

# Duqu 2.0

## 技术分析

版本 2.0，2015 年 6 月 9 日



# Duqu 2.0 技术分析

非官方中文译文·安天技术公益翻译组 译注

文档信息			
原文名称	The Duqu 2.0 Technical Details		
原文作者	全球研究和分析团队	原文发布	2015 年 6 月 9 日
译文首发	2015 年 6 月 12 日	当前译本	2015 年 6 月 24 日
作者简介	<p>卡巴斯基实验室的全球研究和分析团队（GReAT）是公司最重要的资产之一，由最顶尖的安全研究专家们分析最新及最先进的网络威胁。GReAT 创建于 2008 年，保证了卡巴斯基在反恶意软件研究及创新领域的领导地位。</p> <p><a href="http://zh.wikipedia.org/zh-tw/%E5%8D%A1%E5%B7%B4%E6%96%AF%E5%9F%BA%E5%AF%A6%E9%A9%97%E5%AE%A4">http://zh.wikipedia.org/zh-tw/%E5%8D%A1%E5%B7%B4%E6%96%AF%E5%9F%BA%E5%AF%A6%E9%A9%97%E5%AE%A4</a></p>		
发布单位	卡巴斯基实验室		
原文出处	<a href="https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf">https://securelist.com/files/2015/06/The_Mystery_of_Duqu_2_0_a_sophisticated_cyberespionage_actor_returns.pdf</a>		
译者	安天技术公益翻译组	校对者	安天技术公益分析组
免责声明	<ul style="list-style-type: none"><li>本译文译者为安天实验室工程师，本文系出自个人兴趣在业余时间所译，本文原文来自互联网的公共方式，译者力图忠于所获得之电子版本进行翻译，但受翻译水平和技术水平所限，不能完全保证译文完全与原文含义一致，同时对所获得原文是否存在臆造、或者是否与其原始版本一致未进行可靠性验证和评价。</li><li>本译文对应原文所有观点亦不受本译文中任何打字、排版、印刷或翻译错误的影响。译者与安天实验室不对译文及原文中包含或引用的信息的真实性、准确性、可靠性、或完整性提供任何明示或暗示的保证。译者与安天实验室亦对原文和译文的任何内容不承担任何责任。翻译本文的行为不代表译者和安天实验室对原文立场持有任何立场和态度。</li><li>译者与安天实验室均与原作者与原始发布者没有联系，亦未获得相关的版权授权，鉴于译者及安天实验室出于学习参考之目的翻译本文，而无出版、发售译文等任何商业利益意图，因此亦不对任何可能因此导致的版权问题承担责任。</li><li>本文为安天内部参考文献，主要用于安天实验室内部进行外语和技术学习使用，亦向中国大陆境内的网络安全领域的研究人士进行有限分享。望尊重译者的劳动和意愿，不得以任何方式修改本译文。译者和安天实验室并未授权任何人士和第三方二次分享本译文，因此第三方对本译文的全部或者部分所做的分享、传播、报道、张贴行为，及所带来的后果与译者和安天实验室无关。本译文亦不得用于任何商业目的，基于上述问题产生的法律责任，译者与安天实验室一律不予承担。</li></ul>		

## 感言与鸣谢

首先要感谢网友 Sandman, Joe 和 Arcadia 三位老师对译文勘误的大力支持，他们的指导和帮助，让我们知道了在翻译中有很多低级的失误，这既让我们觉得惭愧，又为有向诸位老师学习的机会而感到幸福。

我们将之前比较严重的误译做了一个勘误列表。如译文初稿有误导之处，我们诚挚道歉，如果发现有新问题，欢迎在创意安天论坛“安天公益翻译”版回帖指出。我们会继续更新。

翻译组更多的同事是文科生，尽管我们也在工程师岗位，但基本都不从事编码和分析的工作。安天公益翻译组已经很长时间没有翻译长篇技术文献了，而“The Duqu 2.0 Technical Details”无疑是一篇殿堂级的长篇分析报告，不得不说，小组的小姐妹们对这个是有些打怵的。此时更显出三位老师友情支持的可贵。

当年安天翻译组产生的原因，其实是 seak 不懂英文，导致公司外语盲的问题盛行。希望通过持续的文献翻译改善工程师的视野，但似乎内部学习外语、阅读原文文献的动力也因此更加不足，也许 seak 带头学习英语的时间到了，也许是姐妹们帮助兄弟们更好学习外语的时候了。

作为一个保持着传统反病毒行业正直而刻板风格的团队，我们对卡巴的技术能力与深度一直保持着良好印象。“The Duqu 2.0 Technical Details”让我们看到，无论是技术深度还是直面威胁的勇气，他们依然是业内的榜样厂商。而翻译完这样一篇文档，我们依然会回忆那种技术与逻辑的美感。

他们在报告中写道：“对于一家安全公司来说，最困难的事情之一就是承认自己沦为了恶意软件攻击的受害者。卡巴斯基实验室坚信信息透明的力量，这就是为什么我们在此发布了攻击信息。对于我们来说，用户的安全仍然是重中之重，我们将继续努力以维护您的信任和信心。”

安天市场部的同事曾这样评价：这段话令我们久久不能平静——勇士锋芒来自战场的洗礼，他不会刻意掩盖自己的伤口，更绝不会在伤疤上打上一层粉底，这样的厂商令人肃然起敬。

安天技术公益翻译组

2015 年 6 月 29 日

勘 误 表

序 号	原 页 数 / 行 数	新 页 数 / 行 数	更 改 前	更 改 后
1	3/4	3/4	攻 击 者	威 胁 源
2	3/12	3/15	CVE-2035-2360	CVE-2015-2360
3	4/21	4/19	域 名 管 理 员	域 管 理 员
4	5/9	5/10	再 内 存 中	在 内 存 中
5	6/11	6/12	独 特 的	独 有 的
6	6/13	6/14	内 存	驻 留 内 存
7	6/17	6/18	有 效 载 荷	载 荷
8	7/11	7/12	代 码 页	内 码 表
9	8/1	8/2	块	数 据 块
10	11/5	11/6	该 段 的 名 称	该 共 享 内 存 区 的 名 称
11	13/6	13/6	这种注册的实体被认为是值得信任的	杀毒产品认为这个注册过的项目是受信任的
12	14/2	14/2	存 货	存 活
13	14/15	14/14	非 权 限 用 户	普 通 权 限 的 用 户
14	14/17	14/18	漏 洞	漏 洞 利 用 代 码
15	15/25	15/26	进 程 口 令	进 程 令 牌
16	17/9	17/8	它是基于后门的简单命名管道，听取进程通信。	基于简单命名管道的后门监听来自协调器的通信
17	17/14	17/12	能够包含上百种不同的针对多样的网络监听函数的模块	包含数十种不同的模块，能够提供各种网络间谍功能
18	18/2	17/26	激 活 桌 面 并 终 止 会 话	活 跃 桌 面 和 终 端 会 话
19	18/5	18/2	Windows 基于插口的运输	基于 Windows 套接字的传输
20	18/7	18/4	当 Windows 安装器启动恶意安装包时，MSI 定制行为库就被激活了	当恶意安装包被 Windows Installer 启动后，MSI CustomAction 扩展库就会被激活
21	18/17	18/14	“ /LG/HM/ ” 的 目 录	“ /LG/HM/ ” 文 件 夹 的 文 件
22	19/8	19/5	网 络 和 域 名	网 络 和 域
23	21/21	21/19	辅 助 历 史	帮 助 历 史 记 录
24	22/2	22/1	功 用 DLL	实 用 工 具 DLL
25	22/11	22/10	登 记	层 级
26	22/12	22/11	全 程 可 视 的	全 局 可 见 的
27	30/9	30/9	枚 举 windows	枚 举 窗 口
28	44/15	44/15	同样也是网络军备竞赛升级如此之快的信标	同时也标志着网络军备竞赛迅速升级

## 目录

执行摘要.....	3
初始攻击.....	4
横向运动.....	4
Duqu 2.0 MSI 安装包的分析.....	7
文件属性.....	7
第一层：ActionDLL ( msi.dll ) .....	10
第二层：ActionData0.....	10
第三层：klif.dll .....	11
攻击 AVP.EXE .....	12
CTwoPENC.dll 0 day 漏洞和 KMART.dll.....	14
载荷容器和迁移.....	15
载荷类型 “L” .....	15
载荷运行类型 “G” .....	16
载荷运行类型 “I” .....	16
载荷运行类型 “K” .....	17
载荷运行类型 “Q” .....	17
插件化模块平台.....	17
持续性机制.....	33
命令和控制机制.....	33
“portserv.sys” 驱动程序分析 .....	35
Duqu 和 Duqu 2.0 之间的相似性 .....	37
Duqu 2.0 的受害者 .....	42
归属.....	43
结论.....	44
参考文献.....	45

## 执行摘要

在今年初的一次安全扫描中，卡巴斯基实验室检测到了一起牵涉多个内部系统的网络入侵行动。

沿着这一发现的线索，我们开展了大规模的调查，进而发现了一个新的恶意软件平台。此恶意软件平台来自最富有技术能力、最神秘也最强大的 APT 组织中的一员——Duqu。Duqu 威胁源在 2012 年归于沉寂，业界也一度认为 Duqu 组织已经停止了攻击作业项目，直到最近发生的事件才揭示该组织依然活跃。我们的技术分析表明，在这一轮新的攻击中出现了 2011 年臭名昭著的 Duqu 恶意软件<sup>1</sup>的升级版（Duqu 恶意软件也被称为 Stuxnet 蠕虫的“继兄弟”<sup>2</sup>）。我们将这一新的恶意软件及其对应的平台命名为“Duqu 2.0”。

Duqu 2.0 的受害者遍布世界各地，包括西方国家、中东和亚洲地区。攻击作业者出于提高网络能力的目的，不仅攻击了直接针对的最终目标对象，还攻击了对实现其目的有价值的“功利目标对象”（功利目标对象：攻击者为了提高自己的网络能力而攻击的目标）。

最值得注意的是，在 2014 年至 2015 年的时间段内恶意代码的一部分新感染对象涉及伊核六方会谈（P5 + 1，译者注：“P5+1”组织成员国包括美国、俄罗斯、中国、英国、法国和德国），以及伊核六方会谈的一些场所。Duqu 背后的威胁源似乎倾向于对涉及此类高层次会谈的场所发动攻击。除了伊核六方会谈，Duqu 2.0 组织还针对奥斯维辛-比克瑙集中营解放 70 周年<sup>3</sup>的纪念活动发动了类似攻击。

在卡巴斯基实验室的攻击案例中，攻击者利用了一个 Windows 内核中的 0 day 漏洞（CVE-2015-2360，微软于 2015 年 6 月 9 日修复了该漏洞）以及另外两个目前已被修复的漏洞（当时它们也属于 0 day 漏洞）。

---

<sup>1</sup> <https://en.wikipedia.org/wiki/Duqu>

<sup>2</sup> [http://www.kaspersky.com/about/news/virus/2011/Duqu\\_The\\_Step\\_Brother\\_of\\_Stuxnet](http://www.kaspersky.com/about/news/virus/2011/Duqu_The_Step_Brother_of_Stuxnet)

<sup>3</sup> <http://70.auschwitz.org/index.php?lang=en>

## 初始攻击

针对卡巴斯基实验室的初始攻击，以我们在亚太区的一个较小规模分支机构的某个员工为目标。Duqu 2.0 的初始感染向量目前未知，但是我们怀疑基于电子邮件的鱼叉式钓鱼攻击手段发挥了重要作用。这是因为其中一个“零号病人”（第一批感染源）的邮箱和网络浏览器的记录被清除了，推测其目的是隐藏攻击痕迹。鉴于相关的各台机器都打了完整的安全补丁，我们相信攻击者使用了一个 0 day 漏洞完成初始攻击。

2011 年，我们发现了 Duqu 攻击使用了包含 0 day 漏洞（CVE-2011-3402，依赖于一个恶意的嵌入式 True Type 字体文件）利用代码的 Word 文档。该漏洞允许攻击者通过 Word 文档直接攻入内核模式，这是一种非常强大而且极为罕见的攻击技术。2014 年 6 月，类似的攻击技术和 0 day 漏洞（CVE-2014-4148<sup>4</sup>）利用代码又出现在针对某知名国际组织的攻击中。2014 年的这些攻击所使用的 C&C 服务器和其他因素与 Duqu 有一定的相似性，但是所使用的恶意软件与 Duqu 和 Duqu 2.0 都有所不同。可能的情况是：这是 Duqu 组织在同时发起的另一个攻击作业项目，而同样的 0 day 漏洞（CVE-2014-4148）也被用于安装 Duqu 2.0。

一旦攻击者成功感染一台机器，他们就会进入下一个阶段。

## 横向运动

在一般情况下，一旦攻击者成功获得网络访问能力，攻击行为会进入以下两个阶段：

- 侦察和识别网络拓扑
- 横向运动

在 Duqu 2.0 的情况下，攻击者似乎利用了另一个 0 day 漏洞（CVE-2014-6324，该漏洞于 2014 年 11 月由微软 MS14-068<sup>5</sup>安全公告修复）实现横向运动。该漏洞利用代码允许普通权限的域用户将访问凭证提权至域管理员帐户。虽然我们无法获得该漏洞利用代码的样本，但是操作系统日志中记录的事件，与微软给出的该漏洞检测指引相匹配。我们还发现，恶意代码的模块在本地网络中开展了“哈希传递”攻击（译者注：哈希传递是指用户登录的时候使用密码的哈希值代替密码来完成认证，因此攻击者可以在未获得明文口令的情况下，使用哈希值完成多种 Windows 协议的认证），使攻击者能够使用多种不同方式实现横向运动。

一旦攻击者获得了域管理员权限，他们就能够利用这些权限感染该域中的其他计算机。

为了感染该域中的其他计算机，攻击者使用几个不同的策略。在我们发现的大多数攻击中，攻击者制作包含恶意代码的微软 Windows Installer 安装包（MSI），然后将其远程部署至其它机器。为了启动包含恶意代码的 MSI 安装包，攻击者通过下述命令行指令在目标机器中创建一个服务：

---

<sup>4</sup> <https://www.fireeye.com/blog/threat-research/2014/10/two-targeted-attacks-two-new-zero-days.html>

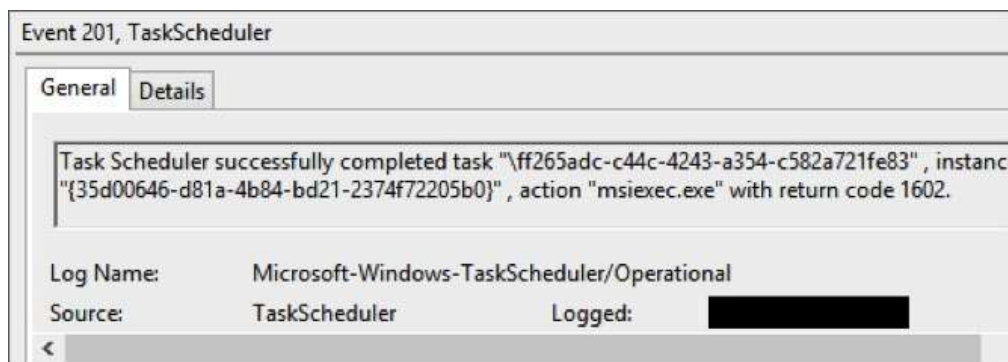
<sup>5</sup> <https://technet.microsoft.com/library/security/MS14-068>



```
msiexec.exe /i "C:\\[...]\tmp8585e3d6.tmp" /q PROP=9c3c7076-d79f-4c
```

