environment. Using a sample of 2,800 malware instances from VXUnderground, the research demonstrated the effectiveness of Sysmon and ELK integration for analyzing malware behavior.

Malik et al. [7] introduced a framework for malware analysis using a Modern Honey Network (MHN) deployed on a cloud machine. The environment employs a honeypot and malware sensor to detect Portable Executable (PE) binaries, extract their MD5 hashes, and utilize the VirusTotal API for further malware analysis.

Dutta et al. [8] presented a survey on malware detection methods, emphasizing the advantages and disadvantages of machine learning techniques. The paper transitions from traditional signature-based and behavior-based detection methods to heuristic-based detection, which incorporates machine learning, highlighting its efficacy and limitations in addressing evolving malware threats.

Hammi et al. [9] explored malware detection by analyzing Windows system calls. The study utilized algorithms such as ensemble learning, k-Nearest Neighbors (k-NN), Naıve Bayes, Random Forest, and Decision Tree, training them on malware samples using custom Python scripts. Comparative analysis identified the most effective algorithm for detecting malware based on performance metrics.

Srinivas et al. [10] introduced a methodology for malware detection using YARA rules. It focused on creating signature-based rules for identifying and classifying malware families. By applying these rules to datasets, the study demonstrated improvements in detection accuracy and the ability to address variations in malware characteristics.

Yousuf et al. [11] proposed a static analysis-based approach for detecting malware by extracting multiple features from PE binaries, such as the DOS header and Windows APIs. The study concluded that feature extraction from PE binaries alone is insufficient for reliable malware detection.

## 3. Environment Setup

The reverse engineering of the malware used in the SolarWinds compromise is completed in a controlled environment which was created using two virtual machines (VMs) configured in an isolated host-only adapter network to ensure security and prevent unintended external communication.

### 3.1 Ubuntu VM(Sniffer and Gateway)

The Ubuntu VM was configured to act as a gateway for the Windows VM, ensuring that all network traffic originating from the Windows VM passed through it. The configuration allowed for comprehensive monitoring and analysis of the mal- ware's network activity. Key tools and configurations included:

- INetSim: Simulated internet services to observe the malware's network activity in a controlled environment. It provided essential responses to network requests, such as DNS queries, HTTP requests, and other protocols, enabling detailed analysis of the malware's behavior.

- Wireshark: A network protocol analyzer used to capture and analyze the traffic generated by the malware. This tool allowed for in-depth inspection of communication patterns and potential indicators of compromise.

### 3.2 Windows VM (Malware Analysis)

The Windows VM served as the primary environment for conducting malware analysis. The environment was equipped with specialized tools tailored for static and dynamic analysis, including:

- FLARE VM: A specialized malware analysis toolkit developed by FireEye. It includes a comprehensive suite of tools for malware analysis, such as debuggers, decompilers, and disassemblers, making it suitable for analyzing complex malware behaviors.

### 3.3 Network Configuration

The network configurations used for securely analyzing the malware within a controlled environment are as follows:

- Isolated Network: Ensured that malware activity was contained within the analysis environment, preventing accidental spread of the malware or its payloads to other systems.

- Gateway Configuration: The Ubuntu VM served as the gateway for the Windows VM, allowing it to monitor and control all traffic originating from the Windows VM. This setup was essential for capturing and analyzing the malware's network interactions.

The environment ensured the safe and effective analysis of malware samples such as SIBOT, Raindrop, GoldMax, and GoldFinder. Figure 1 shows the Environment setup of the malware analysis lab.

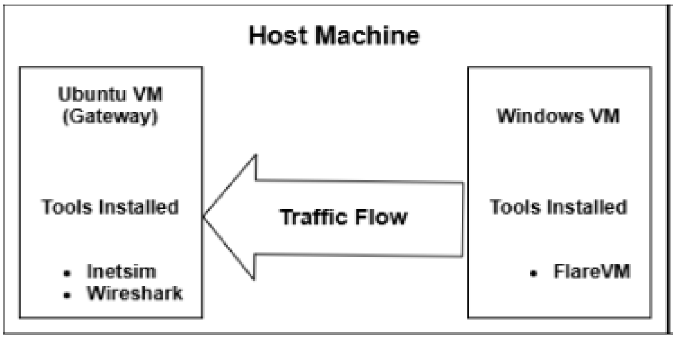

**Figure 1:** Malware Analysis Lab Setup.

## 4. Solarwinds Compromise Overview

The SolarWinds compromise was disclosed in December 2020. It is one of the most significant cyberattacks in recent history. It targeted SolarWinds' Orion platform which is widely used in IT management and monitoring solution deployed across public and private sectors globally. The attackers exploited the trusted software update mechanism of the Orion platform to distribute malicious updates containing backdoors, thereby infiltrating numerous organizations [12].

### 4.1 Attack Overview

The compromise is attributed to the Advanced Persistent Threat (APT) group APT29, also known as Cozy Bear, which is linked to the Russian Foreign Intelligence Service (SVR RF). The attackers gained access to thousands of organizations by compromising the orion software update mechanism. This access enabled the execution of highly sophisticated post- exploitation activities, including data exfiltration, lateral movement, and installation of additional malware [4].

The attack unfolded as follows:

- Supply Chain Compromise: The attackers injected malicious code into Orion updates, which SolarWinds distributed to customers through legitimate channels.

- Post-Compromise Operations: The attackers used additional malware to ensure persistence, evade detection, and achieve their operational goals in targeted environments.

### 4.2 Malware Families Analyzed

This research focuses on the analysis of four key malware families deployed during the SolarWinds compromise:

- SIBOT: A VBScript-based malware designed to establish persistence and download additional payloads.

- Raindrop: A malware backdoor loader used to deploy Cobalt Strike beacons for post-exploitation activities.

