从Solarwinds供应链攻击（金链熊）看APT行动中的隐蔽作战

By zcgonvh&&rem4x

Archived: 2026-04-05 13:25:11 UTC

作者：zcgonvh && rem4x

校对：zcgonvh、rem4x、n1nty、L.N.

编辑：L.N.

0x10 前言

有过实战经历的红队成员都应知晓，任意行动在任意路径下，可能被捕获的任意行为，均可能造成行动暴露。站在攻击者的角度思考，这里体现了三个信息：

1. 攻击者在行动前，应意识到可能导致行动暴露的信息，并制定避免方式。

2. 攻击者在行动时，应严格按照行动前的隐蔽条例作战，并及时处理突发情况。

3. 攻击者在行动后，应清理遗留信息，并封存已用资源。

考虑现阶段安全环境现状，无论是基于主机或是基于网络的防御均日趋完善，完美的隐藏在现实中是不可能的。攻击者能够做到的只是尽量隐匿而不可能完全隐身，无论多么完美的手段都会留下痕迹，这也是攻击侧广为人知却又颇为无奈的事实。

这些不可避免的痕迹构成了防御侧的情报基础。利用这些情报信息，虽然事前的情形我们无从知晓，但是在事中和事后，我们捕获的攻击行为，提供了直接触碰攻击组织的窗口，其中任何有价值的信息均可以作为证据。这些证据无论在事中止损，还是事后溯源，乃至攻击反制都将起到极大作用。

基于现有的事实，结合多维度的分析与推断，最终能够描绘出较完善的攻击者画像。

本文从全局出发，以技术作为手段，以代码、手法和行为作为支撑。通过推测对方的实际行为、人员配比、后续行动，尽最大可能窥视并揭露行动背后掩藏的真相。

这不是威胁情报文章，因此我们不会花费大量的篇幅在描述常规背景、威胁影响与IOC上面，这对从更高维度观察APT威胁毫无帮助。这也不是样本分析文章，我们同样不会花费大量的篇幅描述如何逆向代码、如何使用工具或是发现了某种调用表明做了某些事等等，这些常规代码分析与猜测在针对类似组织的行动时，是极为苍白无力的。

我们希望将实际行动的思想融入到样本分析与APT追踪，同时借此文揭露的真实APT手段，为安全从业者敲响警钟：**APT攻击不是单纯的安全事件，而是不见硝烟的战争。我们面对的是成组织、成建制的攻击，作为样本分析人员应当扮演侦察兵的角色，最大化提升自己的行动意识与攻击视界，同时磨练出对等作战的能力，否则必将在战斗中一败涂地。**

根据行业知识的积累与逻辑思维理解的不同，理解本文至少需要一天至一周的时间。

0x20 代码

有别于样本分析，在探究一个组织真正的目的时，我们不应该过多的关心API是如何调用的、读了什么、写了什么或是用什么加密算法进行了什么变换等细枝末节。举个最简单的例子，msfvenom可以使用任意encoder进行编码，大部分encoder中都存在随机字节用于将输出随机化（例如alpha），这样的样本数量可以认为是无限的。

我们在分析时不会关心alpha编码使用了哪个随机字符作为混淆，因为这是毫无意义的。类比于此，任何攻击者都可能将具体技术进行等价替换，例如：key随机化、加密算法随机化、等效API代替等等。如果一次事件分析仅仅关心相似的细枝末节，把加密key和算法、调用的API、发送的数据包等细节输出成文。这样的文章在整体的攻击行为追溯上可以说是非常失败的。

事实上我们应当首先把自己的角度转化为攻击者角度。作为攻击者只用关心一点：样本中的代码能为行动起到何种推进？带着这个疑问再转回分析人员，对每一段代码功能进行提问：为什么对方的行动需要设计此功能？随即又转回攻击者：我拿到了这段代码的执行结果/我实现了这个操作以后，我要做什么？

有了依据事实而得出的合理推测，结合情报获取到的实质证据，我们至少在行动手段上能够为攻击者制造画像，甚至可以为后续**战术**甚至**战略**层面分析提供大量的帮助。

这才是代码层面分析的核心目的。

0x21 入口选择

根据火眼的分析报告，恶意代码包含在SolarWinds.Orion.Core.BusinessLayer.dll中，恶意类名称为SolarWinds.Orion.Core.BusinessLayer.OrionImprovementBusinessLayer，其入口点是Initialize方法。