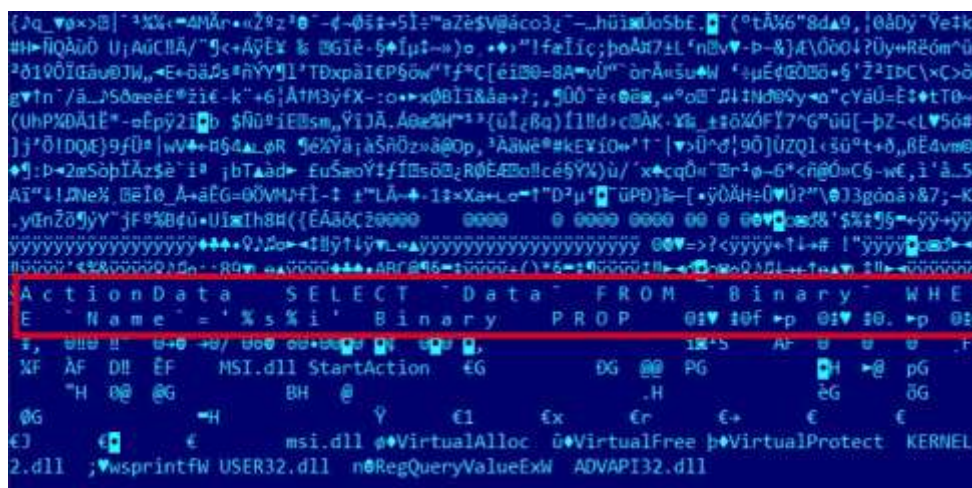
PROP 值被设定为 56 位随机密钥,用于解密安装包中的主要载荷。该参数的其它已知名称包括“HASHVA”和“CKEY”。在各个案例中,安装包所在的文件夹各不相同,这取决于攻击者在远程计算机所能够访问的文件夹位置。

除了通过创建服务来感染局域网中的其它计算机,攻击者还可以使用 Task Scheduler (任务计划) 程序来远程启动“msiexec.exe”进程。赛门铁克在技术分析报告<sup>6</sup>提到,在 2011 年的版本中,Duqu 感染过程中也使用 Task Scheduler 进行横向运动。



“msiexec.exe”: 日志中的 Task Scheduler 痕迹

攻击中使用的 MSI 文件都包含被用作加载器的一个恶意存根 (stub) 程序。该 stub 程序从 MSI 文件中加载并解密其他恶意代码资源,然后在内存中执行解密后的恶意代码。



从 MSI 文件 (红色标注) 加载其他资源的恶意存根

在各个案例中,安装包所用的加密算法都有所不同。需要指出的是,攻击者非常谨慎,在每次攻击中使用独特的方法、加密算法和名称 (如文件名)。采用这种方式,不仅能够规避安全产品的检测,还可以在某一样本被发现后能够限制杀毒软件公司定位其它被感染点的能力。

<sup>6</sup> [http://www.symantec.com/content/en/us/enterprise/media/security\\_response/whitepapers/w32\\_duqu\\_the\\_precursor\\_to\\_the\\_next\\_stuxnet.pdf](http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf)



到目前为止，我们发现攻击者使用了以下加密算法：

- Camellia
- AES
- XTEA
- RC4
- 不同的多字节 XOR 加密技术

对于压缩算法，我们发现了如下几种：

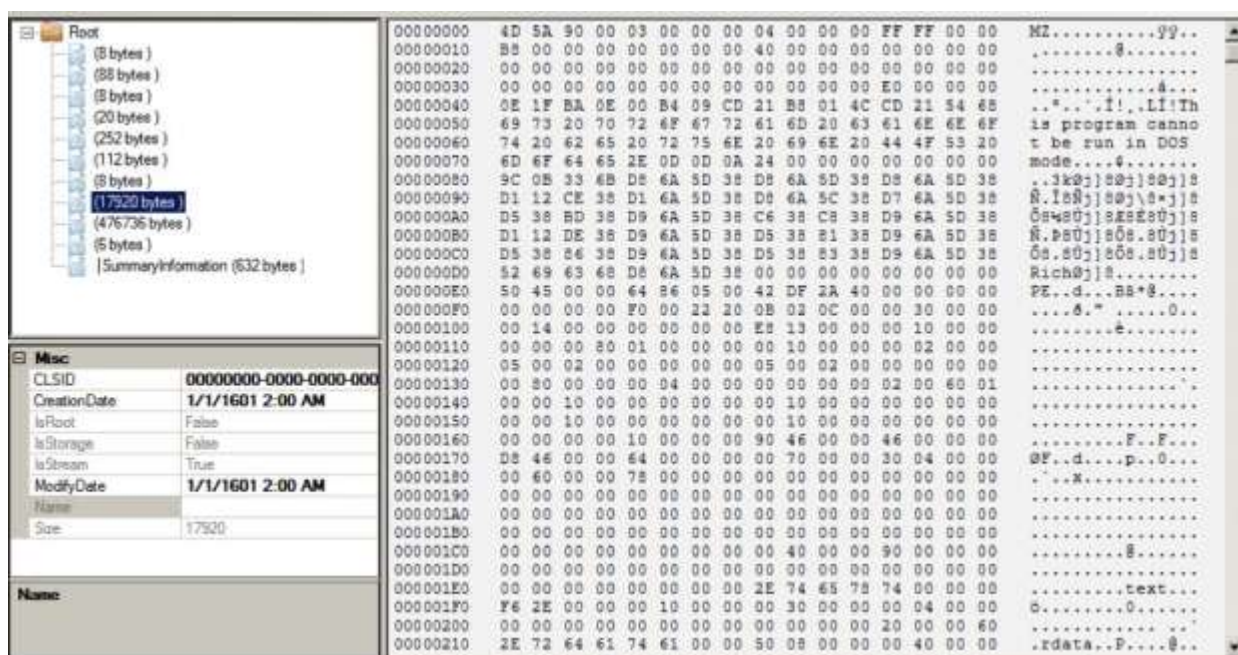
- LZJB
- LZF
- FastLZ
- LZO

大体上，每个编译后的攻击平台都使用独有的加密和压缩算法组合，因此对其检测变得非常困难。

攻击者可以在受害者机器中部署两种类型的安装包：

- “基础的” 驻留内存远程后门（约 500K）。
- 全功能并支持 C&C 控制的驻留内存网络间谍平台（18MB）。

它们具有相似的结构，如下所示：



恶意 Duqu 2.0 MSI 安装包

如上截图所示，包含加载器（ActionDll：17,920 字节）和主要载荷（ActionData0：476,736 字节）。在安装包被执行时，ActionDll 首先被加载，然后 ActionDll 的唯一导出函数 StartAction 执行。

恶意代码通过域控制器向该域内的其它计算机定期推送“基础的”驻留内存远程后门程序，其行为与蠕虫感染非常相似。这使得攻击者能够进入该域中的大多数机器，而且在攻击者需要开展进一步的攻击行为时，他们可以上传一个更复杂的 MSI 文件来部署不同的恶意代码插件用于获取信息，而目前发现了数十欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

种这样的恶意代码插件。

接下来，我们详细介绍“基础的”驻留内存远程后门的加载机制。

## Duqu 2.0 MSI 安装包的分析

文件名：随机 / 各个案例取值不同

MD5 ( 示例，具体案例的取值会有改变 )：14712103ddf9f6e77fa5c9a3288bd5ee

大小：503296 字节

### 文件属性

MSI 文件具有以下一般属性：

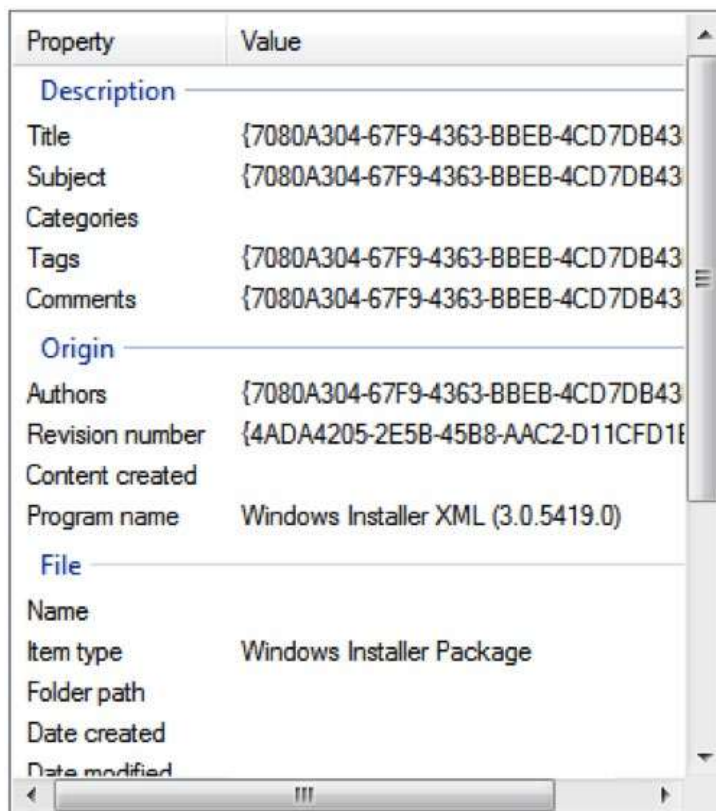
- 复合文档文件格式 v2 文档
- 低字节序
- 操作系统：Windows，6.1 版本
- 内码表：1252 ISO-8859-1
- 标题：{7080A304-67F9-4363-BBEB-4CD7DB43E19D} ( 随机生成的 GUID )
- 主题：{7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- 作者：{7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- 关键词：{7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- 注释：{7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- 模板：Intel；1033
- 最后保存者：{7080A304-67F9-4363-BBEB-4CD7DB43E19D}
- 修订编号：{4ADA4205-2E5B-45B8-AAC2-D11CFD1B7266}
- 页数：100
- 字数：8
- 创建的应用程序名称：Windows Installer XML ( 3.0.5419.0 )
- 安全性：4

应当指出的是，用于其他攻击的 MSI 文件可以具有不同的其它属性。例如，我们发现了以下字段。

- 厂商：Microsoft 或 InstallShield
- 版本：1.0.0.0 或 1.1.2.0 或 2.0.0.0

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

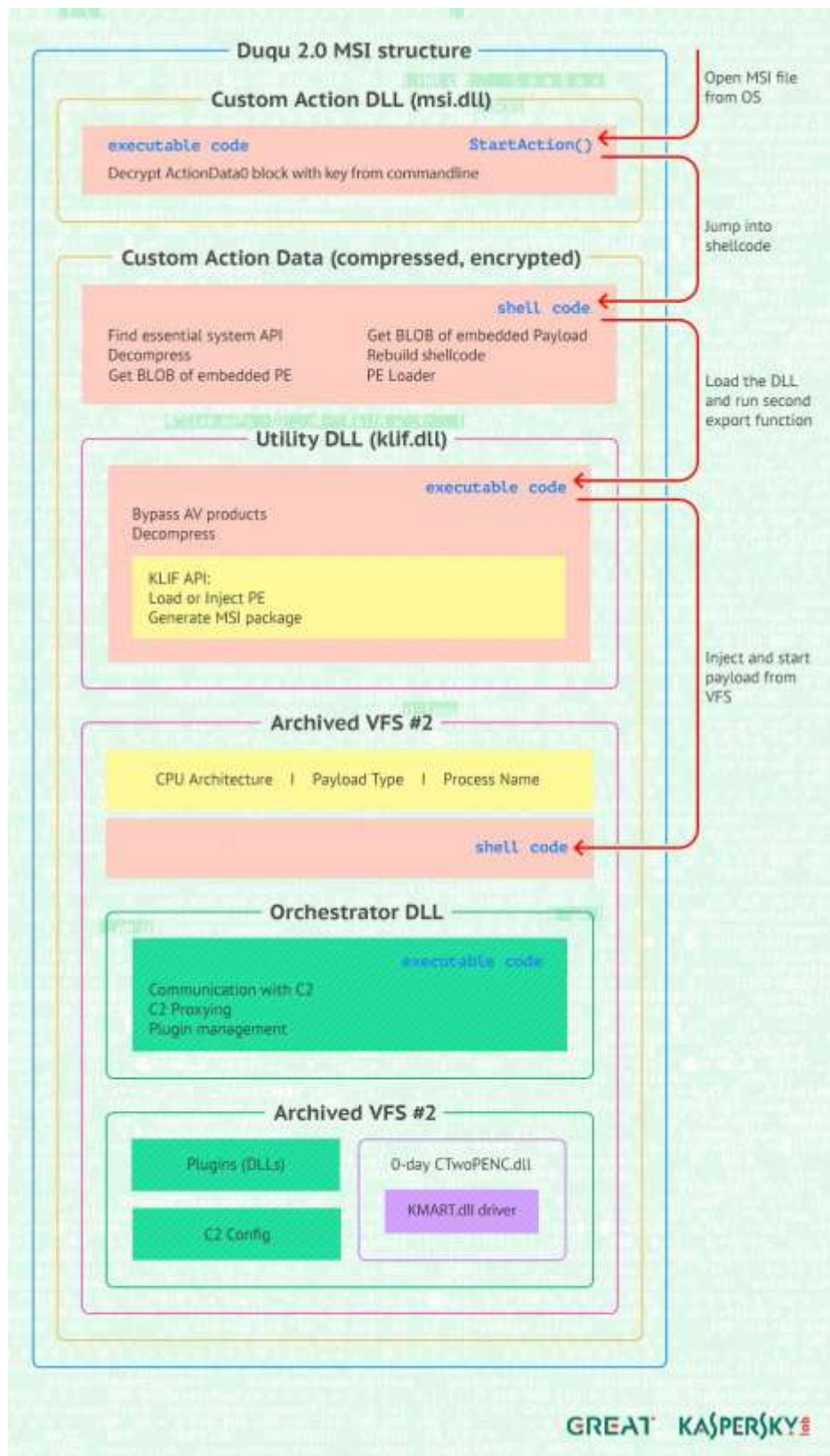
其中一些可以通过 Windows 资源管理器文件属性对话框查看：



MSI 安装包中有两个二进制数据块：

Tables	Name	Data
Binary	ActionDll	[Binary Data]
CustomAction	ActionData0	[Binary Data]
InstallExecuteSequence		
Property		

第一个二进制数据块被称为 ActionDll，它实际上是一个 Windows PE DLL 文件；另一个是使用 Camellia 算法加密并使用 LZJB 压缩的数据载荷（加密和压缩算法视情况而异）。事实上，数据载荷包含多层次的嵌套结构，可执行代码以压缩或加密的二进制数据块形式嵌入上一层可执行代码。Duqu 2.0 的 MSI 安装包及其所有内部载荷结构如下图所示。





下面，我们更详细地介绍这些组件。

## 第一层：ActionDLL ( msi.dll )

Original filename: msi.dll

MD5: e8eaec1f021a564b82b824af1dbe6c4d

Size: 17'920 bytes

Link time: 2004.02.12 02:04:50 (GMT)