- GoldMax: A stealthy backdoor used for covert communication with command-and-control (C2) servers.

- GoldFinder: A reconnaissance tool designed to trace network routes to the C2 infrastructure.

Each of these malware families played a critical role in the attack, showcasing advanced techniques for persistence, evasion, and lateral movement.

### 4.3 Impact

The SolarWinds compromise affected over 18,000 organizations globally. High-profile victims included government agencies, critical infrastructure entities, and leading private- sector firms. The attackers only pursued specific high-value targets out of the affected organizations [13].

The paper provides a detailed analysis of the SIBOT, Raindrop, GoldMax, and GoldFinder malware families. The focus will be on their behavior, persistence mechanisms, and techniques for evading detection. The findings highlight critical vulnerabilities in supply chain security and emphasize the need for robust defenses against such advanced threats.

## 5. Sibot Malware

The section provides a comprehensive analysis of the SIBOT malware, its functionality, and its role in the SolarWinds compromise. We examine its methods of achieving persistence, downloading additional payloads, and evading detection, highlighting its impact on the broader attack strategy.

### 5.1 Introduction

SIBOT is a VBScript-based malware deployed during the SolarWinds compromise. Its primary role was to establish persistence on compromised systems and serve as a down loader for additional payloads. The simplicity and modularity of SIBOT underscore its design for stealth and efficiency in carrying out its tasks.

SIBOT relied on lightweight scripts, making it easier to evade detection mechanisms. By embedding itself into the Windows Task Scheduler, it ensured that its operations could continue across system reboots, allowing the attackers to maintain a foothold in compromised environments over extended periods.

SIBOT's functionality was often tailored to the target environment. It leveraged encrypted communications and proxy configurations to securely download and execute additional malware payloads.

The analysis explores SIBOT's techniques for persistence, payload retrieval, and detection evasion, shedding light on its pivotal role in the broader SolarWinds compromise.

### 5.2 Static Analysis

The static analysis of the SIBOT malware began with the identification of the programming language used to write the script. Initial inspection showed that the malware was written in VBScript, a scripting language commonly used for automation tasks within Windows environments. The script was heavily obfuscated, a technique often employed by malware authors to evade detection. The process of static analysis is stated:

- Script Examination: The script was opened in Sublime Text, which provided an easy-to-read interface to examine the code. The first step in the process was to identify the most frequently invoked function within the script. Once the key function was identified, the next step was to deobfuscate the code by replacing the obfuscated function and variable names with meaningful, descriptive identifiers. This process allowed for a clearer understanding of the script's logic and purpose. Then deobfuscated function was responsible for decoding the strings in the script.

- Decoding Obfuscated Strings: The analyst was able to reverse the obfuscation and decode several key strings used within the malware. The function was responsible for decoding various obfuscated strings, including URLs, command parameters, and other critical components. The analyst was able to reveal the original unencrypted values hidden within the script. The decoding process significantly improved the legibility of the malware code and provided a more comprehensive view of its actions.

- Information Gathering: The next step was to examine the behavior of the script. The analysis revealed that the malware's first task was to gather information about the compromised system. It begins by attempting to retrieve the Globally Unique Identifier (GUID) of the system's LAN connection. If the system does not have a GUID assigned, the script takes the initiative to generate and assign a new GUID to the system. This action is crucial as it allows the malware to uniquely identify the infected machine, which could be leveraged later in its communication with command-and-control (C2) servers or for tracking purposes.

- Network Configuration: The malware checks the system's network configuration by querying the Windows registry for proxy settings. The script looks for registry entries that indicate whether the machine is configured to use a proxy server for network traffic. If the system is not using a proxy server, the script terminates its execution, effectively preventing further actions. This behavior suggests that the malware is designed to operate in environments where network traffic can be filtered through a proxy, which may provide enhanced security or concealment for the attacker. If a proxy server is present, the malware continues its execution to carry out further actions.

- GET Request Construction: The script constructs a GET request to a remote URL. The GET request is routed through the proxy server, so that all traffic between the infected machine and the remote server is encrypted and difficult to detect. The use of encryption for communication between the malware and its C2 server is likely to avoid detection by network monitoring systems and security solutions such as firewalls and intrusion detection systems (IDS). The malware uses this secure connection to download its payload from the remote server.

- Payload Decryption: The malware continues to decrypt the downloaded payload. The payload is encrypted, and the script contains a function that is specifically designed to decrypt the payload. The script writes the payload to the directory of the Windows operating system. The payload is saved as a .sys file.

- Payload Execution: The malware executes the payload using rundll32.exe. The malware configures the payload to run as a scheduled task. This ensures that the malicious code is executed automatically whenever the system is restarted, allowing the attackers to maintain foothold on the compromised machine even after a reboot.

- Self-Deletion: The script deletes itself from the system after maintaining persistence. The step is designed to cover the tracks of the malware. By deleting itself, the script makes it more difficult for incident responders or analysts to trace the attack back to its source.

The static analysis of the SIBOT malware revealed the structure of the script designed for stealth and persistence. The script utilized multiple techniques to evade detection, such as obfuscation, encrypted communications, and the use of legitimate Windows utilities to execute the payload. The malware covered its track by establishing persistence through the creation of scheduled tasks and its use of proxy servers for encrypted communication.

### 5.3 Dynamic Analysis

The dynamic analysis of the SIBOT malware confirmed the findings of the static analysis. The findings of dynamic analysis are:

- CONNECT Request: It was observed that the script initiates a CONNECT request to a specified URL. This behavior is consistent with the malware's use of proxies to facilitate encrypted communication for payload re- trieval. The CONNECT request is a feature of HTTP/1.1 used to establish a tunnel to a server, often for secure HTTPS communication through a proxy.

