URLZone：疑似针对日本高科技企业雇员的攻击活动分析

By 360威胁情报中心

Archived: 2026-04-05 14:36:58 UTC

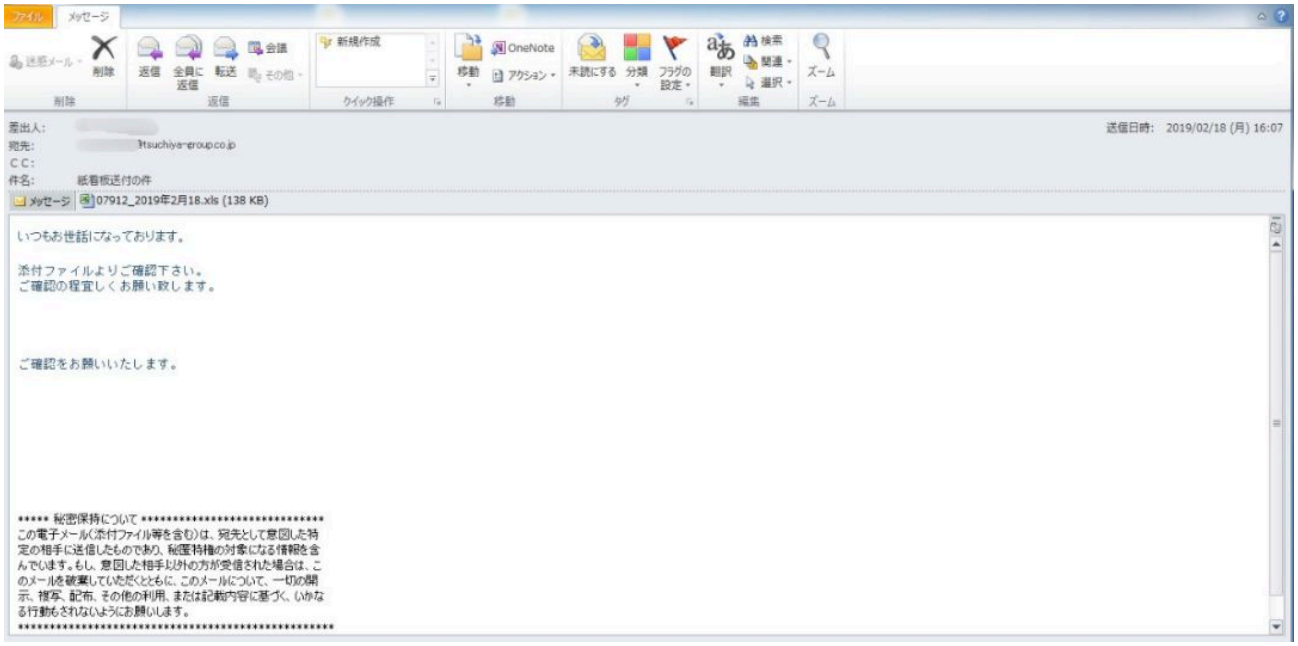
背景

近期，360威胁情报中心捕获到多个专门针对日本地区计算机用户进行攻击的诱饵文档，文档为携带恶意宏的Office Excel文件。通过分析相关鱼叉邮件的收件人信息，我们发现受害者均为日本高科技企业雇员。从攻击的定向性、受害者分布及过往相关背景信息来看，攻击者主要目的是为敛财，同时也不排除其有窃取商业机密和知识产权的可能性。

诱饵文档内的恶意宏代码及后续的PowerShell脚本会调用多个与系统语言区域相关的函数，并依赖于函数的返回值解密后续代码，从而实现专门针对日文系统使用者的精确投递。比如通过判断货币格式化后的长度、使用本机的LCID[1]（LanguageCode Identifier）作为异或解密的密钥等方式来区分是否为日本地区的计算机。攻击者最终通过图片隐写技术下载并执行URLZone[2]，并在随后的代码中进一步检测运行环境，以避免在沙盒、虚拟机以及分析机上暴露出恶意行为。

攻击邮件

360威胁情报中心在进行关联分析后，找到了十余封相关的邮件，针对佳能、藤田医科大学、土屋工业公司、Fujikin等公司和机构的雇员。与以往的攻击类似[3]，邮件内容由日文组成，并通过伪造来自日本的发件人以增加被打开的概率。