Type: 64-bit PE32+ executable DLL for MS Windows

```
mov     [rsp+hInstall], ecx
sub     rsp, 58h
mov     [rsp+58h+pcchValueBuf], 11h
lea     r9, [rsp+58h+pcchValueBuf] ; pcchValueBuf
lea     r8, [rsp+58h+szValueBuf] ; szValueBuf
lea     rdx, szName ; "PROP"
mov     ecx, [rsp+58h+hInstall] ; hInstall
call    MsiGetPropertyW
test    eax, eax
jz      short loc_180003BD2
xor     eax, eax
jmp     short loc_180003C49
```

该 DLL 只有一个称为 StartAction 的导出函数，这是在 msixexec.exe 进程情况下的名称。当这个函数被调用时，它检索被称为 PROP 的 MSI 属性，并将其作为绑定的 ActionData0 安装包的解密密钥。

接下来，代码遍历需要解密和启动的 12 个可能的载荷。载荷是 MSI 的一部分，可能有

以下名称：ActionData0，ActionData1，ActionData2 等。

我们所说的安装包只包含一个名为“ActionData0”的载荷。

## 第二层：ActionData0

```
0000 AppClass      struct ;
0000 dwMagic       dd ? ; 0x72384263
0004 field_4      dd ?
0008 lstrcmplw    dq ?
0010 VirtualQuery dq ?
0018 RtlAnsiStringToUnicodeString dq ?
0020 field_20     dq ?
0028 VirtualProtect dq ?
0030 VirtualAlloc dq ?
0038 GetProcAddress dq ?
0040 RtlFreeUnicodeString dq ?
0048 MapViewOfFile dq ?
0050 FlushInstructionCache dq ?
0058 VirtualFree  dq ?
0060 LdrLoadDll    dq ?
0068 ZwCreateSection dq ?
0070 ZwMapViewOfSection dq ?
0078 ZwUnmapViewOfSection dq ?
0080 FreeLibrary  dq ?
0088 CreateThread dq ?
0090 WaitForSingleObject dq ?
0098 ZwClose      dq ?
00A0 GetSystemDirectoryW dq ?
00A8 ZwOpenSection dq ?
00B0 GetExitCodeThread dq ?
00B8 ZwQuerySystemInformation dq ?
00C0 CreateFileW  dq ?
00C8 GetTickCount dq ?
00D0 GetCurrentProcessId dq ?
00D8 GetCurrentProcess dq ?
00E0 ReadProcessMemory dq ?
00E8 DeviceIoControl dq ?
00F0 GetCurrentThreadId dq ?
00F8 GetModuleHandleW dq ?
0100 LdrUnlockLoaderLock dq ?
0108 LdrLockLoaderLock dq ?
0110 wsprintfW     dq ?
```

这个二进制数据块包含压缩和加密格式的主要代码。它由可执行的、与位置无关的代码块和嵌入的数据对象混合组成。代码似乎基于一个框架，并大量使用包含指针的帮助结构，这些指针指向系统 API（应用程序编程接口）和内部数据块的偏移。这样的结构肯定是开发者的标识。当它们被初始化时，一个包含魔数值的字段 通常是前 4 个字节 就会识别该结构的状态和类型。

开发者的另一个标识是按照模块和导出函数名哈希值导入系统 API。可执行代码的这一层和其他各层中都发现了该哈希算法，可以通过两个 DWORD 常数 0x8A20C27 和 0x67F84FC6 轻松识别。

基本上，ActionData0 中的代码执行内部名称为“klif.dll”的嵌入式可执行文件，即 DLL 文件的导出表中的第二个导出函数。这与导出名称无关，只取决于导出表中的函数顺序。当此导出函数被调用时，下一阶段的帮助结构指针就会传递给它，这样它就可以使用上一层设置的一些值。

然而，在执行 klif.dll 之前，代码会尝试其他路径。首先，它试图找到 “api-ms-win-shell-XXXX.dll” 格式的名称，其中 “x” 可以是任何十进制数。如果当前进程中不存在具有此格式文件名的模块，该名称就是有效的。该代码迭代地搜索这样的名称，包括 api-ms-win-shell-0000.dll，api-ms-win-shell-0001.dll 和 api-ms-win-shell-0002.dll 等。这可能依赖于 Duqu 平台组件，不过尚未确认。

紧接着，如果发现此格式的名称，代码试图通过名称映射一个用基于 PRNG 的算法生成的共享内存区内核对象。该共享内存区的名称具有以下格式 “\BaseNamedObjects\{XXXXXXXX-XXXX-XXXX-XXXXXXXXXXXX}”，其中 “x” 是根据当前系统启动时间生成的任何十六进制数。到目前为止，共享内存区的名称取决于 “机器/启动时间”，这使得它独一无二。但是，如果其他进程使用同样的名称生成算法，它们也可能找到这样的共享内存区。该共享内存区可以在其他不同的代码和模块部分进行访问。以下，我们将该共享内存区称为 OSBoot。一旦共享内存区的名称生成，代码就会试图打开该共享内存区；如果找到这样的共享内存区，代码就使用它的一些值，试图打开一个特定驱动设备并向它提供大量的 IOCTL 代码。驱动设备的名称和 IOCTL 代码位于内核模式驱动程序 KMART.dll 的一个共享内存区中，我们将在下文介绍。

代码开发者偏好使用共享内存区来访问数据。共享内存区的另一个用途似乎是映射包含 klif.dll 的代码/数据的一部分，然后使用硬编码 QWORD 魔数 0xA1B5F8FC0C2E1064 找到包含 klif.dll 的共享内存区。一旦在当前进程的地址空间中找到该共享内存区，代码就会试图执行该共享内存区。这种替代执行路线并不适用于当前的 MSI 文件包，它只存在于代码中，这可能是由创建目前 MSI 软件包的常见代码模板导致的，也可能是我们未发现的另一个 Duqu 平台组件的信标。

### 第三层：klif.dll

Original filename: klif.dll

MD5: 3fde1bbf3330e0bd0952077a390cef72

Size: 196'096 bytes

Link time: 2014.07.06 08:36:50 (GMT)

Type: 64-bit PE32+ executable DLL for MS Windows

显然，这个文件试图模仿卡巴斯基实验室的产品组件 “klif.sys” 的合法名称。虽然代码或文件信息都没有相似性，但是该模块在导出函数名称 KLInit 和 KLDone 中使用了卡巴斯基实验室产品的首字母缩写。

当该 DLL 被加载到一个新的进程时，它会简单地将内部结构初始化（如那些向系统 API 提供所需指针的内部结构）。该模块的实际载荷位于 KLDone 导出函数中，即导出表中的第二项。该导出函数从之前的代码层调用。

首先，它确保通过重要函数（包括 ntdll.dll，kernel32.dll 和 user32.dll）对全局应用程序结构进行初始化。它使用导出函数名称的哈希值将系统 API 函数导入。散列算法与上层所介绍的相同，而且使用相同的魔数常量 0x8A20C27 和 0x67F84FC6。

接下来，代码通过正在运行的进程列表进行迭代，并对每个进程的小写名称计算哈希值，并将该哈希值与 0x3E3021CB 的硬编码值（“avp.exe”字符串的哈希值）进行比较。

## 攻击 AVP.EXE

如果“avp.exe”进程正在运行，该模块就会试图打开 OSBoot 共享内存区并攻击 avp.exe 进程。攻击者首先遍历下列产品硬编码的注册表项和值，从而确定安装了卡巴斯基实验室产品的准确路径。

KES12	AVP15	AVP10	AVP8
KES11	AVP14.0.0	KES9	AVP7
KES10	AVP14	KES8	AVP6
AVP16.0.0	AVP13	AVP80	
AVP16	AVP12	AVP90	
AVP15.0.0	AVP11	AVP9	

该模块查询的注册表值包含到根目录的文件系统路径，相应的产品通过该路径安装。例如：“C:\Program Files\KasperskyLab\Kaspersky Internet Security 15.0.0\”。

一旦发现注册表项和值，该模块就确认 avp.exe 文件位于该目录中。它从 0x5A4D（Windows 可执行文件的魔数“MZ”）开始执行一些基本的文件可用性检查：确保解析环境变量并在读取时打开文件。

在此之后，该模块创建一个新的共享内存区，将 avp.exe 映射到该共享内存区中。为了允许代码执行，该内存区域的属性也相应改变。这允许该模块在内存中改变被映射的 avp.exe 的字节。该模块以极不寻常的方式对被映射的 avp.exe 打两个具有 SSE2 CPU 扩展名的补丁。事实上，它只修复旧的 DOS PE 标头（小于 120 字节）。这些补丁是简单的回调机制，能够调用作为参数传递的任意函数。

紧接着，该模块试图启动一个新的、以打补丁的位置之一开头的线程。到目前为止，新线程貌似以 avp.exe 模块的调用开头，该模块还执行一些其他的调用，即跳转回 klif.dll。显然，该攻击旨在欺骗 avp.exe 进程，使其相信进一步的调用是安全的，相信调用堆栈来自合法的 avp.exe 模块。之后，新线程立即从 avp.exe 跳转回 klif.dll，并尝试与卡巴斯基实验室的产品 minifilter driver（即 klif.sys）通信。

事实上，在与 klif.sys 通信前，它会打开 OSBoot 共享内存区，并获取一个 IOCTL 代码和驱动设备的名称，

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。



然后把这个 IOCTL 代码发送给驱动设备。该共享内存区和相应的驱动程序 (KMART.dll) 给当前模块提供一定的支持。该代码计算当前进程名的哈希值,并验证它是否与以下 3 个哈希值匹配:0x3E3021CB ( avp.exe ), 0xDE6D4DA0 ( msixexec.exe ), 0xB915B2D8 ( RUNDLL32.EXE )。

如果能够识别当前进程名的哈希值,则该模块就会打开\KlifComm minifilter 通信端口并发送一系列驱动通信消息,从而与 klif.sys 通信。通过分析这些消息,我们发现该技术使得 klif 拦截程序无法检测到进程或线程操作。杀毒产品认为这个注册过的项目是受信任的,所以不需要反病毒扫描、进程监控、防火墙等防御措施。此外,该模块请求支持卡巴斯基产品的自防御功能,该功能通常是用于防御侵略性的、利用各种可能的技术来终止安全软件进程的恶意软件。当然,这保证了即使是具有管理权限的用户也无法终止这样的进程。

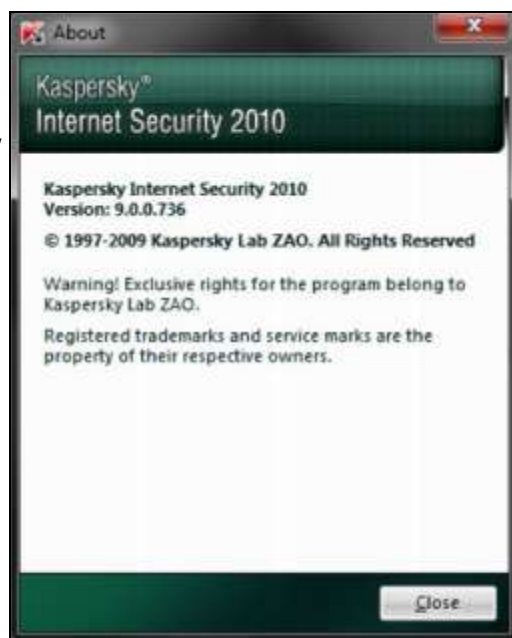
鉴于该攻击针对卡巴斯基实验室产品,所以我们对这些产品进行了其他测试,发现当前的产品都会通过检查自定义数字签名来验证调用进程。截至目前,若没有额外的驱动程序支持,这一技术是无法实施的。

卡巴斯基实验室的所有 2010 年之后的产品都会对开启\KlifComm minifilter 通信端口的数字签名进程进行验证。截至目前,它只能影响卡巴斯基实验室在 2009 年及之前发布的产品,例如 KIS2010。

如果说攻击者从 2009 年或更早期就开始针对卡巴斯基实验室产品进行戏弄攻击,这看起来并不现实。所以我们试着寻找其他合理的解释,并且好像找到了。

通常这种攻击对我们的产品是无效的,因为我们的产品会通过检查自定义数字签名来验证调用进程的合法性。为了绕过上述检测,名为“KMART.dll”的 Duqu 2.0 组件对内存中的“klif.sys”打补丁。攻击者的“KMART.dll”已经在内核模式中运行,所以攻击能够有效进行,这是由 Windows 内核漏洞导致的。

发送代码后,该模块继续执行下一个阶段,如下所述。



## CTwoPENC.dll 0 day 漏洞和 KMART.dll

第三层 klif.dll 发挥着多重功能，以确保恶意代码在内存中存活并绕过反病毒检测。

其中，重要的一步是获取内核级访问权限。在 64 位系统中，如果没有已签名的驱动程序，我们就难以加载并运行内核模式的代码。尽管其他攻击者选择利用第三方签名的驱动程序，例如 Equation 或 Turla，但是 Duqu 2.0 平台依赖于更加狡猾的把戏。

与“klif.dll”捆绑在一起的载荷是“CTwoPENC.dll”。它是一种 Windows 内核模式漏洞 (CVE-2015-2360)，能使自身在系统中获取最高权限以运行代码。我们恢复了带有以下编译时间戳的“CTwoPENC.dll”的多个版本，它们都是 Windows 32 位和 64 位版本：

- 2014.08.25 01:20:04 (GMT)
- 2014.08.25 01:19:03 (GMT)
- 2014.07.06 09:17:03 (GMT)

与其他的 Duqu 2.0 模块不同，这些时间戳看似是合法的。这一现象的原因还不可知——有可能 Duqu 平台的开发者是从别处获取的这个模块，并且忘了更改它的编译时间戳。

“CTwoPENC.DLL”在“win32k.sys”中利用 0 day 漏洞来获取内核权限，与此同时作为普通权限的用户正常运行着。它在多个线程中创建了多个窗口，名称分别是“CPer”、“Zero”、“CTwo”和“Vero”，并控制着回调指针。

```
v0 = GetProcessHeap();
v29 = HeapAlloc(v0, 8u, 0x8000u);
if ( v29 )
{
    v1 = GetProcessHeap();
    lpMem = HeapAlloc(v1, 8u, 0x4000u);
    if ( lpMem )
    {
        WndClass.lpfnWndProc = DefWindowProcA;
        WndClass.lpszClassName = "CPer";
        if ( RegisterClassA(&WndClass) )
        {
            hWndNewParent = CreateWindowExA(0, "CPer", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
            if ( hWndNewParent )
            {
                WndClass.lpszClassName = &ClassName;
                v9 = 0;
                do
                {
                    v25 = v9 + 1;
                    sub_100037FE(&ClassName, "%d%s%d", v9);
                    v10 = RegisterClassA(&WndClass);
                    *((WORD *)lpMem + v9) = v10;
                    if ( !v10 )
                        break;
                    v11 = CreateWindowExA(0, &ClassName, &ClassName, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0);
                    v12 = (char *)v29 + 4 * v9;
                    *((DWORD *)v12) = v11;
                    if ( !v11 || sub_100035AB(v11, &hWnd, 0) )
                        break;
                    v14 = *((DWORD *)v12);
                    if ( Check64(v13) )
                }
            }
```