我们不要匆匆忙忙的进行分析，首先思考一个问题：此dll具有合法的签名，很显然攻击者从源码阶段进行了控制，那么最初的突破点在哪里？考虑软件工程的工作流程，攻击者最有可能发起感染的位置是代码仓库，这样能够最大限度避开commit之前的review，以及提交期间的自动化代码扫描，防止在开发阶段被发觉。

代码仓库必然会包含多套代码，那么再思考这样一个问题：为什么恶意代码会放置于此dll中，此代码谁又会进行调用？类比Web攻击中的后门驻留，恶意代码插入的位置实际上能在某种意义上决定存活时间，而对于位置的选择往往能直接反映攻击者的行动思路。

带着这个疑问查看恶意代码交叉引用，我们可以得到以下（图0）调用堆栈：

图0

恶意代码被插入至RefreshInternal方法中，而非常规的入口方法。实际的入口方法Start作为父类BusinessLayerPlugin的重写，将在主进程SolarWinds.BusinessLayerHost.exe中被调用，中间有着很长的代码路径。

攻击者选择了一个非常深层次的调用堆栈，用来降低代码重构期间被发现的可能。同时，藏匿在C#托管类库，而非直接寄生于主程序本体，实际上也属于后续运行期间的掩护手段之一。

再关注具体实现：RefreshInternal方法中插入了以下代码（图1）。此代码用于检测是否启动，如未启动将开启新线程运行恶意代码：

图1

检查项IsAlive进行了全局锁（图2），保证了在插件运行期间只有一个实例运行。

图2

是的，此后门在插件生存周期内只会运行一次。

在后续分析中还将看到大量类似的行为，这是和常规攻击完全不一样的点。

0x22 环境检测

在执行后门功能前，代码将进行长达9层的判断，用于检测当前运行环境。

几乎所有的判断都是通过自定义hash算法进行的，这保证了无论是在源码，还是在编译后的程序集中，均不会存在敏感字符串，从而降低被查杀的可能，对于开发期间可能的人工检测也是一种掩护。

后门首先会检测父进程名的hash是否为17291806236368054941（图3），对应进程名SolarWinds.BusinessLayerHost.exe（国内部分翻译型安全团队的结论几乎照抄了FireEye的结论，且包含初期的笔误，欲见详情请在“搜一搜”中搜索关键字：SolarWindws.BusinessLayerHost。）。考虑到整体的谨慎性，这可以理解成防止在多进程中多次加载的双保险，也可以认为是一个基本的反调试验证。

图3

如果检测通过，后门将读取SolarWinds.Orion.Core.BusinessLayer.dll的最后写入时间。我们考虑一套软件的安装部署流程，这几乎可以认为是后门落地时间。之后生成一个位于288-336之间的随机数作为经过的小时间隔（图4），与落地时间相加后和当前时间做对比。只有当前时间大于这个随机时间间隔才会上线。

这意味着后门的上线时间和落地时间的间隔被拉长并随机化，在落地的前12天是不会上线的，12-14天则随时间逐渐提高上线几率（图5），直到14天后才会确保上线。

图4
图5

在时间判断通过后，首先根据硬编码名称583da945-62af-10e8-4902-a8f205c72b2e（图6）创建命名管道（图7），用于保证多进程中的互斥行为，如失败将抛出异常并在捕获后退出线程。

图6

图7
随后将检查一个被恶意复用的配置选项ReportWatcherRetry。此选项本用于[SolarWinds.Orion.Core.Common]SolarWinds.Orion.Core.Common.OrionReportHelper::InitReportsWatcher方法中（图8），作为正常处理报告期间的重试次数，其默认值为4。
恶意代码将检查此项，其3、4、5的值分别对应Truncate、New、Append三种持久化状态（图9）。

图8

图9
默认值对应New，表明新植入。而值为3对应的Truncate状态将导致后门直接退出。
这里滥用了一个已知选项，在不影响正常功能的情况下对自身状态进行了持久化，其设计思路隐蔽巧妙，显示出了非常老练的后门功底。**本次行动中攻击者多次将DNS协议本身携带的信息作为控制信道的一部分使用，这明显体现出高对抗状态下磨练出的实战技术视野，从技术和思维上已大大超过常规行动体的范畴**，相关的细节将在0x23、0x24章节进行描述。
如果状态不为Truncate，则将获取当前主机的Windows Domain Name，并进行以下两项检测：
1. 是否在patternHashes黑名单列表中。此黑名单维护了13个已知域（图10），通过简单的爆破可以得到以下结果：