- Execution in Simulated Environment: The analysis was performed in a controlled environment using INetSim, the simulated internet service provided by INetSim responded with a static reply to the CONNECT request. The static reply did not fulfill the malware's requirements for downloading the intended payload. The script terminated its execution because the payload could not be retrieved, effectively halting further actions.

The malware would likely have connected to a command- and-control (C2) server through the proxy to download and execute its payload in a real world scenerio. The controlled environment successfully prevented the download of payload and prevented the risk of getting compromised.

### 5.4 Evasion Techniques

SIBOT malware exhibits a range of sophisticated evasion techniques that are critical to its success in maintaining stealth and avoiding detection in compromised systems. These techniques are evident from both the static and dynamic analyses conducted on the malware.

- Obfuscation: SIBOT's VBScript is heavily obfuscated, making it difficult to analyze and understand its functionality. The malware achieves this by using nonsensical function and variable names and employing functions to encode or obscure key strings and instructions. This obfuscation serves multiple purposes:

– It hinders the efforts of security researchers during static analysis. It complicates the process of extracting meaningful insights without extensive deobfuscation.

– The script relies on encoded strings for crucial operations, and only during runtime are these strings decoded using specific functions within the malware.

• Encrypted Communication: SIBOT leverages proxies to send a CONNECT request to a remote URL, enabling it to establish an encrypted tunnel for payload delivery. By relying on encrypted communication:

– It avoids detection by traditional network monitoring tools that might flag unencrypted data streams.

– It prevents researchers or monitoring systems from easily capturing and analyzing the payload during transit.

- Task Scheduling: SIBOT establishes persistence by creating a scheduled task that executes the downloaded payload using rundll32.exe. This technique ensures the malware can survive system reboots and remain active for extended periods.

- Self-Deletion: After completing its tasks—such as decoding the payload and establishing persistence, the malware deletes its own script. This self-deletion removes traces of its presence.

### 5.5 Indicators of Compromise

Indicators of Compromise (IoCs) play a critical role in detecting and responding to malware infections. The Indicator of Compromise of SIBOT malware are listed.

- URL: The malware establishes a connection to the following URL: sense4baby.com

- IP Address: SIBOT communicates with the following IP address: 185.185.117.15

## 6. Raindrop Malware

This section provides a comprehensive analysis of the Raindrop malware, its functionality, and its role in the SolarWinds compromise. We examine its methods of achieving persistence, downloading additional payloads, and evading detection, highlighting its impact on the broader attack strategy.

### 6.1 Introduction

Raindrop is a backdoor loader that employs advanced techniques to evade detection and execute a Cobalt Strike beacon payload as shell-code. In most of the variants it masquerades as some legitimate application using their recompiled source code [14].

To achieve its objective of executing shell-code for post exploitation activities, the malware employs several evasion techniques to remain undetected until its payload is fully executed. These methods include:

- Packing: The payload is packed used a custom packer.
- Encrypted and Compressed Payload: The payload is encrypted and compressed to obfuscate its contents and evade static analysis [6].
- Payload Segmentation: The payload is divided into smaller chunks and loaded into fixed memory locations, complicating detection and analysis.
- Runtime Decryption and Decompression: The payload is decrypted and decompressed dynamically at runtie, ensuring that its malicious content is only revealed during execution.

Variant is similar to its older variant Teardrop which was delivered in first stage of attack by SunBurst backdoor. From research, an overall picture of Raindrop captures the essence of aid in post exploitation activities, including establishing command-and-control (C2) channels and aiding lateral movement with a network.

## 6.2 Static Analysis

This section delves deep into the static analyses of the Raindrop malware. Inspection of code disassembly, imports, exports, metadata, and embedded resources. Most important findings include some from strings, and code disassembly which confirms its malicious nature as listed below:

• To develop an understanding of the nature of executable, analyst used PEStudio tool, to gather information about it. It yields that malware is a TK GUI based PE64 Dynamic Link Library (DLL) with custom packing detected. Here GUI is a way to masquerade behind a legitimate process. Some conclusions from this are:

– Packer involved might alter the malware during each packing operation producing unique binaries that can bypass detection.

– Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP) is often enabled to mimic legitimate software, complicating attempts by analysts to inject shellcode or payloads during debugging. This demonstrates the malware's sophistication and its intent to resist analysis.

- Metadata suggests it has high entropy (7.4). This can mean file might be encrypted using AES since its ran- domness it very high or it might even be compressed. Any or both can be true.
- Version section masquerades fields from legitimate 7- Zip manifest. This can indicate it might be a recompiled source code from 7-Zip with added methodologies to conceal its shell-code. Same manifest string was also found in floss results.
- Exports section, another key finding we came across is, despite it being a DLL, there are no DLLMain function exports. Here there are two main observations:

– While some legitimate DLLs may omit DLLMain, it's absence is more common in malware, particularly if the DLL is intended to hook into existing processes, run in an unconventional way, or to conceal its entry-points to malicious code.

– Malware leverages on its deceptive 7-zip TK MainLoop process as a Main DLL export to conceal it's other DLLMain entry-point.

• Strings analyzed from floss results were significant and can give clues towards its malicious nature:

– Strings and Exports found were beginning with Tk which might indicate TK/Tcl are used for its GUI guise to lay low under the radar. Malware often uses legitimate-looking functions or libraries (like Tk in this case) to blend in or masquerade as something benign.

– Presence of Is Debugger Present might indicate that malware is trying to evade debuggers to make it more difficult to be analysed.

– ADVAPI32.dll import indicate, it might be manipulating registry.

– GetTickCount and Query Performance Counter string are very significant indicators of sources of entropy for a file. They might even be used as a seed/key for encryption algorithms as well as srand random number generators.