注册窗口类别并创建新窗口的漏洞代码的一部分

这一漏洞利用代码的主要目的是加载名为“KMART.dll”的驱动程序。如果加载成功，它将在内核模式

中运行的同时,从模块主体中解压“KMART.dll”并对此加以控制。“KMART.dll”拥有 32 位和 64 位两种版本。这一驱动程序针对卡巴斯基实验室的产品“klif.sys”执行内核模式攻击。它会给系统 API 函数 ( PsGetCurrentProcessId 和 PsLookupProcessByProcessId ) 的导入表地址打补丁,也会给卡巴斯基实验室的反病毒驱动程序 klflt.sys 的导出函数 KlGetStringRef 的地址打补丁。在函数被拦截之前,恶意代码将恶意模块伪装成来自进程/文件“avp.exe”(使用反病毒组件的真实位置),从而绕过反病毒软件的监控。

在测试过程中,该 0 day 漏洞(微软在 2015 年 6 月 9 日修复)在所有当前的 Windows 版本中都能够运行。

## 载荷容器和迁移

该模块支持各种在内存中以容器形式存在的、位于 klif.dll 之后(位于 PE 区)的载荷。

载荷包含一个简短的标头,描述了目标 CPU 架构、载荷类型和目标进程的名称或哈希值。当该模块利用一些嵌入代码或文件时,代码通常包含两个变种:32 位和 64 位。每种载荷运行类型都通过单一的字符识别。当前的模块支持以下 5 种载荷运行类型:g、i、k、l 和 q。看来开发者使用的字符按照字母表顺序排列,也就是说可能还有其他载荷运行类型,但是在运行过程中被删除了。当前,位于可执行对象中的容器包含一个“l”字符,所以我们先来讨论它。

## 载荷类型“l”

首先,代码从其他进程中寻找权限。通过搜索当前运行的进程、计算进程名称的哈希值并将其与已知哈希值对比,可以获得进程句柄并将其硬编码。该模块包括以下硬编码的进程哈希值:

0x31283314 (services.exe), 0x1026D1BA (lsass.exe), 0x3BD2498C (svchost.exe)

如果找到任何已识别的进程,该模块就会开启进程令牌并验证以下权限的可用性:

- SeDebugPrivilege
- SeTcbPrivilege
- SeAssignPrimaryTokenPrivilege
- SeIncreaseQuotaPrivilege
- SeImpersonatePrivilege
- SeRestorePrivilege

如果权限可用,该模块就会复制进程令牌,并将其保存在内部结构中以备后用。

该模块会寻找当前系统安装的可用安全套件之一；如果找不到安全套件，就会寻找一些通用进程。它试图寻找以下软件的可执行文件：

- 卡巴斯基实验室的 20 多种产品
- McAfee VSCore、SystemCore、HIP
- Symantec Internet Security
- Bitdefender
- ESET
- H+BEDV
- TrendMicro Network Security Component
- Rising AV
- Avast
- AVG 2012/2013/2014/2015
- F-Secure
- Qihoo 360 Security
- 通用的 Windows 进程：
  - o %programfiles(x86)%\windows Defender\msmpEng.exe
  - o %systemroot%\syswow64\svchost.exe
  - o %systemroot%\system32\lsass.exe
  - o %systemroot%\system32\svchost.exe

如果找到这样的可执行文件（最终会找到的，至少 svchost 或 lsass.exe 是存在的），它会从发现的位置运行一个新的进程，并用自身加密压缩格式的代码替换当前可执行文件中的可执行代码。

如果代码能够从具有特定安全权限的进程中找到安全令牌，它将在利用 CreateProcessAsUser API 创建新进程的过程中利用该令牌。否则，它只会生成一个新的进程实例。

新进程在启动时，处于挂起和调试状态。新进程启动之前，为了简化代码注入和数据控制，子进程的 DEP 会被禁用。至此，从外表看来，它像是被启动的安全软件，实际上却与合法模块完全不同。

## 载荷运行类型 “G”

这一种载荷几乎和上述的 L 类型相同，但是它的加载器更加简单——它能越过权限令牌的拦截。它根据已知进程的可用性自动选择代码注入目标。

## 载荷运行类型 “I”

这一种载荷与 G 类型类似，但是加载器解析载荷标头时有所不同：它只用 4 个字节作为进程哈希，而非用进程名称字符串作为进程哈希。这可能是为了隐藏目标进程的名称，但是并非整个模块都使用该方法。欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

## 载荷运行类型 “K”

这一种载荷是专为当前进程环境设计的。代码简单地将待执行代码复制到独立内存中，并在专门的线程中运行该代码。它不断地进行拦截，直至线程完成执行。

## 载荷运行类型 “Q”

这一种载荷同上述的 K 类型相同，但是在新线程启动时，它不拦截执行。目前，新的代码是异步运行的。

载荷容器开启后，代码会迁移至另一个受安全软件保护的进程中，此时真正的恶意代码就被激活了。在多数情况下，基于简单命名管道的后门监听来自协调器的通信。在少数案例中，在选定机器中，协调器模块与 C&C 服务器通信，充当双向代理，并提供大量的二级插件。

## 插件化模块平台

除了基础的驻留内存远程后门外，攻击者将更为复杂的安装包配置给域控制器以及 LAN 内的受害者。这些 MSI 包包含数十种不同的模块，能够提供各种网络间谍功能。

全功能安装包比基础远程后门要大很多——18MB vs 500KB。他们遵循相同的结构，利用 ActionDll 以及加载器机制，不同之处在于它们包含更多的加载和运行插件。

在分析过程中，我们确定了该插件的 100 多个变种。下面是对这些插件的描述。我们使用了一个虚拟识别器，按照它们的 MD5 校验和的前两个字符对其加以区分。

### 03B7：Duqu 2.0 的主要模块，协调器

该模块为 C&C 通信执行多重协议处理，能够启动带有自签名 HTTPS 证书的中间 C&C 代理服务器、启动插件框架、加载并管理所有其他插件。

它通过 HTTP、HTTPS 和 SMB 网络管道或直接 TCP 连接（使用自定义的加密协议）进行运作。通过 HTTP 进行的交互被隐藏在 JPEG 或 GIF 文件中，这与 2011 年版本的 Duqu 相同。不同攻击中的请求名称、URL 以及用户代理字符串可能会有所不同。

其他已知变种：3026、4F11

### 0682：收集基本系统信息

- 运行进程的列表
- 活跃桌面和终端会话

之后，收集的信息通过调用者提供的命名管道传输。



其他已知变种：COB7

073C：在客户端和服务端执行完整的基于 Windows 套接字的传输，提供一个类别库来封装各种网络函数。

0872：当恶意安装包被 Windows Installer 启动后，MSI CustomAction 扩展库就会被激活。加载包含真实恶意载荷的加密二进制对象，将其解密并在内存中运行。

版本信息中的名称不同：svcmsi\_32.dll、msi3\_32.dll、MSI.dll、msi4\_32.dll。

加密算法也不同：Camellia 256、AES、XXTEA。

解密密钥是从 MSI 参数中提取的，可能的名称是：PROP、HASHVA、CKEY。

通过以下前缀搜索加密的二进制对象：ActionData、CryptHashs、CADData。

32 位和 64 位版本都是已知的。其他已知变种：8D7C、16EF、E6E5、434C、44BD、F708。

09A0：64 位，过滤文件内容，尤其是搜索匹配以下规则的文件：

- \*.inuse, \*.hml
- 包含 “data.hmi” 或 “val.dat” 的文件名
- 来自 “/Int/HMI/” 或者 “/LG/HM/” 文件夹的文件

```

Hml:                align 10h                                ; DATA XREF: sub_18001AC58+750fo
                    unicode 0, <hml>,0
Data_hmi:            unicode 0, <data.hmi>,0                  ; DATA XREF: sub_18001BF70:loc_18001C021fo
0+                  align 10h
Val_dat:             unicode 0, <val.dat>,0                  ; DATA XREF: sub_18001C5BC+1Efo
0+                  unicode 0, <val.dat>,0
IntHmi:              ; DATA XREF: sub_18001BF70+1Ffo
                    ; sub_18001C1F8+1Efo
                    ; sub_18001C1F8:loc_18001C251fo
0+                  unicode 0, </Int/HMI/>,0
                    db 'L',27h,9,0
LgHm:                ; DATA XREF: sub_18001AC58+764fo
0+                  unicode 0, </LG/HM/>,0
qword_18003F988      dq 19DB1DED53E8000h                    ; DATA XREF: sub_180023DA0+136ir
dword_18003F990      dd 1                                    ; DATA XREF: sub_180023DA0+1A3ir
                    align 8

```

#### 09A0 插件感兴趣的文件和目录名

其他已知变种：8858

0AB8：提供 25 个用于控制文件和目录函数。

- 列出目录中的文件
- 上传和下载任意文件
- 读取/写入文件内容

在一些案例中，该模块会特别搜寻名为 “\int”、“\lg”、“\of\md”、“\tl”、“\ak” 的目录以及具有 “.part”、“.manual”、“.inuse” 扩展名的文件。

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

```

Alldirs      db 'allDirs',0          : DATA XREF: sub_10003A3A+32fo
Downloadable db 'Downloadable',0    : DATA XREF: sub_10003C43+17fo
              align 10h
Dlg          db 'dLg',0             : DATA XREF: sub_1000409E+18fo
Explorer_0   db 'Explorer',0        : DATA XREF: sub_10004329+20fo
              align 10h
asc_10017210 db '\',0               : DATA XREF: sub_1000455A+18fo sub_10007882+77fo sub_10007882+88fo
              : sub_10008FE8+18fo
              align 10h
              dd offset Int          : "\\int"
              dd offset Lg           : "\\lg"
              dd offset Tl           : "\\tl"
              dd offset Ak           : "\\ak"
              dd offset OfMd         : "\\of\\md"
              dd offset Int          : "\\int"

```

#### 0AB8 插件感兴趣的文件和目录名

其他已知变种：A69E

0B97：网络传输。执行 API，以便连接调用者提供的套接字和管道。

其他变种：56A2

152B：搜索网络和域

- 枚举域中所有的服务器
- 尝试连接远程注册表并枚举“HKEY\_USERS”中的所有用户，用户名被转换为 SID。
- 枚举所有可见的网络共享

其他已知变种：A987

1C71：内存存储。接收并存储文件映射中的字符串数据。

2125：网络感染模块。试图从正在运行的进程中获取管理凭证，然后利用 Windows 共享连接到目标机器。机器被恶意 DLL 后门感染，而且目标目录有所不同，可能是 UPDROOT\$、ADMIN\$或 C\$。利用自定义的 SMB 包来识别目标操作系统版本，利用新创建的名称为“svcm si\_32@”的服务开启恶意 DLL。该模块通过 Windows 管道与目标后门进行通信。

其他变种：DB65。不同于恶意 DLL，它通过新创建的服务“msisvc\_32@”上传一个新的 MSI 包，依赖 MSIEEXEC.EXE 来启动 MSI。目标 MSI 名称是利用模版“tmp%x.tmp”随机生成的。

```

A service was installed in the system.

Service Name: svcm si_32@
Service File Name: msieexec /i %systemroot%\temp\tmp129deb0.tmp /q PROP=fa5d78ba-f17
Service Type: user mode service
Service Start Type: demand start
Service Account: LocalSystem

```

#### 被感染时，Windows 创建的事件日志（系统）项目

模块通过窃取当前运行进程的令牌来获取管理凭证，这些进程需要域管理员权限予以启动。为此，首先，它直接向域控制器查询当前域中的前 100 个管理员用户名。

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。





- 域控制器的名称
- 域中用户的列表
- 域管理员
- 枚举可信任域
- TCP 表
- UDP 表
- SNMP 发现（操作系统，解析所有回复）
- USB 驱动历史记录，已安装的设备
- 安装的程序
- 时间区域
- 操作系统安装日期
- ODBC.ini、SQL Server 实例信息、Oracle ALL\_HOMES、SyBase、DB2、MS SQL、MySQL 的最后连接
- DHCP/路由
- 网络资料
- 零配置参数
- 已连接的打印机
- WinRAR、WinZip、Office 使用的 MRU 列表，IE 类型的 URL，映射的网络驱动，Visual Studio MRU
- 终端服务客户端默认用户名提示
- 用户帮助历史记录
- PuTTY 主机密钥和会话
- 登录用户
- 网络适配器配置
- VNC 用户密码
- 利用 SMB 数据包扫描网络并识别操作系统

```

Hostname:          unicode 0, <HostName>,0      ; DATA XREF: sub_10008AF1+1610
                  align 4
Logfilename:       unicode 0, <LogFileName>,0    ; DATA XREF: sub_10008AF1+2710
Portnumber:        unicode 0, <PortNumber>,0     ; DATA XREF: sub_10008AF1+3510
                  align 4
Portforwardings:   unicode 0, <PortForwardings>,0 ; DATA XREF: sub_10008AF1+4A10
SSUUS:             unicode 0, <%s - %s:%u [ %s %s ]>,0 ; DATA XREF: sub_10008AF1+CC10
                  align 4
SoftwareSimonTathamPuttySshHostKeys:  unicode 0, <Software\SimonTatham\Putty\SshHostKeys>,0 ; DATA XREF: sub_10008C2B+59C0
                  align 4
SoftwareSimonTathamPuttySessions:      unicode 0, <Software\SimonTatham\Putty\Sessions>,0 ; DATA XREF: sub_10008C2B+5BA0
SoftwareMicrosoftWindowsCurrentVersionExplorerComDlg32File:  unicode 0, <Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg>,0 ; DATA XREF: sub_10008C2B+4510
                  unicode 0, <%2\filemr>,0
SoftwareMicrosoftWindowsCurrentVersionExplorerComDlg32Open:  unicode 0, <Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg>,0 ; DATA XREF: sub_10008C2B+BA10
                  unicode 0, <%2\OpenSavePidlMRU>,0
SoftwareMicrosoftVisualStudio9_0FileMRUList:                 ; DATA XREF: sub_10008C2B+D410
                  unicode 0, <Software\Microsoft\VisualStudio\9.0\FileMRUList>,0
SoftwareMicrosoftVisualStudio9_0ProjectMRUList:              ; DATA XREF: sub_10008C2B+11E0
                  unicode 0, <Software\Microsoft\VisualStudio\9.0\ProjectMRUList>,0

```

该模块获取的一些注册位置

其他已知变种：A7F8

2BF7：实用工具 DLL。提供基本的 API 来创建新的 MSI 包，加载并注入随机 PE 模块。还负责加载恶意 MSI 文件中的 VFS 的第一层。存在 32 位和 64 位两个版本。

已知的名称：“ntdll.dll”、“klif.dll”、“apiset.dll”