图10
注意红圈所在的三个域名，我们有理由相信swdev为SolarWinds develop的缩写，swi为SolarWinds Internal的缩写，这从侧面证明了0x21中代码仓库已被实际控制的推测。
2.域名是否包含test或solarwinds（不区分大小写），用于排除测试域或供应商内网（图11）。

图11
域名检测通过后，首先检测网卡，根据第一个非回环启用网卡生成唯一标识符（图12），并在后续通信中进行发送。
图12

随后进入运行环境检测阶段。阶段第一步会对137个进程Hash黑名单进行检测，此黑名单中绝大多数为常见分析工具如ProcExp、WinHex、WinDbg等。
第二步会遍历进程，计算Hash并尝试查找对应的杀软服务名称，共8组22个进程41个服务。如当前环境任一进程Hash与列表匹配，则遍历注册表HKLM\SYSTEM\CurrentControlSet\services子项，修改任何匹配的服务项权限为完全控制，并将服务状态设置为禁用（图13）。
图13
成功设置禁用后，会使用另一个被恶意复用的配置选项ReportWatcherPostpone保存服务状态，此选项默认值为250。新状态通过位进行保存，恶意代码会在默认值的基础上与新状态求和并持久化写入（图14）。
图14
如果未修改任何服务状态，则会执行WMI查询：Select * From Win32_SystemDriver来获取运行环境中已加载的驱动程序列表（图15），并和已知黑名单列表进行对比。此列表包含以下17项，基本为edr的过滤驱动：
图15
如进程、服务、驱动状态检查通过，最后将解析官方域名api.solarwinds.com，如成功则转入上线逻辑（图16）。上述过程可用流程图表示如下：

图16
根据0x21的分析，如任一检测失败，在父进程SolarWinds.BusinessLayerHost.exe不重启或插件不重新加载的情况下不会再次上线（最坏的可能需要等待系统重启）。
上述整体流程体现了在攻击者思想中**隐蔽处于绝对的首位**，攻击者宁可放弃大量的上线机会也不愿在某个非安全环境上线，这种谨小慎微的做法和常规情况是完全相反的。

0x23 敲门上报

通过重重检测，后门将进入DNS敲门阶段。此整个生存周期内会通过一个DnsRecords对象（图17）保存敲门阶段获取的信息，并传递至控制阶段：

图17
代码通过DNS协议中正常的名字与结构进行混淆视听，在后面我们还会看到更多写作A表示B的情形，一方面作为隐蔽措施，另一方面在一定程度上增加了分析成本，这些在常规样本中都是难以遇到的。
恶意代码在敲门的第一步采用了dns协议，通过DGA生成随机域名，其实现为状态机。根据植入状态的不同，实际上存在两种DGA算法进行计算（图18）。

图18
当持久化状态为New时，此时恶意代码认为处于首次运行（下称一阶段）。一阶段DGA由GetPreviousString方法生成，会进行以下操作：
1. 编码内网DNS主域名
   a)如全部由小写、数字、下划线、点组成，则执行码表变换。
   b)否则，进行自定义映射表base32编码。
2. 编码guid，并将随机key高位置位，随后进行base32编码。
3. 编码后的主域名进行分段，并将编号通过编码后的guid的第一个字符求和进行编码。

4. 合2、3、1三部分，作为DGA域名。
随后，DGA域名将与四个随机的主域名进行拼接，并进行DNS查询。
由于分段的原因，此阶段可能进行多次连续请求，任一分段发送失败都将调用GetCurrentString进行重发。
返回的CNAME信息将保存至DnsRecords对象，根据查询结果，将进行以下判断：
1.如请求失败，则设置随机7-9小时的休眠时间（图19），并在下一次尝试时首先进行休眠操作。

图19
2.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，填充DnsRecords对象的以下字段：
a)length字段，此字段用于保存控制阶段应用的代理类型（图20）。

图20
b)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（图21）。

图21
c)A字段，此字段决定敲门阶段每次DNS请求之间的延时（图21）。

图21
3.无论CNAME是否存在，之后均将遍历以下名单（图22）：

图22
注意这里的AddressFamilyEx枚举，其中使用Atm作为内网IP黑名单，ImpLink作为外网IP黑名单，Ipx作为等待名单，NetBios作为白名单。包括IPAddressesHeler类在内，这同样都是通过看似正常的代码混淆视听的行为。
如返回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，当后续恶意代码执行时，由于0x22检测 阶段的判定检测失败从而永不上线（图23）。