邮件正文内容并不具有针对性：

样本分析

VBA Macro

文件名	67874_2019年2月18.xls
MD5	b158d69db6ef7110ef4308f4b0749e0f
文档作者	ユーザー

捕获到的该诱饵文档是一个Office Excel文件，其内嵌VBA宏。打开该文件会展示含有日文的图片来诱使目标启用宏，其内容如下：

当受害者启用宏后，将自动执行恶意宏代码。我们在分析过程中提取的宏代码如下图所示：

宏代码在Workbook_Open函数一开始就对运行环境进行了检测与过滤，当Select_t的长度与msoContactCardTypeUnknownContact不一致时就执行退出操作：

msoContactCardTypeUnknownContact是一个Office内置的常量，在MSDN文档中[4]其值定义为2：

因此为了保证后续代码的执行，Select_t的长度也应该是2，而该变量的值与货币类型相关：

在日文操作系统中，将0格式化为货币后是“¥0”，故其长度值满足判断要求。然而在其它类型操作系统中，其结果的长度值很可能不满足要求。比如在中文操作系统下，格式化后的值是“¥0.00”，因此程序到此就退出结束了。

接下来的代码会解密一串CMD命令，而其解密方法中也会使用到上述代码。这会造成在很多非日文系统环境下解密数据不完整，从而影响后续执行：

正确解密后，最终将执行一系列CMD命令，把PowerShell代码写入环境变量ladnI以及相关的启动代码写入Zgo环境变量，并随后执行：

Downloader（PowerShell Scripts）

PowerShell的执行分两个阶段。首先执行VBA宏所解密出来的PowerShell脚本，从硬编码的URL地址下载PNG图片，随后提取隐藏的数据，进行解密、解压缩后得到第二阶段的PowerShell脚本。第二阶段的PowerShell脚本使用类似的方式获得URLZone恶意程序，并通过反射加载的方式在内存中直接运行。PowerShell脚本在执行过程中采用多种手段获取语言区域相关的常量值，并将结果与解密关联起来，从而使其只能在日文操作系统中正确运行。

阶段1：

VBA宏解密后得到的是经过混淆的PowerShell脚本，对关键代码去混淆后的程序逻辑将变得更加清晰，部分处理后的代码如下图所示：

脚本首先会加载一个Base64编码的PE程序（MD5：61120c989de3d759c1a136ffd91e59d2）到内存。该PE程序用于获取Windows的语言信息，反编译后的代码如下图所示：

PowerShell脚本通过调用PE程序的DezA方法，得到当前系统的语言代码（简体中文系统是CHS，英语是ENU，日文是JPN），该语言代码随后将被用于生成AES解密的密钥。

随后脚本尝试从硬编码的URL下载PNG图片文件：

hxxp://imagehosting.biz/images/2019/02/14/in1.png

hxxp://images2.imagebam.com/f1/b1/50/dd7e561126561184.png

hxxps://mger.co/img/w84vm.png

hxxps://images2.imgbox.com/34/60/1Zc8BevK_o.png

下载的图片如下所示：

由于大部分图片像素RGB中绿色（G）和蓝色（B）的低4位被用于存储加密压缩后的信息，这些值的改变对图片空白区域外观的影响更为明显：

脚本从图片前360x291像素矩阵中，依次提取每个像素绿色（G）和蓝色（B）的低4位，并分别作为高4位和低4位形成一个字节。接下来取前104644字节大小的数据，附加到字符串“4B2T”后得到最终的待解密数据。

CrowdStrike分析文章中[5]提到的相关Python代码经修改后便可用于提取该图片中的隐藏数据：

待解密数据经过Base64解码、AES解密和gzip解压缩三步处理后，将得到编码后的第二层PowerShell脚本。AES在解密时，将使用之前获得的系统的语言代码（JPN）和Base64解码后的前32字节生成解密密钥，用于解密偏移52字节以后的数据。解密后的第一个字节将与0x1F（gzip首字节数据）做比较，相同时才对其进行解压缩操作。因此，在非日文操作系统上直接运行该脚本，是无法正确地解密和解压缩数据的，因此也就无法执行第二阶段的PowerShell脚本。相关代码如下：

阶段2：

从图片中解出来的PowerShell脚本在运行时首先判断处理器架构，以便兼容x86和x64平台：

接下来会加载另外一段二进制数据到内存，该段数据还原出来后是一个DLL模块（MD5：f0c47667c50cf18c66094094b44627ba），主要用于gzip解压缩：

接下来的大部分代码来自开源项目PowerSploit 中的Invoke-ReflectivePEInjection.ps1模块[6]，主要功能是在内存反射加载${GloBal:MGGG}变量中的数据：