其他已知变种：6DA1、32DB、8304、9931、9E60、A2D4、ABA9、B3BB、DC5F、DD32、F7BB

3395：MS SQL 发现模块。模块能够将 ARP 包发送至网络并寻找 MS SQL 服务器端口。其他函数负责连接并读取远程注册内容。

35E9：寻找文件系统

- 枚举网络共享
- 枚举本地磁盘
- 遍历文件系统的层级并枚举文件；识别重解析点。

3F45：管道后门。开启一个新的全局可见的 Windows 管道，接收并执行加密指令。识别加密协议的魔数字符串是“ttttttt”。

- 枚举运行的进程
- 加载并执行随机 PE 文件

存在 32 位和 64 位两个版本。

已知的管道名称：

- \\.\pipe\{AAFFC4F0-E04B-4C7C-B40A-B45DE971E81E}\\.\pipe\{AB6172ED-8105-4996-9D2A-597B5F827501}
- \\.\pipe\{0710880F-3A55-4A2D-AA67-1123384FD859}\\.\pipe\{6C51A4DB-E3DE-4FEB-86A4-32F7F8E73B99}
- \\.\pipe\{7F9BCFC0-B36B-45EC-B377-D88597BE5D78}\\.\pipe\{57D2DE92-CE17-4A57-BFD7-CD3C6E965C6A}

其他已知变种：6364、3F8B、5926、A90A、DDF0、A717、A36F、8816、E85E、E927

## 4160：密码窃取器

- 提取谷歌和火狐浏览器的登陆数据
- LSA 凭证

```

Localappdata: align 4 ; DATA XREF: sub_1000401A+2E10
0+ unicode 0, <%localappdata%>,0
align 4
Local: ; DATA XREF: sub_1000401A:loc_1000408710
0 unicode 0, <local>,0
align 8
SGoogleChromeUserDataDefaultLoginData: ; DATA XREF: sub_1000401A+BD10
0+ unicode 0, <%s\Google\Chrome\User Data\Default\Login Data>,0
E+SelectUsername_valuePassword_valueOrigin_urlFromLogins db 'SELECT username_value,password_value,origin_url FROM logins',0
1+ ; DATA XREF: sub_10004158+5910
; MEDIA_TYPE Unknown
Unknown: ; DATA XREF: sub_10004511:loc_100045A910
0+ dw 3Ch
unicode 0, <Unknown>

```

使用这些数据来定位 Chrome 保存的登录信息

其他已知变种：B656

41E2：密码窃取器，64 位模块，提取以下信息：

- IE IntelliForms 历史记录
- POP3/HTTP/IMAP 密码
- TightVNC、RealVNC、WinVNC3/4 密码
- Outlook 设置
- SAM、LSASS 缓冲
- Windows Live 和 Net Passport 密码

```

; CHAR CredEnumerateW[]
+CredEnumerateW db 'CredEnumerateW',0 ; DATA XREF: sub_BD6588+2E10
align 10h
; CHAR CredFree[]
CredFree db 'CredFree',0 ; DATA XREF: sub_BD6588+3E10
align 20h
Microsoft_wininet: ; DATA XREF: sub_BD6588+C310
+ unicode 0, <Microsoft_WinInet>,0
align 10h
+Abe2869f9b474cd9A358C22904dba7f7 db 'abe2869f-9b47-4cd9-a358-c22904dba7f7',0
; DATA XREF: sub_BD6588+D310
align 20h
WindowsliveName: ; DATA XREF: sub_BD6588+F610
+ unicode 0, <WindowsLive:name>,0
align 10h
+_netPassport: ; DATA XREF: sub_BD6588+A610
+ unicode 0, <.Net Passport>,0
align 10h
+82bd0e679fea47488672d5efe5b779b0 db '82BD0E67-9FEA-4748-8672-D5EFE58779B0',0
; DATA XREF: sub_BD6588+B610
align 20h
A db 'A',0 ; DATA XREF: sub_BD69C0+2610 sub_BD69C0+
align 10h
dword_BE1430 dd 20000010h ; DATA XREF: sub_BD84CC+1211r sub_BD84CC+
db 0

```

该模块收集的信息

其他已知变种：992E、AF68、D49F

482F：收集系统信息

- 枚举磁盘
- 获取运行进程的列表
- 大范围地收集进程信息，包括进程运行的时长。

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

- 内存信息
- SID 信息

其他已知变种：F3F4

559B：调查活跃目录

- 利用 ADSI 连接 Active Directory Global Catalog
- 枚举 AD 中的所有对象
- 以人类可读的形式呈现每一个项目

```

v6 = ADsOpenObject(L"GC:", v5, v3, 1u, &stru_100030C8, &ppObject);
*a3 = v6;
if ( v6 >= 0 )
{
    v7 = ADsBuildEnumerator((IADsContainer *)ppObject, &ppEnumVariant);
    *a3 = v7;
    if ( v7 >= 0 )
    {
        VariantInit(&varg);
        v8 = ADsEnumerateNext(ppEnumVariant, 1u, &varg, &pcElementsFetched);
        *a3 = v8;
        if ( v8 < 0 || pcElementsFetched != 1 )
        {
            *v4 = -16;
        }
        else
        {
            *a3 = (*(int (__stdcall **)(LONG, IID *, int *))pvarg.lVal)(pvarg.lVal);
            VariantClear(&varg);
            if ( *a3 < 0 )
            {
                *v4 = -17;
            }
        }
    }
    else
    {
        *v4 = -15;
    }
}
else
{
    *v4 = -14;
}
if ( ppEnumVariant )
    ADsFreeEnumerator(ppEnumVariant);

```

活跃目录的枚举例程

580C：收集系统和网络信息

- 检索域控制器的名称
- 枚举域中的所有用户和组
- 收集任务计划日志
- 收集磁盘信息、移动设备的使用记录
- 检索防火墙策略
- 枚举所有命名的系统对象
- 枚举所有的系统服务

5B78：收集系统信息和实用工具。两个导出函数的其中之一是“GetReport”。

- 枚举运行进程，提取令牌和 SID，收集时间信息
- 利用泄露的凭证登陆用户帐号
- 模拟用户的运行进程
- 利用硬编码模版创建新的 32 位和 64 位 shell 代码

存在 32 位和 64 位版本。

其他已知变种：E8C7、EE6E

5C66：加密的文件 I/O，实用工具。

- 文件 I/O 运行：打开/搜索/读取/写入
- 管理压缩和加密的临时文件



## 622B：生成特殊模式的 XML 报告

- 计算机名称
- 目录
- 枚举所有的逻辑驱动器
- 列举所有文件
- 操作系统序列号
- 域名
- 网络适配器配置：IP 地址、MAC、MTU、适配器列表

```

S_info_xml:                                     ; DATA XREF: sub_100B50E+6Df0
        unicode 0, <?xml version="1.0" ?>
GatherMetadataError:                           ; DATA XREF: sub_100B50E+loc_100B71Cf0
        unicode 0, <Gather metadata error>,0
ArchiveErrorWriteFailed:                       ; DATA XREF: sub_100B50E+123f0
        unicode 0, <Archive error: write failed>,0
ArchiveErrorEndFileFailed:                     ; DATA XREF: sub_100B50E+loc_100B746f0
        unicode 0, <Archive error: end file failed>,0
        align 4
unk_100B71DC:                                   ; DATA XREF: sub_100B7F1+9f0
        db 0FFh
        db 0FEh
?xmlVersion1_0?:
        dw 3Ch
        unicode 0, <?xml version="1.0" ?>
        dw 3Eh, 0Ah, 0
        db 0
        db 0
SurveyresultXslnsXsiHttpWww_w3_org2001XslschemaInstan: ; DATA XREF: sub_100B7F1+1Af0
        dw 3Ch
        unicode 0, <SurveyResult xmlns:xsi="http://www.w3.org/2001/XMLSchema-
        unicode 0, <instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        dw 3Eh, 0Ah, 0
        align 4
UniqueidCompnameSBootosserial08xUniqueidS: ; DATA XREF: sub_100B7F1+35f0
        unicode 0, <
        dw 3Ch
        unicode 0, <UniqueID compname="%s" bootOsSerial="%08X" uniqueid="%s" >
        unicode 0, </>
        dw 3Eh, 0Ah, 0
Surveyresult:                                   ; DATA XREF: sub_100B50E+EBf0
        dw 3Ch
        unicode 0, </SurveyResult>
        dw 3Eh, 0Ah, 0
        align 4
True_0:                                         ; DATA XREF: sub_100B843+2Af0 sub_100B843+AFf0
        unicode 0, <true>,0
        align 4
; BoolValue False
False:                                         ; DATA XREF: sub_100B843+22f0 sub_100B843+84f0
        unicode 0, <false>,0
        align 8
ParametersDironlySMaxdepthU:                  ; DATA XREF: sub_100B843+33f0
        unicode 0, <
        dw 3Ch
        unicode 0, <Parameters DirOnly="%s" MaxDepth="%u">
        dw 3Eh, 0Ah, 0
        align 10h
TimefilterS:                                   ; DATA XREF: sub_100B843+50f0
        unicode 0, <
        dw 3Ch
        unicode 0, <TimeFilter %s />
        dw 3Eh, 0Ah, 0

```

## 生成系统报告的 XML 标签

6302：实用工具，内部名称为“d3dx9\_27.dll”。执行基于计时器的事件。

其他已知变种：FA84

669D：实用工具，检查给定的文件名和目录是否存在。

其他已知变种：880B



6914 :基于嗅探器的网络攻击。利用合法的 WinPcap 驱动程序“npf.sys”。检测感兴趣的 NBNS ( NetBIOS 协议 ) 互联网请求并发送响应。

- 响应 WPAD 请求 ( NBNS 数据包中的 “FHFAEBE” )
- 向 HTTP GET 请求发送响应

网络过滤器基于 BPF 库，HTTP 和 WPAD 响应的载荷由外部提供。

```
Str2      db 'GET ',0           : DATA XREF: sub_1000565E+9010
          align 4
DetectedGetRequestFromSToS:      : DATA XREF: sub_1000565E+F710
          unicode 0, <Detected GET request from %s to %s>,0
          align 10h
NoMoreAttacksLeftNotResponding_: : DATA XREF: sub_1000565E+11C10
          unicode 0, <No more attacks left, not responding..>,0
          align 10h
SentResponsePacketToSForSAttacksLeftU: : DATA XREF: sub_1000565E+21A10
          unicode 0, <Sent response packet to %s for %s (attacks left = %ui)>,0
          align 10h
; char SubStr[]
SubStr     db 'User-Agent: ',0    : DATA XREF: sub_10005890+110
          align 10h
Http1_12000kContentTypeTextHtmlConnectionCloseCon db 'HTTP/1.1 200 OK',0Dh,0Ah
          : DATA XREF: sub_100058E2+F610
          db 'Content-Type: text/html',0Dh,0Ah
          db 'Connection: Close',0Dh,0Ah
          db 'Content-Length: %d',0Dh,0Ah
          db 'Accept-Ranges: none',0Dh,0Ah
          db 'Cache-Control: no-cache, no-store, must-revalidate',0Dh,0Ah
          db 'Pragma: no-cache',0Dh,0Ah
          db 'Expires: Wed, 21 Jan 1995 11:56:00 GMT',0Dh,0Ah
          db 0Dh,0Ah,0
          align 4
NotWpadRequest: : DATA XREF: sub_10005852+10c_10005CBA10
          unicode 0, <Not WPAD request>,0
          align 10h
DetectedWpadRequestFromSToS: : DATA XREF: sub_10005852+C010
          unicode 0, <Detected WPAD request from %s to %s>,0
          align 10h
SentResponsePacket: : DATA XREF: sub_10005852+15010
          unicode 0, <Sent response packet>,0
```

伪造的 HTTP 响应和相关的状态信息

6FAC : 文件 API

- 获取文件大小和属性信息
- 安全删除文件
- 开启/关闭/读取/写入文件内容

其他已知变种 : A7EE

7BDA : 收集系统信息

- 利用 wscapi.dll API 查看当前的杀毒状态和防火墙防护
- 检测 “sqlservr.exe” 是否正在运行
- 计算机名称
- 工作组信息
- 域控制器的名称
- 网络适配器配置
- 时间和时间区域信息
- CPU 频率

其他已知变种 : EF2E

7C23 : 从文档中提取元数据并收集系统信息

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

- 计算机名称
- 系统卷标序列号
- 完整的文件 API，如 6FAC 中的文件 API

搜索文档和压缩包，执行例程来提取所有有价值的信息：

- 电子邮件信息：eml、msg
- 图像文件：jpg、jpe、jpeg、tif、tiff、bmp、png
- 多媒体文件：wmv、avi、mpeg、mpg、m4a、mp4、mkv、wav、aac、ac3、dv、flac、flv、h264、mov、3gp、3g2、mj2、mp3、mpegts、ogg、asf。这些文件都是利用 libffmpeg 进行再编码。
- PDF 文档的内容
- Microsoft Office：doc、docx、xlsx、pptx。调用相应的例程：“OfficeRipDoc”、“OfficeRipDocx”、“OfficeRipXlsx”、“OfficeRipPptx”。提取 PPT 幻灯片并将其转换成 HTML 摘要的格式。
- 压缩包：gz、gzip、gzX3、zip、rar。

创建扩展名为 “.fg4” 的临时文件。

其他已知变种：EB18、C091

```

_docx:                                ; DATA XREF: 10010508fe
        unicode 0, <.>.docx>,0
_pptx:                                ; DATA XREF: 10010514fe
        unicode 0, <.>.pptx>,0
_xlsx:                                ; DATA XREF: 10010520fe
        unicode 0, <.>.xlsx>,0
_zip:                                  ; DATA XREF: 1001052cfe
        unicode 0, <.>.zip>,0
        align 4
_rar:                                  ; DATA XREF: 10010538fe
        unicode 0, <.>.rar>,0
        align 4
; const WCHAR Gdiplus_dll_0
Gdiplus_dll_0:                        ; DATA XREF: sub_1000AAAC
        unicode 0, <GdiPlus.dll>,0
; const WCHAR ImageJpeg
ImageJpeg:                            ; DATA XREF: sub_1000ABDC
        unicode 0, <image/jpeg>,0
        align 4
asc_10013978:                          ; DATA XREF: sub_1000AD7C
        unicode 0, <%s\\s>,0
GatheringRarS:                        ; DATA XREF: sub_1000AD7C
        unicode 0, <Gathering Rar: %s>,0
Rar:                                  ; DATA XREF: sub_1000AD7C
        unicode 0, <Rar>,0
Rar_error_D:                          ; DATA XREF: sub_1000AD7C
        unicode 0, <RAR_ERROR %d>,0
        align 4
; const WCHAR Ooxml
Ooxml:                                ; DATA XREF: sub_1000B25C
        unicode 0, <OOXML>,0
; const WCHAR String
String:                               ; DATA XREF: sub_1000B31C
        unicode 0, <i>,0
        ; sub_1000B310+80fe sub_1
        ; sub_1000B310+11ffe sub_
; const WCHAR Image
Image:                                ; DATA XREF: sub_1000B8AC
        unicode 0, <Image>,0
; const WCHAR Ffmpeg
Ffmpeg:                               ; DATA XREF: sub_1000B9BC
        unicode 0, <ffmpeg>,0
        align 4
RunningLibffmpegS:                   ; DATA XREF: sub_1000B9BC
        unicode 0, <Running libffmpeg: >