图23
若处于等待名单，则将状态重置为New，退出此次活动，并等待下次调用。（图24）

图24
若处于白名单，则循环发送被分段的域名直到结束，随后进入二阶段（图25）。

图25
其他任何情况下，将退出此次活动，并等待下次调用。
这实质上是一个**自毁/上线/休眠**开关，判定的依据则是DNS协议返回的IP本身。攻击者充分利用了常规DNS协议中的数据交互，巧妙地将所需信息以寄生的方式隐写在正常协议数据内，摆脱了常规基于TXT记录接收信息返回的方式，其谨慎程度可见一斑。
当域名被完整发送后将进入阶段二，此阶段DGA由GetNextString方法生成，该方法会进行以下操作：

1.计算自2010-1-1 0:0:0开始，以半小时为单位的时间戳，取后三字节并对高4位进行混淆。
2.以时间戳为key对guid进行循环异或。
3.拼接2、1，产生高位未置位的随机key并进行编码，随后进行base32编码作为DGA。
同样的，该DGA域名将与四个随机的主域名进行拼接，并进行DNS查询，查询结果同样将进行上述黑白灰名单验证，如验证通过、包含CNAME且CNAME所指向IP被标记为ext，则敲门阶段结束，进入后续控制阶段（图26）。

图26
综上，敲门阶段整体的流程图（图27）可总结如下：

图27
结合流程图和代码实现可以明显看出，一阶段DGA中包含了可以完整解析的域名，二阶段DGA中至少包含了有效的服务器时间戳，两个阶段具有相同的guid且可作为关联。这些都属于攻击者在设计阶段认为有用从而刻意搜集的信息。
同时，一二阶段的CNAME存在与否、IP本身最后两字节在后续攻击中具有实际含义，这些不可能通过传统的配置NameServer进行实现。
所以我们完全有理由相信攻击者编写了一个定制化的NameServer服务，并承担了整个供应链攻击过程中的控制中枢，此服务至少应当具备以下几点能力：
1.收集被感染目标相关信息。
2.根据回传信息进行目标精准辨识。
3.根据既定作战任务进行目标筛选、研判并作出具体指令：终止、等待或是持续渗透。

0x24 控制机制

相比复杂的DGA敲门，HTTP控制阶段相对简单得多。

该阶段的入口点为
SolarWinds.Orion.Core.BusinessLayer.OrionImprovementBusinessLayer.HttpHelper::Initialize方法，和DGA敲门的代码结构类似，核心循环同样通过状态机实现。
HttpHelper的构造函数会处理由敲门阶段决定的请求类型与代理类型（图28）：

图28
requestMethod字段伪装成合法的HTTP请求方法，其实际上决定了整个控制阶段的HTTP数据格式，具体为以下三点：
1.决定User-Agent（图29）：

图29
可以看到存在两种User-Agent，在正常情况下，第一种会用于Windows检查证书吊销列表，第二种用于SolarWinds本身的通信过程。
考虑到SolarWinds本身是一个服务器软件，攻击者精心选择了正常服务器发送的User-Agent，并根据系统环境精确伪造了版本，在对抗层面上进一步提升了伪装的可信度。
2.决定Url：

无论采用了哪种User-Agent，后门最终使用的Url都将在合理范围内进行最大限度的随机化。
如果采用了第一种，则Url伪装为一个证书吊销列表文件（图30）：

图30
否则将伪装为SolarWinds正常请求链接或静态资源（图31、32）：

图31

图32
Valid方法用于百分比随机，使得预测某个Url的产生是不可能的，这在一定程度上对基于统计学的恶意行为发现起到了对抗作用。
3.决定如何解析返回数据（图33）

图33
在整个执行流程中，User-Agent、请求的Url、对应的返回格式均严格对应，显示出了极为缜密的编码思路。
而后的出网代理主要分为三类，无代理、系统代理以及SolarWinds本身配置的代理（图34）：