– Create Thread, Create Process W, Terminate Process, Get Current Process Id, Get Current Thread Id, Sleep win32 API's string found are indicators that our malware is involved in significant thread and process manipulations.

– Registry entry strings 23170F69-40C1-278A-1000-000100020000 confirms out suspicion of it mas- querading a 7-Zip program.

• For further analyses IDA Pro and Cutter were utilized to get into the flow and working of the file and eventually confirm if it's malicious or not. Some of the observations from disassembled code are as follows:

– UI Event Loop Sleep: At the start, the TK MainLoop malware initiates a recursive sleep cycle lasting 60 seconds This is a major Indicator to evade detection by automated sandboxes and debuggers. This behavior is employed to keep the headless GUI process running and server as a facade for the 7-Zip program.

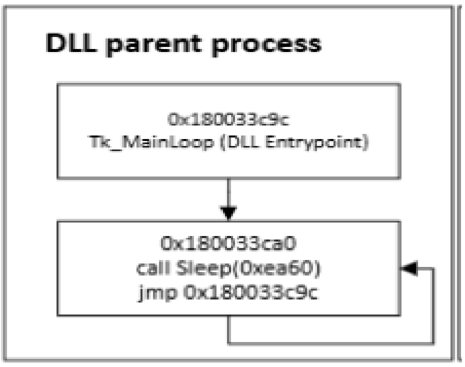

**Figure 2:** TK Infinite Sleep Event Loop

- Polymorphic Behavior: Following thread trail, ”call cs:qword 180056E90”, is highly suspicious. This is an indirect call to the address stored in memory at the location 0x180056E90. Address 0x180056E90 is part of the .data section and is defined as a pointer to a function. The dynamically resolved function pointer could be pointing to malicious code or routines that are injected or altered during execution. Malware is using this technique to avoid signature-based detection or to make analysis more difficult.

- Decryption: Thread does extensive xor (A fundamental operation in many cryptographic algorithms, used for bitwise mixing of plaintext), bit shifts (Often used in cryptographic routines to extract or isolate specific bits in a word.) in each iteration. This might suggest possible chaining decryption methodology. Block-Like Memory Usage (e.g. rsp+0F8h+Block) seems to hold a processed block of data, which is likely reused across iterations. This is also consistent with cryptographic algorithms, where temporary buffers are often used for intermediate results.

- Decompression: The combination of VirtualAlloc, data manipulation (memset, sub * calls), and Virtu- alFree. Sub functions inside the parent are also involved in heavy memory manipulation, bit shifts, xor, movzx, mul instructions which strongly indicates decompresses of data into a new memory location.

- DLL Injection Thread: When DLLMain is loaded as a fallback, it creates a new thread, often indicating potential DLL side-loading. The thread performs health checks on fixed memory addresses, gathers data chunks, and executes decryption and decom- pression routines, including XOR-based decryption with a single-byte key. Byte key is hard-coded (in our case 0x82).

–

–

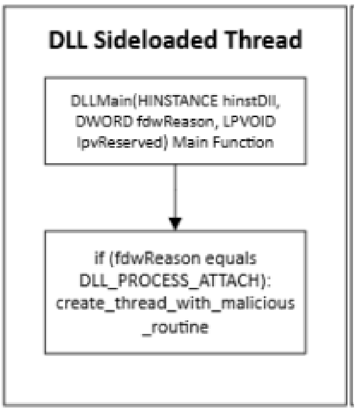
DLL Sideloaded Thread
DLLMain(HINSTANCE hinstDll, DWORD fdwReason, LPVOID lpvReserved) Main Function
if (fdwReason equals DLL_PROCESS_ATTACH): create_thread_with_malicious _routine

**Figure 3:** Side-loading a Thread Routine as part of entry Fallback

- Possible Payload Execution: At the end after pos- sible decryption and decompression payload, it is xor with a bye key. Again VirtualProtect is used to get readwrite access to mem location and call the shellcode which has been loaded in rbx after possible decryption and decompression. After shellcode has been executed, it add sleep and eventually exits.

- Long Sleeps: In decryption loops after each iteration, thread sleeps. This could be used to prevent rapid execution and evade memory protection controls.

Dynamic Analysis of Raindrop yield major conclusions that displays its malicious behavior. Some of the observations are:

### 6.3 Dynamic Analysis

- Absence of Conventional Entry Point: It is a PE64 DLL. Although despite it being a DLL, it cannot be run directly using rundll32. This is because DLL executable did not expose DLLMain or any other function that might be closely related to suspicious thread routine. On running rundll32 malicious dll.exe,DllMain, it gives error ”Missing Entrypoint”. This tactic may evade automated detection that rely on identifying traditional entry points for malicious DLLs.

- Recursive Sleep Functionality: Static analysis revealed the malware exports a Tk MainLoop function, induc- ing infinite loops with long sleeps. When executed via regsvr32, the process shows no significant changes in the registry, file system, or network, serving as a decoy to mislead analysis tools and obscure its true objectives.

- Custom DLL Wrapper for Analysis Executing a mal- ware DLL requires a victim-like environment. To address this, a custom DLL wrapper was developed to load the malware and invoke a function at a specific offset from the base address. This method bypasses the malware’s obfuscation strategies, enabling direct analysis of its malicious functionality.