```

部分文件扩展名以及相应的状态信息

8172：基于嗅探器的网络攻击。执行 NBNS 名称解析以骗取：

- WPAD 请求
- 以 “SHR” 开头的名称
- 以 “3142” 开头的名称（只限日志）

```

DetectedShrRequestFromStoS:          ; DATA XREF: SHRRequest+91fo
*   unicode 0, <Detected SHR request from %s to %s>,0
   align 4
SentShrResponsePacket:              ; DATA XREF: SHRRequest+122fo
*   unicode 0, <Sent SHR response packet>,0
   align 10h
GotUnexpectedErrorWhileRunning:      ; DATA XREF: SHRRequest:loc_10006FASfo
*   unicode 0, <Got unexpected error while running>,0
   align 4
DetectedLog3142C:                   ; DATA XREF: Log3142+40fo
*   unicode 0, <Detected Log: 3142%C>,0
   align 4
DetectedLogS:                       ; DATA XREF: sub_10006D77+D5fo
*   unicode 0, <Detected Log: %S>,0
   align 4
; const WCHAR String
String:                             ; DATA XREF: sub_100072CB+Efo sub_1000
*                                     ; 100213FCfo
*   unicode 0, <services.exe>,0
   align 4
; char Str2[]
Str2                                db 'GET ',0          ; DATA XREF: DetectReplyGET+71fo
   align 10h
DetectedGetRequestFromStoS:         ; DATA XREF: DetectReplyGET+E3fo
*   unicode 0, <Detected GET request from %s to %s>,0
   align 4
NoMoreAttacksLeftNotResponding_:    ; DATA XREF: DetectReplyGET+108fo
*   unicode 0, <No more attacks left, not responding..>,0
   align 4
SentResponsePacketToSForUriAttacksLeftU: ; DATA XREF: DetectReplyGET+213fo
*   unicode 0, <Sent response packet to >
*   dw 27h
*   unicode 0, <%s>
*   dw 27h
*   unicode 0, < for URI >
*   dw 27h
*   unicode 0, <%S>
*   dw 27h
*   unicode 0, < (attacks left = %u)>,0

```

#### 与攻击相关的状态信息

其他功能：该模块能利用硬编码模版创建新的 shell 代码。

#### 81B7：驱动程序管理

- 将驱动程序写入磁盘
- 开启/停止驱动程序
- 从磁盘安全移除驱动程序的文件

其他已知变种：C1B9

## 8446 : Oracle 数据库和 ADOdb 客户端

- 利用 “oci.dll” API 访问 Oracle 数据库
- 从数据库中提取所有可用信息
- 连接到 ADOdb 提供者

```
Gj43koDdi: ; DATA XREF: sub_10004026+1B7e
unicode 0, <GJ43KO-%dOI>,0
Table_04d_bin: ; DATA XREF: sub_10005060+4A1e
unicode 0, <table_04d.bin>,0
align 4
Table_bin: ; DATA XREF: sub_1000519E+BE7e
unicode 0, <table.bin>,0
Db: ; DATA XREF: sub_10007AF0+BC7e
unicode 0, <DB>,0
align 10h
byte_100100F0 db 8 dup(0) ; DATA XREF: sub_1000579E+427e sub_10006257+F17e
; sub_10007AF0+237e
AlterSessionSetCursor_bind_capture_destinationOff: ; DATA XREF: sub_100059E7+597e
unicode 0, <alter session set cursor_bind_capture_destination = off>,0
AlterSessionSetCursor_sharingForce: ; DATA XREF: sub_10006257+127e
unicode 0, <alter session set cursor_sharing = force>,0
align 10h
AlterSessionSetNls_date_formatDdMmYyyyHh24MiSs: ; DATA XREF: sub_10006257+947e
unicode 0, <alter session set nls_date_format=>
dw 27h
unicode 0, <dd/mm/yyyy hh24:mi:ss>
dw 27h, 0
align 8
BeginDbms_application_info_set_moduleSSEnd: ; DATA XREF: sub_10006257+12A7e
unicode 0, <BEGIN dbms_application_info.set_module>
dw 27h
unicode 0, <%s>
dw 27h
unicode 0, <,>
dw 27h
unicode 0, <%s>
dw 27h
unicode 0, <|: END>,>,0
align 10h
BeginDbms_application_info_set_client_infoSEnd: ; DATA XREF: sub_10006257+15E7e
unicode 0, <BEGIN dbms_application_info.set_client_info>
dw 27h
unicode 0, <%s>
dw 27h
unicode 0, <|: END>,>,0
AlterSessionSetCurrent_schemaS: ; DATA XREF: sub_10006257+19E7e
```

## SQL 查询和相关数据

## 8912 : 处理加密文件并收集系统信息

- 共享文件映射通信
- 将加密数据写入文件
- 枚举窗口
- 枚举网络共享和本地磁盘
- 检索 USB 设备历史记录
- 收集网络路由表

已知的互斥体和映射名称：

- Global\{DD0FF599-FA1B-4DED-AC70-C0451F4B98F0}  
Global\{B12F87CA-1EBA-4365-B90C-E2A1D8911CA9},
- Global\{B03A79AD-BA3A-4BF1-9A59-A9A1C57A3034} Global\{6D2104E6-7310-4A65-9EDD-F06E91747790},
- Global\{DD0FF599-FA1B-4DED-AC70-C0451F4B98F0} Global\{B12F87CA-1EBA-4365-B90C-E2A1D8911CA9}

其他已知变种：D19F、D2EE

9224：运行控制台应用程序。利用桌面“默认”创建进程，将其添加到控制台并将 I/O 重定向至命名管道。

92DB：修改 cmd.exe shell。

```

; wchar_t Else
Else:
    unicode 0, <ELSE>,0
    align 4
; wchar_t Date
Date:
    unicode 0, <DATE>,0
    align 4
    unicode 0, <: >,0
asc_42050C:
    unicode 0, <{>,0
    align 4
; const WCHAR Comspec
Comspec:
    unicode 0, <COMSPEC>,0
; wchar_t Res
Res:
    unicode 0, <REP>,0
Chdir_0:
    unicode 0, <CHDIR>,0
; wchar_t Cd_0
Cd_0:
    unicode 0, <CD>,0
Cmd_exe:
    unicode 0, <CMD.EXE>,0
    align 4
Vol:
    unicode 0, <VOL>,0
; const WCHAR Path
Path:
    unicode 0, <PATH>,0
    align 4
; wchar_t Time
Time:
    unicode 0, <TIME>,0
    align 4
Set:
    unicode 0, <SET>,0

```

#### shell 处理的一些 CMD 命令

9F0D ( 64 位 ), D1A3 ( 32 位 ): 具有合法签名的驱动程序 NPF.SYS ( WinPcap ), 随 VFS 插件一起传播。它执行基于嗅探器的网络攻击。

A4B0：网络调查

- 使用 DHCP Server Management API ( DHCPAPI.DLL ) 来枚举 DHCP 服务器的所有客户端
- 查询所有已知的 DHCP 子网络
- 搜索开启端口 UDP 1434 或 137 的机器
- 枚举所有网络服务器
- 枚举网络共享
- 尝试连接远程注册表来枚举 HKEY\_USERS 中的所有用户，将这些用户名转换成 SID。

B6C1：WNet API

为 WnetAddConnection2 和 WNetOpenEnum 函数提供打包工具。

其他已知变种：BC4A

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。



## C25B：基于嗅探器的网络攻击

运行伪造的 SMB 服务器，欺骗其他机器使用 NTLM 进行身份验证。

- 执行基础的 SMB v1 命令

```

dword_10013340 dd 72h ; DATA XREF: sub_1000
off_10013344 dd offset smb_cmd_negotiate ; DATA XREF: sub_
dd 73h
dd offset SMB_COM_SESSION_SETUP_ANDX
dd 2Bh
dd offset SMB_COM_ECHO
dd 75h
dd offset SMB_COM_TREE_CONNECT_ANDX
dd 0A2h
dd offset SMB_COM_NT_CREATE_ANDX
dd 0A0h
dd offset SMB_COM_NT_TRANSACTION
dd 32h
dd offset SMB_COM_TRANSACTION2
dd 2Eh
dd offset SMB_COM_READ_ANDX
dd 0Bh
dd offset SMB_COM_WRITE
dd 2Fh
dd offset SMB_COM_WRITE_ANDX
dd 4
dd offset SMB_COM_CLOSE
dd 71h
dd offset SMB_COM_TREE_DISCONNECT
dd 74h
dd offset SMB_COM_LOGOFF_ANDX
dd 0

```

该模块处理的 SMB 命令

- 假装进行 IPC\$ 和 A：共享
- 接受用户身份认证请求
- 处理 HTTP “GET/” 请求

```

; char Device[]
Device db '\Device\',0 ; DATA XREF: SelectAdapter+153io
align 10h
; char NtLm0_12[]
NtLm0_12 db 'NT LM 0.12',0 ; DATA XREF: smb_cmd_negotiate+9Eio
align 4
+challenge db 6Ch, 5Bh, 4, 86h, 0Dh, 0C2h, 0DBh, 0Eh, 0E4h, 65h, 51h, 0E5h, 0CDh, 0FEh
; DATA XREF: smb_cmd_negotiate+18Fio
SMB1 db 4 dup(0)
Windows: db 0 ; DATA XREF: SMB_COM_SESSION_SETUP_ANDX+7B1o
+ unicode 0, <Windows>
Windows_0: db 20h, 5, 0, 2Eh, 1, 3 dup(0)
+ unicode 0, <Windows>
LanManager: db 20h, 2, 4 dup(0)
+ unicode 0, <LAN Manager>
; wchar_t Ipc
Ipc: unicode 0, <IPC$>,0 ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+D5io
align 10h
Ipc_0 db 'IPC:',0 ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+F0io
align 4
A db 'A:',0 ; DATA XREF: SMB_COM_TREE_CONNECT_ANDX+11A1o
Fat: unicode 0, <FAT>,0
db 0

```

NTLM 挑战和 SMB 服务器数据

#### ED92：文件系统调查

- 枚举所有本地磁盘并连接到网络共享
- 列举文件

#### EF97：文件系统实用工具

- 枚举文件
- 创建并移除目录
- 复制/移动/删除文件和目录
- 从文件中提取版本信息
- 计算文件哈希值

其他已知变种：F71E

## 持续性机制

Duqu2.0 恶意软件平台旨在存活于被感染系统的内存中，不需要具有持续性。为了实现这一点，攻击者感染长时间运行的服务器，这样一来，即使用户重启计算机来清除恶意软件也无济于事。它只存活于内存中，利用漏洞执行内核级的代码，这说明该攻击组织手段很高明。根本上说，攻击者非常自信能够存活于整个由被感染机器组成的网络中，不需要任何持续性机制。

Duqu 2.0 不具备持续性的原因可能是，攻击者想尽可能地规避检测。大多数现代反 APT 技术都能够发现磁盘上的异常，如罕见的驱动程序、未签名程序或恶意程序。另外，不能通过重启清除该恶意软件的系统已被截图，便于后续的全面研究。对于 Duqu2.0 攻击来说，对被感染的系统进行取证分析是非常困难的，研究人员需要捕获被感染机器的内存截图，然后确定内存中的感染。

然而，该机制也有弱点；万一发生大规模停电，所有的计算机将被重启，到时候该恶意软件将会被根除。为了避开这个问题，攻击者设计了另一种解决方案：他们在直接连网的少数计算机中部署了驱动程序。这些驱动程序可以从外部向内部网络传输流量，从而允许攻击者访问远程桌面会话或者通过使用之前获得的用户凭证连接到该域中的服务器。通过使用这些用户凭证，他们就可以重新部署整个平台。




## 命令和控制机制

Duqu 2.0 使用 2011 年变种中最复杂且高度灵活的 C&C 机制，该机制拥有受其他顶级恶意软件（如 Regin）激发的新特征，包括使用网络管道和邮筒、网络流量的原始过滤、在图像文件中隐藏 C&C 流量。



在 Windows LAN 中，新感染的客户端的 MSI 安装包可能不包含硬编码的 C&C 机制。如果没有 C&C 机制，这些客户端就会处于“休眠”状态，攻击者可以使用包含字符串“ttttttttttttt”的特殊 TCP/IP 数据包通过 SMB 网络管道将它们激活。如果 MSI 文件的配置部分包含 C&C 机制，这要么是一个充当跳转点的本地 IP 地址，要么是一个外部 IP 地址。在总体感染战略中，攻击者确定长时间运行的服务器并将它们设置为中间 C&C 点。因此，被感染的计算机可以在 LAN 中的多个服务器之间跳转，直至连接到互联网。

为了连接到 C&C 服务器，2011 年和 2014/2015 版的 Duqu 可以将流量作为加密数据隐藏在一个无害的图像文件中。2011 版使用 JPEG 文件来实现这一点；新版本可以使用 GIF 文件或 JPEG 文件。图像文件如下所示：

Duqu 2011 – JPEG	Duqu 2015 – GIF	Duqu 2015 - JPEG
 54x54 pixels	 11x11 pixels	 33x33 pixels

2014/2015 版变种的另一个修改是增加了用于 HTTP 通信的多用户代理字符串。2011 版使用以下用户代理字符串：

- Mozilla/5.0 (Windows; U; Windows NT 6.0; en-US; rv:1.9.2.9) Gecko/20100824 Firefox/3.6.9 (.NET CLR 3.5.30729)

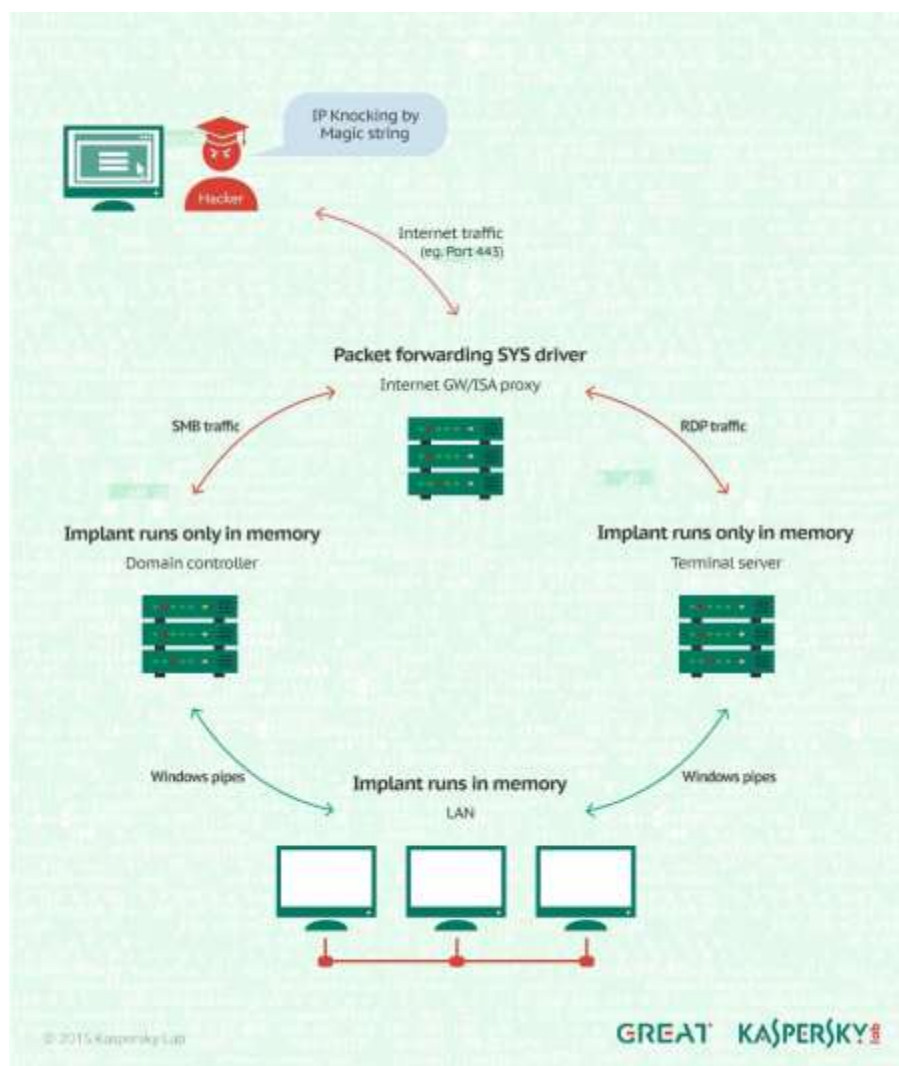
新变种从有 53 种不同字符串的列表中随机选择一个用户代理字符串。

另一个不寻常的C&C机制依赖于驱动程序文件。该文件用来传输C&C通信和攻击者的RDP/SMB活动。攻击者通过直接的互联网连接将该转换驱动程序部署于服务器中。通过试探机制，攻击者可以为它们的IP激活该转换机制，并将流量直接传输至LAN。在LAN之外，流量可以通过端口443隐藏；在LAN内，流量可以是直接的SMB/RDP，或通过伪造的TCP/IP包进一步发送至IP 8.8.8.8。