图34
注意红框位置，这段代码调用了SolarWinds本身功能实现，显然攻击者对该套系统极为熟悉，同时也为控制代码仓库的假设提供了另一重佐证。
之后也不要忘记，这两个字段由DNS请求返回的IP解析而来，将IP地址本身作为控制信道的一环来决定后续行为，是本次行动一个非常显著的特征。这同时存在三个好处：
1.基于域名的数据重放，很有可能无法得到有效信息甚至会得到完全错误的结果。
2.若对方可以通过DNS出网但无法上线，可通过修改解析的方式更换代理配置。这也为上面自定义NS服务器的假设提供了证据。
3.可能用于对抗在斯诺登事件中泄漏的TUTELAGE系统，该系统由NSA开发，其中CLOUDSHIELD模块使用了DNS劫持技术对未知威胁进行诱捕。据泄露信息，该系统被广泛部署在美国的国防承包商、军工企业以及情报机构，这与本次行动的受影响目标相吻合。
控制过程中的HTTP请求同样经过高度伪装，根据代码还原的一个请求大致如下（图35）：

图35
有效信息被完全打散并伪装成合法的json格式进行传输，同时传输过程中刻意仿冒了Solarwinds的真实URL，容易让专业水平较低的安全分析师误判为合法通讯。
和协议相比，具体的控制指令实现反而乏善可陈，事实上这在远控开发中也是最微不足道的细枝末节。我们查看实际实现的功能，可以很明显的看出仅有基本信息、文件管理、注册表管理、无回显命令执行、重启等基本功能（图36），不存在动态拓展、也不存在横向移动等后渗透阶段的相关能力支持。

图36
如此简陋的控制功能和极为复杂的上线逻辑形成了相当大的反差，那么只有一种可能：此后门在实质上仅仅是一个高隐蔽性的探针，后续攻击一定会更换后渗透的攻击套件进行下一阶段的行动。

同时结合敲门阶段的自毁功能，我们甚至有理由怀疑真正进入后渗透阶段后探针将停止一切活动，该功能可由控制中枢（NameServer）下发黑名单域名直接完成，也标志着这套寄生于SolarWinds中的供应链后门生命周期的结束。
通过分析相关的DGA数据获取的信息也可以得出本次行动的供应链攻击环节由三月份开始并在七月份达到顶峰，随后成逐渐下降的趋势并于十月份消失在视野之中，详见0x41章节。

0x25 行动架构

结合上述已知行为，我们可以大致勾勒出该后门探针的执行流程（图37）。这个流程的核心体现在了上线阶段的高强度运行环境检测、高度定制化的NameServer作为控制中枢以及寄生于DNS协议的控制信道：

图37
将这样复杂的定制化后门作为一次性探针使用，这样高的开发成本和攻击成本对小团队来说是不现实的。
实际上能够设计出这样一种执行流程本身体现了攻击者在行动方面造诣颇深，通过IP这种难以精确控制的资源作为控制载体从侧面能够体现出攻击者本身维护了一个庞大的基础设施网络资源库，同时也体现出了该组织较为雄厚的经济实力。

0x30 追溯

在前言中提到，已被捕获的攻击行为实质上提供了一个直接触碰攻击组织的窗口。

现在我们捕获了样本，并对所实现功能进行了完整分析并初步推断了原因。接下来只要以此为基础进行拓展，便能追寻出关于攻击者更多的蛛丝马迹。

0x31 域名

从样本DGA算法中我们知晓，敲门阶段的DGA域名包含可解密信息，而互联网上大量可供查询的PDNS记录为我们收集此类信息提供了足够多的样本。

从代码层面分析可以得出，攻击者设计了一套精妙的DGA技术用于信息渗出及指令下达。但也正因为如此，我们可以通过奇安信司南系统（PDNS）提供的数据对攻击组织的活动进行时空层面的回溯并反推出具体的行动规划。
一阶段DGA域名的作用是通过DNS请求，将内网DNS域完整传递至探针主控端，在绝大多数时候这等同于Windows AD域名。通过AD域名可以推断出实际的受影响范围并尝试猜测攻击者的战略目的。
将DGA域名进行Base32解码，查看首字节最高位是否置位，即可筛选出全部一阶段DGA域名；将第15个字符转换后和第1个字符相减，即可确认这是分段传输中的第几部分；从第16个字符开始是有效内容的片段，将所有片段按顺序组合并进行Base32，最终将得到完整的AD域名。具体代码实现参考文末链接，在此不做赘述。
根据奇安信CERT对该事件的持续追踪分析，截止12月16日，已确认受害的重要机构至少200家，波及北美、欧洲等全球重要科技发达地区的敏感机构，其中美国占比超过60%。