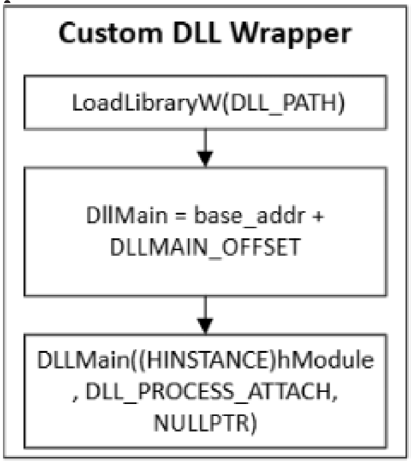
Custom DLL Wrapper
LoadLibraryW(DLL_PATH)
DllMain = base_addr + DLLMAIN_OFFSET
DLLMain((HINSTANCE)hModule , DLL_PROCESS_ATTACH, NULLPTR)

**Figure 4:** Wrapper script to call offset DLLMain

Running it with wrapper, reveals the following observations from PROCMON:

– It validates existence of various registry keys for checking if it is a virtual or sandbox environment. EnablePerProcessSystemDPI, Machine LanguageConfiguration, PreferredUILanguage, Display keys can be undefined in a Virtual environment. This can change malware behavior and act benign where these environments are detected.

– Software Restriction Policy (SRP) is read, specifically CodeIdentifiers registry for DLL execution. It is suspicious if DLLs are checking this key since SRP is an administrative tool used to define policies for controlling the execution of applications (including DLLs) based on their path, hash, or publisher. Typically, it's the responsibility of system administrators or security management software (like endpoint protection solutions) to configure and manage SRP settings, not individual applications or DLLs.

– It creates file (CreateFile) for CRYPTBASE.dll. It can for one of the two reasons. Firstly, it has advapi32.dll import as seen from static analyses. advapi32.dll contains cryptographic APIs. When they are called, Windows internally loads CRYPTBASE.dll as part of the underlying cryptographic infrastructure. Secondly it can be due to GetProcAddress from imports is also a string indicator that malware is dynamically resolving API functions during execution.

– If these registry keys are not found, malware process exits instantly after running its health checks. This is quite clear that it is checking for virtual environment from this. On the contrary, if we add these variables into the registry manually, process executes success- fully until the payload is executed and process exits gracefully.

- Regshot helped us see that on successful execution it is saving the wrapper state in HKLM\System\CurrentControlSet\Services\bam
- \State\UserSettings\S-1-5- 21\Device\HarddiskVolume1\Users\malware- victim\Desktop\wrapper dll.exe. This might be used on next execution to check for malware state.
- Process hacker, it was observed that the DLL was loaded into wrapper's memory as as 7-Zip extension. This is intended to bypass antivirus engines.
- C2 Connection Attempt: After the loader has successfully run and run its payload and exited, it constantly tries to connect to domain https://www.bigtopweb.com. At the time of writing this analyses, the domain WHOIS point to machines in autonomous systems Amazon Inc. It might have been different at the time of attack and can be suspected as a C2 server communication to perform its post exploitation activities since virus total and Kaspersky have flagged this domain as malicious.

Raindrop is a very sophisticated malware when it comes to defense evasion. It has static evasion as well as dynamic evasion techniques that were able to bypass Windows defender as well as many major vendors. It employs a combination of static as well as dynamic techniques to fly under the radar until it drops its shellcode.

### 6.4 Evasion Techniques

This section covers its evasion techniques in detail.

- Process Masquerading: The malware exhibits process masquerading by embedding a significant amount of recompiled source code from the legitimate 7-Zip application. This includes references to the XML manifest associated with 7-Zip, as observed in the binary analysis.
- Debugger Detection and Evasion: The malware actively checks for the presence of a debugger during its execution. If a debugger is detected, the malware terminates itself, effectively avoiding further analysis.
- Architecture and GUI Framework Analysis: Analysis of the decompiled modules reveals that the malware is a compiled DLL designed to mimic 7-Zip, leveraging the cross-platform GUI support of the TK library. This is deduced from the presence of the TK MainLoop function, the main event loop in TK-based applications, as well as the usage of ”Tk” in exported symbols. The presence of DllMain confirms that the sample is a compiled DLL.
- Event Loop Sleep for Evasion: During initialization in the TK MainLoop, the malware incorporates a recursive 60-second sleep, likely to evade detection by automated sandboxes and debuggers. This behavior is consistent with its use of the TK library to mimic 7-Zip's user interface.

- Thread Creation at DLL Load: Upon receiving the DLL PROCESS ATTACH event during DLL loading, the malware immediately spawns a new thread. While the behavior is not definitively malicious at this stage, it raises suspicion and warrants further investigation into DLL sideloading.

- **Polymorphic Behavior:**

– Dynamic Function Resolution: The malware employs polymorphic techniques by invoking function pointers stored in memory. These pointers are populated at runtime within a potentially malicious thread routine, complicating static analysis.

– Randomized Obfuscation: The malware initializes a random seed using the GetTickCount function, which retrieves system uptime. The rand function is subsequently used, possibly to decrypt blocks of pay- load dynamically. This mechanism not only supports obfuscation but also introduces polymorphic traits, making the malware's behavior less predictable.

- Sleep Behavior in Cryptographic Routines: During payload decryption, the malware introduces delays by sleeping between operations. This tactic likely serves to slow down execution in debugging environments, pre- venting rapid analysis of cryptographic routines.

- Anti-Sandbox Techniques and Static Analysis Evasion: Malware employs techniques to detect the presence of a sandbox environment by querying specific registry keys or system attributes. The execution of its malicious routines is contingent upon the absence of these keys, allowing it to evade detection in controlled analysis settings.

### 6.5 Indicator of Compromise

Major IOCs identified from static and dynamic analyses for Raindrop malware from file system and network are:

- SHA256: be9dbbec6937dfe0a652c0603d4972ba354e83c06b8397d6555fd1847da36725

- MD5: 0d7a178a0c0a7d2f2cc63e16dad95b45

- C2 Domain: https://bigtopweb.com (TCP)

- Dll extension: 7z.dll

- Dll extension: 7z.dll.2.Manifest

- Reg Key Access: HKLM\System\Current Control Set\Control\Srp\GP\DLL

- Reg Key Access: HKLM\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers

## 7. Goldmax Malware

This section provides a comprehensive analysis of the GoldMax malware, focusing on its functionality, techniques for achieving persistence, and methods for evading detection.