在研究期间，我们发现几个这样的驱动程序，如下所述：

## “portserv.sys” 驱动程序分析

MD5：2751e4b50a08eb11a84d03f8eb580a4e



大小：14336

编译时间：2006 年 2 月 11 日（伪造的时间戳）

内部名称：termport.sys

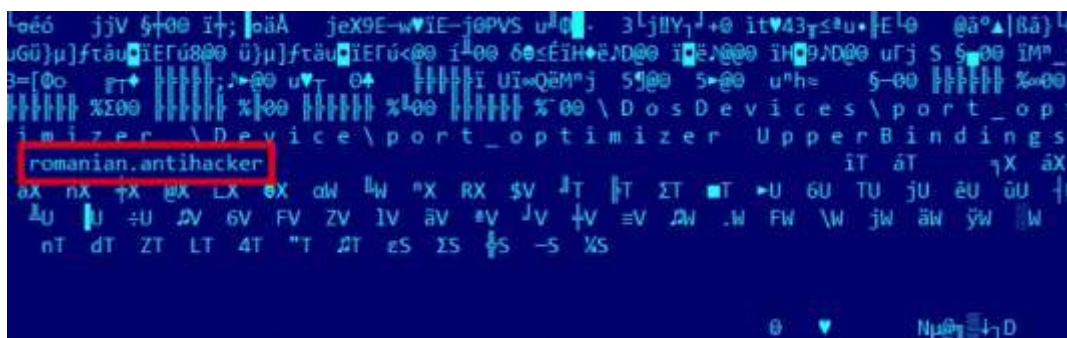
类型：Win32 device driver（也存在已知的 64 位版本）

欲了解更多信息，请联系 [intelreports@kaspersky.com](mailto:intelreports@kaspersky.com)。

这是一个恶意 NDIS 过滤驱动程序，旨在操控 TCP/IP 数据包以允许攻击者访问受害者基础设施中的服务器。

启动时，该过滤驱动程序挂钩到 NDIS 堆栈并处理 TCP/IP 数据包。

为了利用该驱动程序，攻击者首先向被感染服务器的硬编码 IP 发送一个包含字符串 “romanian.antihacker” 的特殊的 TCP/IP 数据包。一般情况下，这样的服务器是能够直接连网的计算机，例如网络服务器或代理服务器。该驱动程序识别字符串 “romanian. antihacker ” 并保存攻击者的 IP 以备后用。



驱动程序中用于试探的魔数字符串

当有来自攻击者 IP（之前保存的）的数据包时，以下逻辑也适用：

- 发送至服务器 1 的 IP 端口 443 的数据包被重定向到端口 445（Samba/Windows 文件系统）
- 来自服务器 1 的 IP 端口 445 的数据包被重定向到攻击者的 IP 端口 443
- 发送至服务器 2 的 IP 端口 443 的数据包被重定向到端口 3389（远程桌面）
- 来自服务器 2 的 IP 端口 3389 的数据包被重定向到攻击者的 IP 端口 443

这使得攻击者能够有效地将 SMB（远程文件系统访问）和远程桌面发送给这两个服务器，同时使其看起来像是 SSL 流量（端口 443）。

这些驱动程序允许 Duqu 攻击者轻易地远程访问 LAN 内的服务器，包括通过端口 443（通常是 SSL）传输 RDP 会话。这也给他们提供了一个持续性机制，即使所有被感染的计算机都重启，恶意软件还是会存活。攻击者可以使用现有的用户凭证登录任何一个这种服务器并将后门重新初始化。

## Duqu 和 Duqu 2.0 之间的相似性

2014/2015 版 Duqu 2.0 是 CrySys 实验室发现的 2011 版 Duqu 的升级版本<sup>7</sup>。它包含了许多现代恶意软件（如 Regain）的新特征，能够执行横向运动，其能力超越了其他 APT 组织。

并驾齐驱：

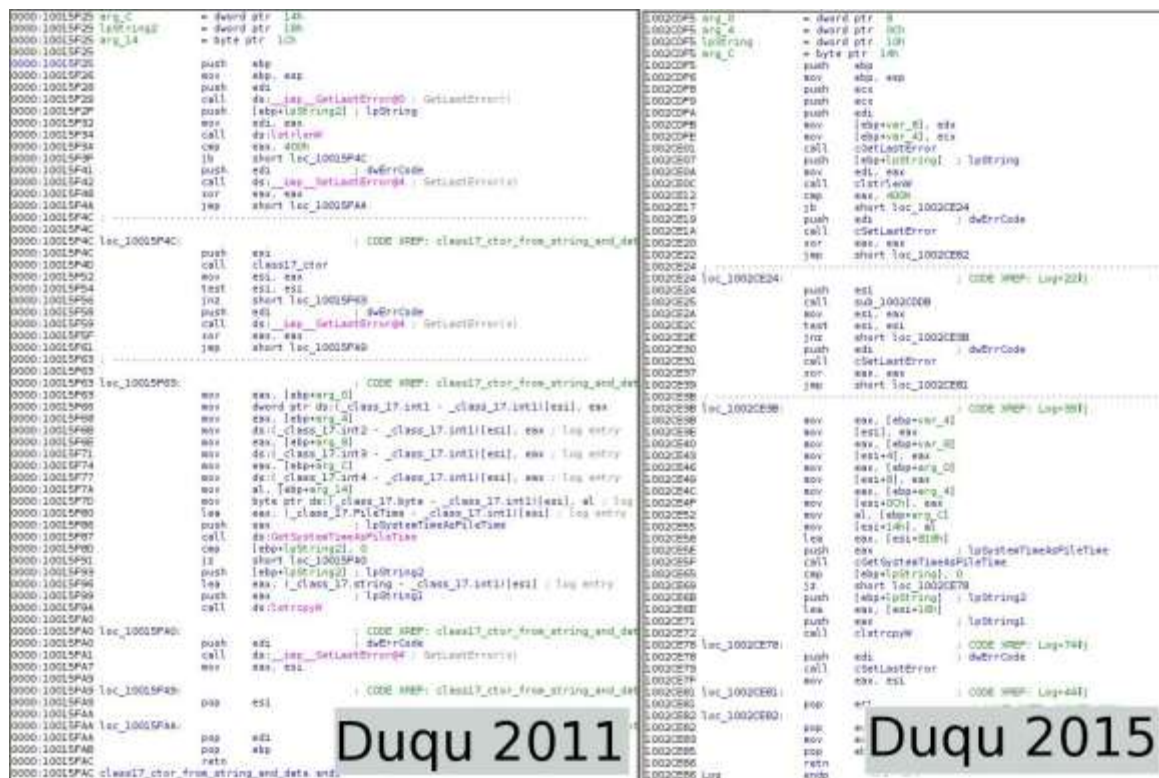
	2011 Duqu	2014/2015 Duqu 2.0
受害者数量	<50 ( 预计 )	<100 ( 预计 )
是否具备持续性机制	是	否
加载器	sys 驱动程序	msi 文件
是否使用 0 day 漏洞	是	是
主要存储	pnf ( 自定义 ) 文件	msi 文件
C&C 机制	http/https，网络管道	http/https，网络管道
已知插件	6	>100

两者的代码具有相似性，因此我们认为 Duqu 2.0 是在 Duqu 的原始源代码的基础上创建的。对此感兴趣的读者可以阅读下文的相似性说明。

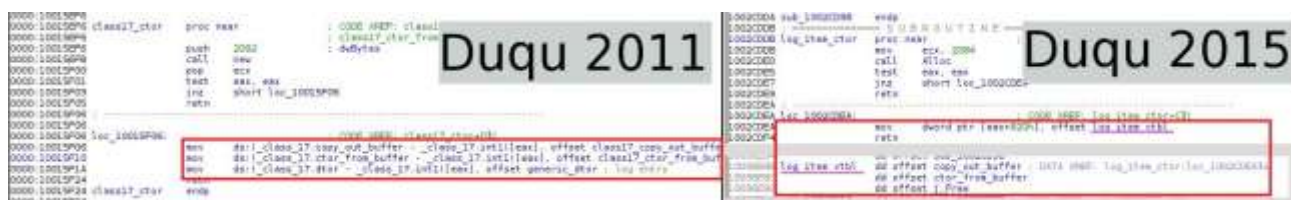
---

<sup>7</sup> <https://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf>

原始 Duqu 最独特的“标识”之一是如何设置提供记录功能的函数。与许多其他的 APT 不同，Duqu 以一种特殊的方式记录其活动的每一个重要步骤：日志中没有写入可读的字符串，而是用一系列独特的数字来识别每一种状态、错误或消息。我们比较了生成 Duqu 和 Duqu 2.0 的每个日志项的函数，发现它们几乎是相同的。



第一代 Duqu 也是用一种非常罕见和独特的方式编写的。它用 Visual Studio 编译，某些部分用 C++ 编写，大部分并非由 C++ 编译器原始编译。分析了所有变种后，我们认为这些类是用 oo-c（C 语言的变种）编写的，然后以某种方式转化成可编译的 C/C++ 源代码。所有这些类都有一个非常特殊的功能：每个实例的虚拟函数表在生成器中都是“手动”填充的。有趣的是，Duqu 2.0 并非如此。开发者将编译器从 Visual Studio 2008（2011 年使用）升级为 Visual Studio 2013，并且使用看起来更像由原始 C++ 编译的类。

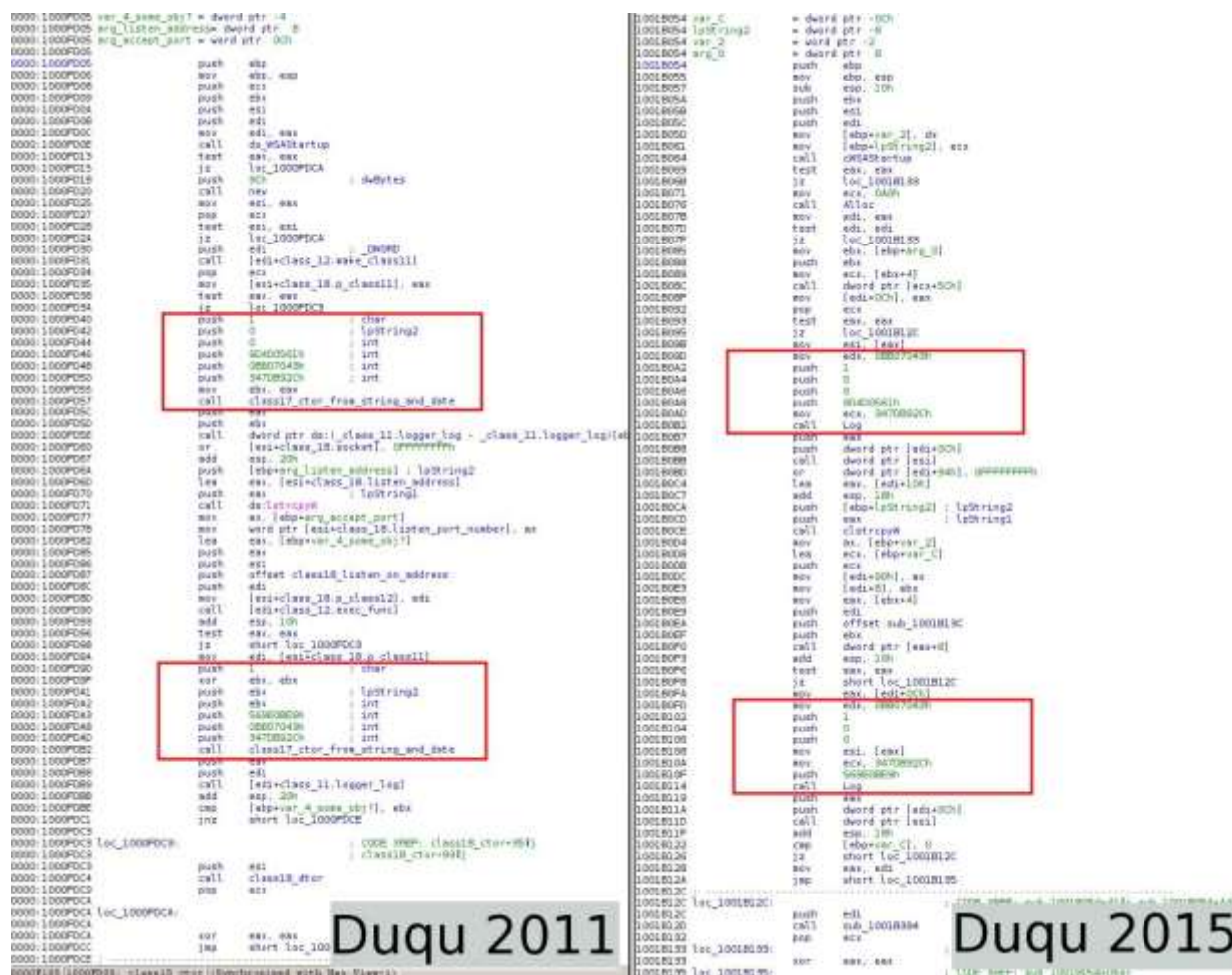


左侧：Duqu 中用 oo-c 编写的“手动”或“编译器协助”类；

右侧：Duqu 2.0 中该类的虚拟表更像 C++ 原始编写的，然而指针偏移不是零。



如果寻找那些真正使用记录工具的函数，我们可以发现更具体的相似性证据。开发者仍然使用相同的唯一编号来识别内部状态、错误和函数结果。可以通过网络功能进行比较：



Duqu 和 Duqu 2.0 执行相同的网络函数。注意：同样的唯一编码（红色方框）PUSHed 是记录函数的参数。

Duqu 2011

Duqu 2015

**方框中的唯一编码用来识别网络例程的当前状态。**

用户代理字符串列表而非单一的硬编码字符串。

Duqu 2011

Duqu 2015

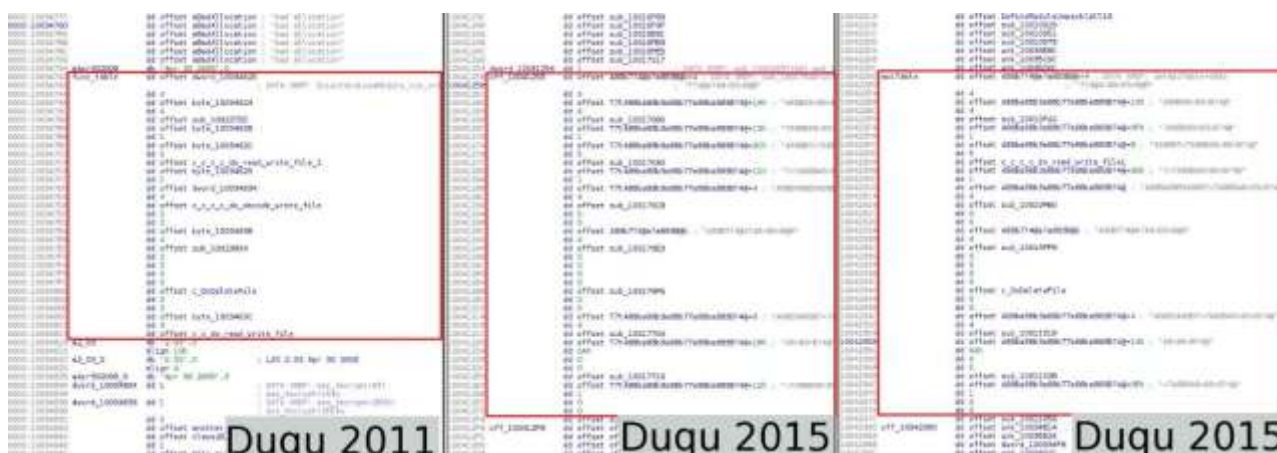
开发者还修改了识别加密网络流量的双字节魔数值：用更中立且更难追踪的“ww”替换了“SH”。