0x32 时间

同样的，敲门第二阶段包含大量时间信息。同样将DGA域名进行Base32解码，首字节最高位未置位即表明是二阶段DGA域名；将解码后的数据和第一字节异或，即可得到原文；原文的9、10、11三个字节即为以大端序存储，从2010-1-1开始，以半小时为单位的时间戳。
同样结合0x31中的DGA域名，我们可以成功解密出更多的数据，**这些数据可以揭示攻击组织花费了6个月以上的时间进行目标筛选与识别工作**。而在具体的时间跨度上，第一个出现的时间戳为2020/4/4 6:30:00（图38），考虑到至少12天的延时，最早在2020/3/23 6:30:00已有受害者被植入恶意样本。

图38
另外一个时间角度，我们可以从VT上面查找所有SolarWinds.Orion.Core.BusinessLayer.dll，并确认其编译时间。根据VT上全部66个样本的结果，最早一个存在后门的版本编译于2020/3/17（图39）。

图39
通过奇安信司南系统提供的WHOIS变更记录也可以看到，攻击者在2020/02/26 12:04:03 GMT对NS记录进行了初次变更（图40），并指向了已知的六个恶意控制中枢服务器。

图40
这里值得注意的是，在行动筹备、实施阶段（2020年2月26日至2020年10月8日）发生的whois记录更新为北京时间的20点至22点，美国相关组织强制修改的时间为北京时间的0点至2点。
众所周知，美国与中国的时差大致为12小时（地球半圈儿），说明美国人在正常的工作时间内完成了本次接管操作。由此推断，攻击者与美国的距离大约是小半圈儿。

0x33 伪装

任何有代码功底的人看到本次样本的代码都应如同感到耳目一新，和充斥于安全圈的低质量代码不同，此样本的编码非常工整。代码本身高度仿照了SolarWinds的编码方式与命名规范，类名、方法全部采用Pascal命名，而字段和方法则使用驼峰式命名，将大量可复用功能单独抽取变为方法，将同功能代码封装为类，这些均是微软推荐的标准C#编程规范。

类似编码规则会大大增强针对Review以及代码自动扫描时的存活能力。事实上如果不是对安全有所涉猎的人员，单纯开发人员几乎很难发现代码中隐藏的问题。
同时，代码本身融入了安全人员“滥用”的思想，其代码中大量写做A表示B的成员提供了绝佳的掩饰，例如：
AddressFamilyEx.Atm表示内网黑名单。
AddressFamilyEx.ImpLink表示外网黑名单。
AddressFamilyEx.NetBios表示白名单。
CryptoHelper.Base64Encode实际为自定义映射表的Base32编码。
CryptoHelper.Base64Decode实际用于映射表替换。
DnsRecords._type和DnsRecords.length用于控制HTTP行为。
大量的TimeStamps实际上表示字符串Hash。

不影响原本功能的情况下复用数值型配置ReportWatcherRetry和ReportWatcherPostpone。
等等等等，不一而足。
上述举重若轻的实现体现出攻击者极为老练的代码功底。NativeMethods、AddressFamilyEx、nList、nCount等命名有着太过强烈的微软风格。事实上结合上述信息我们完全可以断定，探针的开发人员中至少有一名极为熟悉WindowsAPI编程与DotNetFramework源码的核心人员，此人在对方组织内负责把控整体的代码工程，包含设计、构建、编写甚至Review。根据代码工整度与实现逻辑的清晰程度，我们有理由相信其从事相关工作的时间至少在三至五年以上。

0x34 WebShell

除供应链攻击的核心探针工程之外，还存在一个带有webshell功能的扩展文件：App_Web_logoimagehandler.ashx.b6031896.dll，其作用为动态编译代码并反射加载执行。

和探针工程相比，这是一个极为蹩脚的后门实现。首先代码隐藏的不够深入；其次，有一行遗留的调试信息未删除（图41）：

图41
编码中的clazz和\n（图42）暴露了开发者实际上是java出身的事实：

图42
C#在运行时没有class的概念，只有Type；微软一直坚持的分隔符也是\r\n。
但即使是这样，不可否认这是一个新型后门。我们可以大胆推测攻击者开发了一套对应的连接工具，用于对部署了此后门的站点进行管理。
原始安装包中自带的文件带有数字签名（图43），而我们发现的恶意样本（c15abaf51e78ca56c0376522d699c978217bf041a3bd3c71d09193efa5717c71）没有数字签名（图44）。