### 7.1 Introduction

GoldMax is a backdoor malware written in the Go programming language and deployed during the SolarWinds compromise. It is associated with SUNSHUTTLE malware due to similarities in behavior and functionality but the two are distinct tools used in related contexts. GoldMax is designed to enable long-term persistence, communication with command- and-control (C2) servers, and flexible operational capabilities for attackers.

GoldMax employs techniques such as encrypted communication, time-based execution delays, and traffic mimicking to blend into legitimate network activity. These features make it highly effective at evading detection and analysis. The malware provides attackers with remote access, enabling com- mand execution, file transfer, and the deployment of additional payloads.

This section examines the technical aspects of GoldMax, including its functionality, persistence mechanisms, evasion strategies, and its impact on targeted environments during the SolarWinds supply chain attack. Understanding its capabilities sheds light on the tactics employed by threat actors in this operation.

### 7.2 Static Analysis

Static analysis of the GoldMax malware revealed several insights into its design, functionality, and potential objectives. The analysis began with the identification of various strings embedded within the binary. Key findings included strings for an HTTP GET request and an RSA public key, suggesting the use of encryption, potentially for securing network communications or payload delivery. Several cryptographic functions were also identified. The findings are stated:

- Malware Unpacked State: Analyzing the strings highlighted that the malware was written in GO programming language. Examination of the binary's virtual size and raw data size revealed no evidence of packing, indicating the malware was not packing using traditional packers. This finding indicated that the malware is ready to analyze and there is no need for packing. Analyzing the imports table revealed that many functions such as WriteFile, VirtualAlloc, VirtualQuery, and VirtualFree are being used. Which could point to the malware's capability to manipulate memory and files.

- Advanced Threading Library: Application Programming Interfaces (APIs) like CreateThread, Sus- pendThread, and ResumeThread suggested the use of advanced threading techniques to manage its operations. Dynamic library loading like LoadLibraryA, LoadLibraryW, and GetProcAddress were used to load and re- solve functions dynamically, which complicates detection in static analysis. Functions such as AddVectoredEx- ceptionHandler, SetUnhandledExceptionFilter, and Set- WaitableTimer were identified as potentially suspicious, as they are often associated with evading debugging, achieving persistence, and manipulating system behavior.

- Environment Fingerprinting: Code analysis revealed that the malware was using HardwareAddr.String which suggested that the malware was reading the MAC address of the host system. This behavior is commonly associated with attempts to fingerprint the environment, such as detecting virtualized environments in which the malware is being executed. The malware's retrieval of the MAC address likely serves to identify analysis environments, such as sandboxes or virtual machines, and terminate its operations if such environments are detected.
- Session key Request: The malware requested a session key which could be related to securing communication channels with its command-and-control (C2) server or managing encryption for subsequent payloads. Two code blocks were identified as mechanisms for placing the malware in a hibernation state for randomized periods. This mechanism is used to evade detection and analysis, as the delays make its behavior less consistent and more challenging to monitor.
- Cryptographic Use with RSA: The presence of cryptographic functions and the RSA key indicates the use of encryption for data protection. The encryption likely secures C2 communications, ensuring confidentiality and integrity, while also preventing interception by defenders. GoldMax ensures secure and private communication between the compromised host and its C2 server by leveraging RSA keys and cryptographic functions. This not only protects sensitive data from interception but also conceals the nature of the malware's activities from network defenders.
- Beaconing for Persistent C2 Communication: Gold- Max exhibited a beaconing mechanism, repeatedly trying to contact its C2 server. This behavior is typical for establishing persistent communication channels, allowing attackers to issue commands, exfiltrate data, or deploy additional payloads. Repetition of C2 communication attempts ensures that malware can reliably reconnect with its operators, even in the face of network disruptions or environment changes. This mechanism also allows attackers to maintain control over the compromised system for extended periods.

The static analysis findings are verified by doing dynamic analysis of the GoldMax malware.

### 7.3 Dynamic Analysis

The dynamic analysis of the GoldMax malware provided significant information on its runtime behavior, network activity, file manipulation, and registry modifications. The observations highlight the sophistication of the malware in maintaining persistence, evading detection, and communicating with its command-and-control (C2) infrastructure. The techniques used by GoldMax malware to evade detection are as follows:

- Communications with Command-and-Control Server: The malware-initiated DNS requests to megatoolkit.com, which is likely the C2 server used for communication and control. GoldMax engaged in repeated TCP com- munication over ports 443 and 80, indicating the use of HTTPS and HTTP protocols. These connections involved full TLS handshakes followed by periodic data exchanges every 8-10 seconds, likely serving as a beaconing mech- anism or a means of sending system status information to the C2 server.
- Encrypted Communications via TLS: The TLS hand- shakes and subsequent application data transmission in- dicate that the malware uses encryption to protect its network communications, ensuring confidentiality and integrity against interception.
- Creation of Encrypted Configuration File: GoldMax created a new file named runlog.dat.tmp in the same directory as its executable right after the execution of the malware. The file contained a single line of encrypted data, possibly used to store configuration details, a system fingerprint, or a one-time initialization value.
- Registry Key Deletion: The deletion of the multiple registry keys indicates an attempt to disable legitimate system policies, potentially to prevent system hardening measures, alter authentication rate limits, potentially bypass login restrictions or brute-force protections, to gather intelligence about user interactions or frequently accessed files. Removal of entries related to OneDrive and BITS, along with modifications to Windows Update configurations, suggests an effort to disrupt legitimate update mechanisms. This could prevent system patches from being applied or allow the malware to replace updates with malicious payloads.
- Registry Manipulation for Persistence: The interaction with the advapi32.dll library indicates registry manipulation, which is likely to disable security policies or alter configurations to maintain persistence.
- Payload Encryption: The use of cryptographic functions points to encrypted C2 communications or data storage, ensuring that communications remain confidential. These functions may be used to encrypt payloads or credentials locally.
- Network Communication Through DNS: The use of dnsapi.dll and wsock32.dll libraries allows network communications, confirming the malware's reliance on DNS and socket-based interactions for contacting the C2

infrastructure.