设置该变量的代码也被同样方式混淆，去混淆后的代码如下：

这段代码同样首先尝试从内置的URL地址下载图片：

hxxp://oi68.tinypic.com/2saxhrc.jpg

hxxps://thumbsnap.com/i/aqiAmg1b.png?0214

hxxps://i.postimg.cc/0jFwGVb3/l1.png

下载后的图片如下图所示：

图片中大部分像素RGB中绿色（G）和蓝色（B）的低4位同样被用于存储隐藏信息，方式也与第一阶段的PowerShell脚本一致，只是像素选取和数据长度略有区别。这里提取图片前195x236的像素矩阵中的隐藏信息，以同样的方式进行转换，然后获取转换后数据的前46000字节。随后在尾部附上内置的Base64数据，相关代码如下：

接下来对这段数据进行Base64解码及异或解密，解密的Key与Windows操作系统的LCID[1]（LanguageCode Identifier）相关。代码把本机LCID转换为字节数组后，作为Key与Base64解码后的数据进行异或解密：

解密后的数据经过Base64解码后会用gzip进行解压缩，从而得到URLZone（MD5：ddc16b26c2cd6f8d157bed810bf944f4）。解码和解压缩方式与系统的MajorVersion有关，该值为6则直接在PowerShell脚本中进行，否则就调用加载的DLL模块来完成。URLZone最终将被反射加载到内存中执行。

Payload（URLZone）

文件名	内存加载，文件不落地
MD5	ddc16b26c2cd6f8d157bed810bf944f4

主要功能

URLZone在运行时首先对API进行动态加载，计算API名称校验和的代码如下，其计算方式与APT-28[7]和TelsaCrypt[8]采用的方法一致：

随后对加密的字符串进行两层异或解密，部分解密后的字符串内容如下：

接下来将通过多种方式对运行环境进行检测，以防止被调试以及避免在虚拟机、沙箱和分析人员计算机上运行：

1. 检测处理器类型，从而避免在至强处理器上运行
2. 通过查找模块“SbieDll.dll”来检测沙盒
3. 通过GetTickCount函数计算代码片段运行时间来检测调试器
4. 通过查询注册表来检测VirtualBox虚拟机
5. 调用SetupDi系列函数来检测VMware
6. 检测启动路径是否包含“sample”、“virus”、“sandbox”等字符串，以避免在沙箱或分析人员计算机上运行
7. 调用IsDebuggerPresent来检测调试器
8. 循环调用rdtsc指令来计算程序片段的累计执行时间，确保其结果在期望范围之内，以避免被调试或在模拟器中运行

当所有的运行环境检测通过后，URLZone将以挂起状态启动新的explorer.exe进程，把自身映射到该进程中新创建的区段。随后解密位于RVA 0x92E0处的shellcode，用以覆盖explorer进程的入口代码，以便执行注入后位于RVA 0x8AFC处的代码：

Explorer进程中执行Shellcode转到新的入口代码：

注入代码在执行时会解密位于RVA0x218C的配置文件，解密后的配置文件中包含硬编码的C2地址：panisdar.com/auth/

接下来将搜集本机信息并做一系列检查，包括操作系统版本、进程权限以及64位环境，修改ACL、写入木马安装时间到注册表，相关代码如下图所示：

搜集的信息会被RSA加密，加密所需的公钥位于RVA地址0x2346，相关值如下：

-----BEGIN PUBLICKEY-----

BgIAAACkAABSU0ExAAQAAAEAAQBxQGcZ/br3h0ZKC2scS7WJ6Je4pOB5TOvOAe6mkPQ5zSruIlx

kPpr9PLLGfuTEmAqVTBunRk0+ZpvnAXfHbTSYOD/bfmqtaE1pEeci8jjukAZZwCX+rWoh2O0ymX

2v4rLbEqiC7apbg+ughDvLEjkvkx2FBoHU55ALmfldAtPLnA==

-----END PUBLICKEY-----

随后获取本地IP、键盘布局等信息，并填入URL参数对应字段。调试过程中形成的参数如下：

tver=1208743166&vcmd=0&cc=0&hh=00000000&ipcnf=10.0.2.15+&sckport=0&pros=0&keret=04090409;&email=

接下来通过HTTP POST请求将数据发送到C2。在调试时该样本未能从C2获得有效数据，因此接下来会进入DGA环节，尝试连接更多的域名。根据已有的公开报道，URLZone会下载执行Cutwail、Ursnif[9]以及Gozi ISFB[5]。