图43

图44
由此我们可以做一个大胆假设：该后门有可能是在针对某个目标的具体行动中由具体执行的成员临时修改logoimagehandler.ashx文件进行植入，最终DLL由Asp.Net动态编译生成。
一名不愿意透漏姓名（未登录）的用户于2020-11-24 19:55:35，通过WEB接口将恶意样本上传至VT，其来源显示为US（图45）。这表明可能来源于美国的某名安全分析师于2020年11月24日启动了SolarWinds供应链攻击事件的相关调查工作。

图45

0x40 行动

我们通过代码分析以及信息追溯，大致整理了部分信息作为描述攻击者的论据。

接下来要做的便是结合时间节点以及真实APT攻击的上下游环境，通过合理的推断，为攻击者描绘更为深入的画像。

0x41 周期

在0x32中我们已经通过开源情报搜集，结合DGA解密获取到了全部的时间信息，将时间信息按月份进行排列（图46），可得到下列结果：

图46
很明显，从四月初开始请求频次逐渐增加，七月份到达顶峰，九月份大幅降低，十月份近乎消失。
随后注意WHOIS变更记录，前文提到攻击者在2020/02/26 12:04:03 GMT对NS记录进行了初次变更（图47），这实际上可以标志着整个供应链打击行动的开始。

图47
最后，我们观察VT样本列表。如图所示，黄色部分表示样本中已存在相关类，但不存在恶意代码，红色部分表示为完整功能的样本。

我们可以看到攻击者早在2019/10/10便已经控制了代码仓库，由此逆推SolarWinds公司被控制的时间可能在2019年初甚至更早。
此次行动仅仅植入了一行包含无害信息的测试代码。我们推测此次意图是用于验证攻击手法是否能够成功，并据此推断后续行动所需时间。攻击者有可能控制了某个安装了SolarWinds的目标，或是本身即为SolarWinds客户的一员。总之，攻击者可以通过某种方式判断在接下来的软件更新中是否可以将恶意代码分发至下游。
通过2019年10月10日的测试之后，攻击者于2020/3/17成功在Solarwinds中植入了恶意代码，并在三、四、五三个月内接连在三个主要版本上发布了六次，这和DGA解密出的请求时间相匹配。
随后在八月份，SolarWinds接连发布了四个不存在恶意代码的版本，我们推测正是由于此版本的推出造成大量客户端更新，最终导致了请求量急剧下降。由于官方在本次事件披露之前并未发布任何安全公告，我们更倾向于认为这是攻击组织有预谋的撤退手段，而非实际意义上的权限丢失。
综合上述信息，我们可以总结出时间轴（图48）如下：

图48

0x42 疑问

显然，攻击者具有极强的耐心，以及有足够的技术支撑与基础设施支持。这样的攻击者不可能由业余人员临时组成。这些攻击者有着极强的专业能力以及高度的纪律性，本次事件绝不可能是第一次攻击活动，那么之前这些攻击者的目标是什么？

SolarWinds并非唯一的供应链提供商，事实上软件供应商多如牛毛，那么本次攻击是否仅仅是冰山一角？该组织在攻入SolarWinds之后看似“停滞”的时间中是否已经控制了其它更多的供应商？被供应链攻击打击

的目标同样可能扮演供应商的角色（无论是软件、服务亦或其他），攻击者是否已经通过本次攻击移动到了其它打击阵地？
同时供应链攻击的特性决定了此次攻击只属于完整APT链条的开始，后续必然跟随着更多的行动。根据已知信息，近几个月来至少有接近千家公司存在被控制的可能，这样大量的内网渗透行为由谁来开展，又会有多少人？
APT攻击的根本目的是用于将攻击转换为实际价值，无论是商业目的、金钱目的或是其他。或许有部分黑客的目的只是在炫技中得到无上的成就感，但本次攻击显然不是。本次攻击波及到包括政府、经济、金融、医疗等方方面面，攻击者的根本目的或倾向是什么？
最后，又是谁在指挥上述的一切？