- Persistence: Interaction with sysmain.sdb database suggest the use of Windows Application Compatibility features to establish persistence. Manipulation of sdb extension files is a known tactic for bypassing standard execution restrictions or injecting malicious behaviors into legitimate processes.
- Hibernation Mechanisms for Evasion: The malware likely incorporates hibernation mechanisms, delaying execution or communication for random intervals. This behavior reduces the likelihood of detection by automated systems, which often rely on consistent patterns.
- Registry Manipulations for Credential Theft: The use of registry manipulations and cryptographic libraries indicates attempts at credential theft or privilege escalation, enabling malware to access restricted areas of the system.
- Heartbeat Communication: The periodic communication with the C2 server serves as a heartbeat, signaling the malware's continued presence and operational status. This enables attackers to issue commands, deploy additional payloads, or exfiltrate data.

### 7.4 Evasion Techniques

GoldMax malware exhibits a range of sophisticated evasion techniques that are critical to its success in maintaining stealth and avoiding detection in compromised systems.

- Beaconing Mechanism with Unpredictable Intervals: GoldMax communicates with its C2 server by performing periodic TLS handshakes and sending data at intervals of 8-10 seconds. The beaconing mechanism is unpredictable, making it harder for automated tools to detect the regular patterns often associated with malicious activity.
- Hibernation to Evade Detection: The malware includes functionality to enter a hibernation state, where it pauses its execution for random periods of time. This technique reduces the likelihood of detection by automated sandbox environments that monitor malware behavior for only a limited duration.
- MAC Address Retrieval: The use of functions like HardwareAddr.String to retrieve the system's MAC address could indicate checks for virtualized or sandboxed environments.
- Encrypted C2 Communications with TLS: All communications with the C2 server are encrypted using TLS. This encryption ensures that network monitoring tools cannot decipher the content of the communication, obscuring any commands or data being exfiltrated.

### 7.5 Indicator of Compromise

The indicator of compromise identifed during the analysis of GoldMax malware are:

• Cryptographic Libraries: crypto/aes, crypto/rc4, and crypto/tls.

• Key strings: runlog.dat, HardwareAddr.String.

• API calls: VirtualAlloc and LoadLibraryA.

• C2 Communication: megatoolkit.com.

## 8. Goldfinder Malware

This section provides a detailed analysis of the GoldFinder malware. This section focuses on the structure, functionality and the evasion techniques used by the malware.

### 8.1 Introduction

GoldFinder is a reconnaissance malware linked to the SolarWinds Compromise. It is deployed to test the network environment of targeted machines. It is designed to evaluate network connectivity and routing, enabling attackers to understand the machine's network setup and the presence of proxies, firewalls, or other intermediaries.

### 8.2 Static Analysis

Static analysis of the GoldFinder malware revealed that it is packed. It indicates that the malware has obfuscated its internal structure and functionality. When a malware is packed the true code is hidden until the malware is unpacked at runtime. This is a common evasion technique used by malware to delay detection and analysis by security tools.

In the code review it was observed that the malware is configured to make HTTPS requests to google.com. This action indicates that the malware is testing network connectivity for testing purposes. The malware verifies network connectivity and determines the presence or configuration of security mechanisms like proxies or firewalls. The malware can infer the following by analyzing responses of requests:

- Proxy Interception: If a proxy intercepts the request, additional headers like Via or X-Forwarded-For may be added, which the malware can analyze.
- Defense Mechanism: The malware can deduce the presence of network-level filtering or blocking mechanisms if network packets are modified or filtered.

Detecting traffic with a firewall is challenging because the malware generates innocent-looking traffic.

### 8.3 Dynamic Analysis

The findings of the GoldFinder malware are as follows:

- File Creation: The malware creates a file named loglog.txt in the same directory where the malware is executed. The malware initiates an HTTPS request to google.com. The malware generates legitimate traffic to avoid suspicion and bypass network defenses such as firewalls and intrusion detection systems (IDS).

- Logging: The response of the generated traffic is written into the loglog.txt file including status code and other metadata. This behavior indicates that the malware is performing reconnaissance to assess the network environment. By analyzing the logged response, the malware can deduce whether the machine has an active internet connection or not.

## 8.4 Evasion Techniques

The following evasion techniques were noticed during the analysis phase:

- Packed Malware: The malware is initially packed. It means that the inner functionality is obfuscated and true code is hidden until the malware is unpacked at runtime.
- Legitimate Traffic: Malware generates legitimate traffic bypassing security controls.

## 8.5 Indicator of Compromise

Indicators of Compromise are used to identify the malware and make rules to filter it. Following indicators of compromise were identified in GoldFinder malware:

- Executable Packer: Ultimate Packer for Executables (UPX).
- File Creation: loglog.txt.

## 9. Conclusion

The activities of GoldFinder, GoldMax, Raindrop, and Sibot represent a coordinated cyber operation that demonstrates the strategies used in advanced persistent threat campaigns. Each malware had a distinct role in achieving the goals of the SolarWinds supply chain compromise. GoldFinder focused on reconnaissance to test network defenses and identify security mechanisms. GoldMax operated as a command-and-control backdoor that maintained long-term access to compromised systems. Raindrop acted as a loader to facilitate lateral movement and deploy additional payloads. Sibot served as a downloader to establish persistence and execute further malicious components.