DGA

该样本的DGA算法（RVA：0x625C）与之前分析文章中[10]描述的方法非常相似，同样需要使用当前的C2域名和字符串“qwertyuiopasdfghjklzxcvbnm123945678”来进行新域名的生成。不同的是，这个样本在生成DGA的过程中，还会使用到一个类CRC值，用于异或sum_of_chars：

该值在计算时会用到CRC查找表，用以计算RSA公钥的类CRC值，相关Python代码如下：

该值计算出来后，已有的DGAPython代码[10]只需要作少量改动便可用于生成此样本对应的域名：

总结

URLZone从2015年底就开始发起针对日本用户的攻击[3]，不过之前只是在邮件及附件内容中采用日语，且冒充邮件来自日本的用户。360威胁情报中心捕获的这轮攻击中不仅保留了其初始手法，攻击者还在诱饵文档相关的宏代码和PowerShell脚本中采用了多种手段，让其仅能在日文系统中正常运行：

1. 判断货币格式化后的长度，同时用该值计算待解密数据的起始位置
2. 使用当前系统的语言代码值来生成AES解密密钥
3. 使用本机的LCID（Language Code Identifier）作为异或解密的密钥

此外还采用了图片隐写、DGA、RSA加密等多种手段，来躲避防火墙等安全软件的检测，并以多种方式检测沙盒、虚拟机、调试器等环境，以避免暴露恶意行为。虽然这很可能只是针对用户的网银木马类攻击，但其相关手法也可以被有针对性攻击所采用。

相对于使用Office0day，利用恶意宏进行攻击需要更多的用户交互以完成攻击。虽然这会降低其攻击的成功率，但仍具有很好的成本优势，因此仍被许多攻击组织大量采用。企业用户应尽可能小心打开来源不明的文档，如有需要可通过打开Office Word文档中的：文件-选项-信任中心-信任中心设置-宏设置，来禁用一切宏代码执行。

目前，基于360威胁情报中心的威胁情报数据的全线产品，包括360威胁情报平台（TIP）、天擎、天眼高级威胁检测系统、360 NGSOC等，都已经支持对此类攻击的精确检测。

IOC

MD5 (Email)
165587b4de646744fd685fdccad483aa
83599a1ac098a6139eb2329040da64f0
aa54935d07d2f3f120484095e3a622e9
c9fe46a97f382f5507a137b55aa9a180
d1eb688573524b62eac643184afe14f7
d388b03e657a21251383de725f4602a2
d6aee99594fafd6293cb3dff71e1710a
e7e3581b38de0054d5ec67009b07208a
ec617c9083f6e02cb9ab32a45a3ced3b
f325516686b6096224c0ef66cecb6e28
fef3e566e2bc7a520f423a223970af95
MD5 (Attachment)
a9dca658ba431a4123be8aa3f13284bc
b158d69db6ef7110ef4308f4b0749e0f

c909568a2dce7a3214a6f2e131a74f9c
MD5 (PNG)
dd7e569e55b7cd8b6b2ed266a8e56f97
5ce3d93453a5af55577da49236ae887d
285d70d4e25d9f68ef165189d8af55e0
URL
hxxp://imagehosting.biz/images/2019/02/14/in1.png
hxxp://images2.imagebam.com/f1/b1/50/dd7e561126561184.png
hxxps://mger.co/img/w84vm.png
hxxps://images2.imgbox.com/34/60/1Zc8BevK_o.png
hxxp://oi68.tinypic.com/2saxhrc.jpg
hxxps://thumbsnap.com/i/aqiAmg1b.png?0214
hxxps://i.postimg.cc/0jFwGVb3/l1.png
CC地址
panisdar.com

参考链接

[1]. https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-lcid/63d3d639-7fd2-4afb-abbe-0d5b5551eef8

[2]. https://www.virusbulletin.com/virusbulletin/2012/09/urlzone-reloaded-new-evolution

[3]. https://www.fireeye.com/blog/threat-research/2016/01/urlzone_zones_inon.html

[4]. https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2010/ff861819(v=office.14)

[5]. https://www.crowdstrike.com/blog/cutwail-spam-campaign-uses-steganography-to-distribute-urlzone/

[6]. https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

[7]. https://github.com/Neo23x0/signature-base/blob/master/yara/apt_grizzlybear_uscert.yar

[8]. https://github.com/pan-unit42/public_tools/blob/master/teslacrypt/deobfuscate_api_calls.py

[9]. https://threatvector.cylance.com/en_us/home/threat-spotlight-urlzone-malware-campaigns-targeting-japan.html

[10].https://www.johannesbader.ch/2015/01/the-dga-of-shiotob/