0x43 推测

由于资料不足，上面的疑问依旧处于黑暗之中。我们希望在后续的持久关注中能够搜集到更多的信息，最终将关于此次行动以及背后组织的事实，完整的展现在公众面前。

但综合已知的种种，我们至少可以总结出以下信息：
1. 攻击者技术高超，具有非常熟练的APT行动基础框架设计能力以及对应项目的开发能力。
2. 攻击者手段老练，具有高度的耐心与隐藏能力。
3. 攻击者储备了大量的IP资源等基础设施，用于提供C2基础服务、敲门-控制阶段的出网策略配置等。
4. 攻击者存在专业支撑团队进行其他后门的开发、留存等操作。
5. 攻击者具有单独的后续攻击团队用于后渗透。
6. 攻击者在后渗透过程中具有单独的C2通道。
7. 攻击者具有在广泛攻击面中筛选、搜寻、处理、回传、消化有价值信息的能力。
8. 攻击者具有稳定的匿名身份与足够的财力，用于支撑上述基础设施与人员开销。
9. 攻击者具有统一的指挥机构设计整体战略，并协调上述全部战术行为。
最后，结合所有证据，终将指向以下推测：
SolarWinds供应链攻击（金链熊）事件，是一场由专业APT组织进行谋划，由基础建设团队提供链路与武器，由供应链团队经过至少两年的踩点与渗透，由分析团队进行目标确认，由后渗透团队实行深入控制，通过供应链打击的实质形式，有组织、有目的、成建制进行的网络攻击行动。

0x50 总结

由于域名已被微软进行接管并指向了黑名单（20.140.0.1），本次攻击行动实质上已经完全终止。限于信息的不足，对于该行动的分析至此告于段落。
但这不是终点，本次的攻击者如同老练的猎手，在黑暗中一步步谨慎的筹划并实施这一切。从本次分析的结论来看，攻击者体现出了极强的纪律性：明确放弃了至少30%以上的目标。这完全不符合一般的以获利为主要目的商业黑客行为，也有别于一些缺乏明确战略情报意图驱动的普通行为体。
纵观本次暴露的种种迹象，无论是控制SolarWinds、实施供应链攻击或是具体目标的后渗透行为均表现出了明显的非线性作业模式，表现出的综合能力、跨领域知识，令人惊叹。我们推测这背后是在某个指挥机构的领导下，按照能力与机会导向的行动模式，按部就班有条不紊的实施攻击。同时不排除其背后有相关的科研机构、研发中心支持，及与情治机构的信息共享的可能。

具有如此能力水准的攻击组织必然会执行其他的行动。事实上纵观已被披露的APT攻击历史，一个被披露的行动背后往往是十个未被发现的行动，对此需要长期关注，并有意识的根据技术特点、战术行动导向及可能的战略意图进行归类。
其次，样本分析作为安全从业者必不可缺的技能，实质上是多种技能的结合体。没有实战行动经验的分析人员不可能领略攻击者的真实意图，没有代码功底的分析人员不可能完全看到代码设计背后暴露出的思想。样本分析工作绝对不是对照各类日志和官方文档说明，就能做好的事情，类似由专业组织策划并实施的行动，只有同样老练的专业人员才能进行对抗，并在蛛丝马迹中寻觅到被层层遮掩的真相。
每一位安全从业者都应当更多的接触真实攻击的内容，并将其与已知攻击进行对照，逐步形成自己的攻防体系。只有这样才能更贴合实际，提升自己的能力水准，最终在对抗中，真正将“以攻促防”进行落地。
没有人愿意处于危机四伏的世界，而在虚拟世界中进行破坏的成本要远比现实世界更低。
让网络更安全，让世界更美好。
不忘初心，方得始终。
zcgonvh@A-TEAM、rem4x@A-TEAM于庚子年冬月初八戌时一刻

相关工具：
https://github.com/QAX-A-Team/sunburst_decoder
招聘：

【奇安信 A-TEAM武器化方向人员招聘】

工作地点：北京-奇安信安全中心

岗位：

红队工程师（武器研发方向）

职责：

1.研究能对实战起到实质推进作用的已有技术。

2.发掘能对实战起到实质推进作用的新技术。

3.对上述技术施行工程化落地。

要求：

1.品格过关，性格稳重，思想坚定。

2.具有较强的学习能力、理解能力、接受能力与抗挫折能力。

3.在对抗方向的安全研究具有浓厚兴趣。

4.熟悉真实环境渗透流程，对于对抗有一定理解。

5.熟悉C/C#语言，或接受从其他语言进行转变。

加分项：

1.有开源项目。

2.对代码工程、设计模式、编码规范有一定理解。

3.对windows、linux操作系统有一定理解。

想要在红队武器方向发展，打磨研究实战武器的小伙伴发送简历至邮箱：

zhangchongyao@qianxin.com

zcgonvh@qq.com

或者私聊@zcgonvh

头像哥带你搞事情。