GoldFinder was a reconnaissance tool designed to evaluate network configurations, identify the presence of security mechanisms, and identify the presence of internet connection. It assessed defenses such as firewalls and proxies by generating traffic that appeared harmless and directing it to legitimate sites like Google.com. The tool logged the responses of the requests and provided critical insights into the network environment. The reconnaissance activity was a necessary step in facilitating the deployment of more malware programs of the operation.

GoldMax was a command-and-control backdoor. It secured communications through encryption to prevent detection during data transmission. The backdoor used beaconing to provide updates on system status and ensured its presence through persistence mechanisms that allowed it to remain operational on compromised systems. The modular structure of GoldMax enabled the attackers to deploy commands dynamically and exfiltrate data as required. The functionality highlighted the use of sophisticated backdoor techniques to coordinate and carry out malicious actions across a wide array of victims.

Raindrop operated as a loader and facilitated lateral movement for delivering additional payloads. It allowed the attackers to expand their reach by targeting more systems within the compromised network. The functionality allowed the attackers to strengthen foothold and exploit the inter-connectivity of the organization. Due to the facilitation of lateral movement of malicious code, Raindrop highlighted the cascading effects of the campaign, where the compromise of one system could lead to widespread disruptions across the entire network.

Sibot is a script-based downloader. It utilized Windows Script Host to execute its payloads so that its activities remain less suspicious and less likely to trigger security defenses. Sibot was responsible in retrieving and executing additional malware components within the compromised environment.

The malware components represent a carefully planned and executed campaign that targeted government agencies and critical infrastructure entities along with private-sector organizations on a global scale. The SolarWinds compromise enabled by the collection of malware tools demonstrates the potential for supply chain attacks to bypass traditional security measures and infiltrate highly secure environments. The attackers use of legitimate traffic patterns and encryption techniques ensured the success of the operations while making detection and remediation efforts more difficult.

The campaign serve as a stark reminder of the vulnerabilities present in modern supply chains and the critical need for advanced security measures. Organizations must prioritize the implementation of comprehensive monitoring solutions, incident response strategies, and regular assessment of risks to mitigate the threats posed by such advanced campaigns. The SolarWinds incident shows how important it is for organizations to work together to improve cybersecurity. Sharing information and building strong defenses as a group can help tackle the growing challenges in today's complicated digital world.

## References

1. SolarWinds, SolarWinds Network Performance Monitor Datasheet, Jul. 2023

2. SolarWinds, SolarWinds Security Event Manager Datasheet, July 2023.

3. SolarWinds, Orion Platform Datasheet, 2024.

4. R. Nair, K. Dodiya, P. Lakhalani, and K. Dodiya, “A Static Approach for Malware Analysis: A Guide to Analysis Tools and Techniques,” International Journal for Research in Applied Science and Engineer- ing Technology, vol. 11, no. 12, pp. 1451–1474, Dec. 2023. DOI: 10.22214/ijraset.2023.57649.

5. M. Guven, “Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence,” International Journal of Computational and Experimental Science and Engineering, vol. 10, no. 3, Sept. 2024. DOI: 10.22399/ijcesen.460.

6. R. V. Mahmoud, M. Anagnostopoulos, S. Pastrana, and J. M. Pedersen, ”Redefining Malware Sandboxing: Enhancing Analysis Through Sys- mon and ELK Integration,” IEEE Access, vol. PP, no. 99, pp. 1–1, Jan. 2024,

7. M. Malik and M. B. Rahardjo, ”A Framework for Collecting and Analysis PE Malware Using Modern Honey Network (MHN),” in Proc. 8th Int. Conf. Cyber and IT Service Management (CITSM), Pangkal Pinang, Dec. 2020, pp. 1–5, doi: 10.1109/CITSM50537.2020.9268810.

8. V. Dutta, M. Raghavendra, and R. Ramesh, ”Machine Learning in Malware Detection: A Survey of Analysis Techniques,” Int. J. Adv. Res. Comput. Commun. Eng., vol. 12, no. 4, pp. 204–208, Apr. 2023, doi: 10.17148/IJARCCE.2023.12435.

9. B. Hammi, J. Hachem, A. Rachini, and R. Khatoun, ”Malware Detection Through Windows System Call Analysis,” in Proc. 9th Int. Conf. Mobile Secure Services (MOBISECSERV), Miami, USA, Nov. 2024, pp. 1–7, doi: 10.1109/MobiSecServ63327.2024.10759991.

10. N. Srinivas, ”Detection of Malware by Using YARA Rules,” in Proceedings of the 9th IEEE International Conference on Cybersecurity and Malware Analysis (CYBERMA), 2024, DOI: 10.1109/CY- BERMA.2024.10549308.

11. M. I. Yousuf, I. Anwer, A. Riasat, and S. Kim, ”Windows malware de- tection based on static analysis with multiple features,” PeerJ Computer Science, vol. 9, no. 1, p. e1319, Apr. 2023. DOI: 10.7717/peerj-cs.1319.

12. SolarWinds Compromise, Campaign C0024 — MITRE ATT&CK®, attack.mitre.org. https://attack.mitre.org/campaigns/C0024/

13. Newsweek, ”SolarWinds Orion Software Cyberattack: Hack Victims, Targets List,” Dec. 22, 2020. [Online]. Available: https://www.newsweek.com/solarwinds-orion-software-cyberattack- hack-victims-targets-list-1555840. [Accessed: Nov. 15, 2024].

14. Security.com, SolarWinds Raindrop Malware: Understanding the Threat.