验证网络流量中“魔数值”的代码。

如果 x86/64 架构中的数据采用低字节序，则字符会被交换位置。

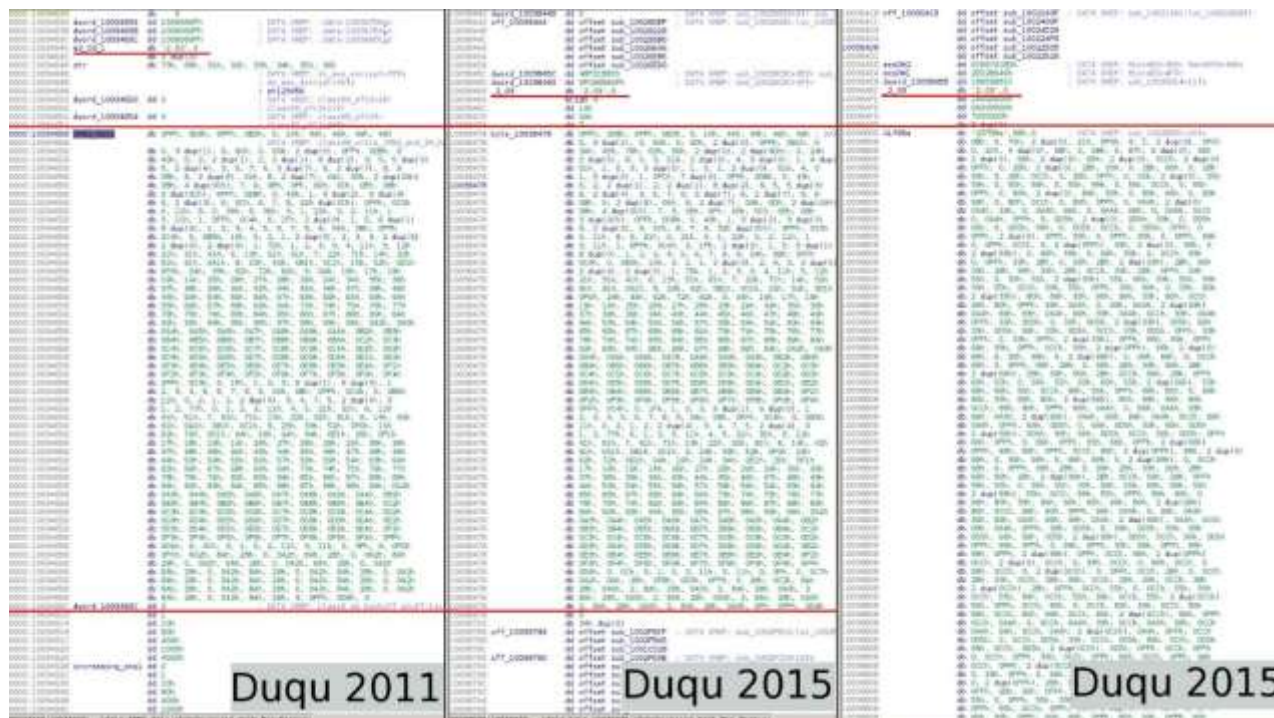
Duqu 和 Duqu 2.0 都使用特殊的结构来确定插件的接口。协调器也具有一个用自有代码编译的“核心”插件。较新的版本具有一个更大的表（因此拥有更多的函数），也具有不同的符号来描述插件的功能。特殊字符串（如“A888A8>@”）描述每个函数的签名。较早的 Duqu 包含类似的二进制形式的字符串（不可读）。



Duqu 和 Duqu 2.0 的两个不同版本的“核心”插件的数据结构，请注意相同的常量和类似函数。



该 Duqu C&C 代码使用小型图像文件，通过未加密的通道（即 HTTP）来隐藏其通信。原始 Duqu 使用 JPEG 文件，已知的 Duqu 2.0 版本使用类似的 JPEG 文件和新的更大的 GIF 文件。此外，数据段的布局并没有太大变化 图像数据以简短的 AES 加密密钥（Duqu 中的字符串“sh123456”，Duqu2.0 中两个二进制 DWORD）开头，接下来是 LZO 版本字符串“2.03”。



用于隐藏 C&C 通信的图像数据：Duqu（JPEG），Duqu Bet（类似的 JPEG），Duqu Bet 的另一个版本（GIF）。注意 LZO 版本的字符串“2.03”和加密密钥。

Duqu 2011 和 Duqu 2.0 样本之间的大量相似性表明，新代码建立在原始恶意软件平台的基础之上。如果没有 2011 版 Duqu 的源代码，新版本就无法创建。因此，我们认为两者的开发者是同一伙人或者他们密切合作。

## Duqu 2.0 的受害者

Duqu 2.0 的受害者遍布世界各地，包括西方国家、中东和亚洲地区。攻击作业者出于提高网络能力的目的，不仅攻击了直接针对的最终目标对象，还攻击了对实现其目的有价值的“功利目标对象”。

最值得注意的是，在 2014 年至 2015 年的时间段内，恶意代码的一部分新感染对象涉及伊核六方会谈（P5+1，译者注：“P5+1”组织成员国包括美国、俄罗斯、中国、英国、法国和德国），以及伊核六方会谈的一些场所。Duqu 背后的威胁源似乎倾向于对涉及此类高层次会谈的场所发动攻击。除了伊核六方会谈，Duqu 2.0 组织还针对奥斯维辛-比克瑙集中营解放 70 周年的纪念活动发动了类似攻击<sup>8</sup>。

我们将新攻击的其他类型的目标称为“功利目标对象”，攻击者为了提高自己的网络能力而攻击这些公

<sup>8</sup> <http://70.auschwitz.org/index.php?lang=en>

司。例如，2011 年，攻击者攻击了匈牙利的一家证书颁发机构；显然，这使得攻击者能够生成数字证书，进一步为恶意软件样本签名。Duqu 2.0 也发动了相同模式的攻击，感染了工业控制系统和工业计算机系统行业的若干公司。

## 归属

一如从前，追踪针对互联网的网络攻击是一项艰巨的任务。在 Duqu 攻击案例中，攻击者使用了多个代理服务器和跳跃点来掩饰他们的连接，这使得追踪变得更为复杂。

此外，攻击者在代码中添加了几个假标识，旨在误导研究人员。例如，其中一个驱动程序包含字符串“ugly.gorilla”，这显然指的是汪东<sup>9</sup>（一名中国黑客，据说与 APT1/Comment Crew 有关）。之前，我们在 Poison Ivy 样本的 MSI VFSes 中发现了 Camellia 密码，这也是攻击者植入的一个假标识，旨在让研究人员认为这是与 APT1 有关的恶意软件。“portserv.sys”驱动程序中的“romanian.antihacker”字符串可能旨在模仿常常出现在服务器日志中的“w00tw00t.at.blackhats.romanian. anti-sec”请求，也有可能只是说明攻击来源于罗马尼亚。使用罕见的压缩算法同样具有欺骗性。例如，一些样本中使用的 LZJB 算法在恶意软件样本中很少见；它曾被我们在 2013 年初报道的 MiniDuke 使用过。

然而，这种假标识都很容易被发现，尤其是当攻击者把心思放在避免犯其他错误的时候。

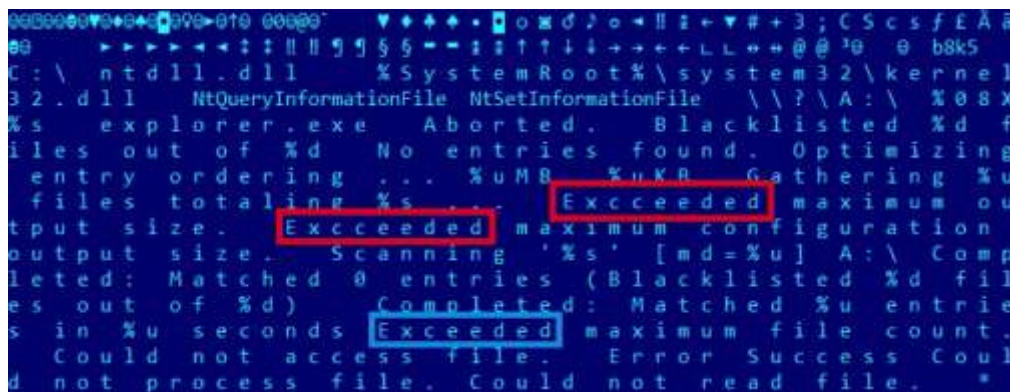
2011 年，我们从一些代理服务器收集到的日志表明，攻击者在星期五不怎么工作，星期六不工作，他们的正常工作日从星期日开始。他们甚至还在 1 月 1 日编制了二进制文件，这说明 1 月 1 日对他们来说很可能是正常工作日。该二进制文件的编译时间戳似乎显示了 GMT+2 或 GMT+3 时区。最后，他们通常在星期三发动攻击，这就是为什么我们一开始将他们称为“周三帮”。2014 年针对卡巴斯基实验室的攻击也发生在星期三；比起 2011 年的攻击行动，该组织在 OPSEC 方面取得了巨大的进步，包括伪造 PE 文件中的所有时间戳、删除所有插件的调试路径和内部名称。

2014 版 Duqu 2.0 的二进制文件包含若干用几乎完美的英语编译的字符串，但其中一个字符串的编译却犯了小儿科错误，这表明该组织中有母语不是英语的人员。我们只发现了 Duqu 2.0 的一个语言错误，即在文件收集模块用了单词“Exccceeded”，而正确的单词应该是“Exceeded”。

---

<sup>9</sup> <http://www.fbi.gov/wanted/cyber/wang-dong/view>





Duqu 2.0 中单词 “Exceeded” 的拼写错误

最有趣的是，其中一个受害者似乎同时被 Equation 组织和 Duqu 组织所攻击；这表明这两个攻击组织是不同的，并且相互竞争地从该受害者窃取信息。

## 结论

2011 年 Duqu 发动攻击期间，我们认为其主要目的是刺探伊朗的核计划。一些受害者似乎是“功利目标对象”，如 Duqu 攻击了匈牙利的一家凭证管理中心，该攻击最终导致了 Duqu 被发现。Duqu 组织攻击这些“功利目标对象”是为了获得技术能力（如利用可信任的证书为其恶意软件签名）或将其作为进一步攻击的平台。

虽然主要的协调器和 C&C 核心基本保持不变，但是 2014/2015 版 Duqu 2.0 大大超越了较早的“Tilded”平台。早在 2011 年，我们就指出使用 Object Oriented C 是一种很不寻常的编程技术<sup>10</sup>。虽然 2014 版添加了 C++中的一些新对象，但是仍保持着相同的核心模块。2014 年版本使用了升级的编译器，这导致了不同的代码优化。尽管如此，核心模块的功能依然如故，并且我们认为，如果没有原始的 Duqu 源代码，2014/2015 Duqu 2.0 根本无法创建。鉴于这些源代码从未被公开，而且主要目标似乎也没变，我们认为 Duqu 和 Duqu 2.0 的幕后黑手是同一伙人。

将卡巴斯基实验室作为目标说明攻击者取得了大进步的“进步”，同时也标志着网络军备竞赛迅速升级。早在 2011 年和 2013 年 RSA<sup>11</sup>和 Bit9<sup>12</sup>都遭到了讲汉语的 APT 组织的攻击，然而，此类事件被认为是罕见的。一般情况下，攻击者将安全公司定为目标需要冒很大的风险，因为他们迟早会被抓住和曝光。虽然攻击者似乎想要获取有关卡巴斯基的未来技术、安全操作系统、反 APT 解决方案、KSN 和 APT 研究的信息，但是卡巴斯基实验室究竟为什么会被定为目标尚不清楚。

<sup>10</sup> <https://securelist.com/blog/research/32354/the-mystery-of-duqu-framework-solved-7/>

<sup>11</sup> <https://blogs.rsa.com/anatomy-of-an-attack/>

<sup>12</sup> <https://blog.bit9.com/2013/02/08/bit9-and-our-customers-security/>

从威胁源的角度来看，将世界级的安全公司定为目标是一项非常艰难的决定。一方面，这意味着攻击一定会被曝光，几乎不可能不被发现。因此，将安全公司定为目标表明：要么攻击者非常自信不会被抓到，要么他们根本不关心是否会被发现和曝光。决定将卡巴斯基实验室定为目标时，Duqu 攻击者很可能下了一个巨大的赌注，希望不会被发现并能够继续潜伏。

对于一家安全公司来说，最困难的事情之一就是承认自己沦为了恶意软件攻击的受害者。卡巴斯基实验室坚信信息透明的力量，这就是为什么我们在此发布了攻击信息。对于我们来说，用户的安全仍然是重中之重，我们将继续努力以维护您的信任和信心。

## 参考文献

1. Duqu: A Stuxnet-like malware found in the wild <https://www.crysys.hu/publications/files/bencsathPBF11duqu.pdf>
2. Duqu: The Precursor to the next Stuxnet [http://www.symantec.com/content/en/us/enterprise/media/security\\_response/whitepapers/w32\\_duqu\\_the\\_precursor\\_to\\_the\\_next\\_stuxnet.pdf](http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf)
3. The Mystery of Duqu: Part One <https://securelist.com/blog/incidents/31177/the-mystery-of-duqu-part-one-5/>
4. The Mystery of Duqu: Part Two <https://securelist.com/blog/incidents/31445/the-mystery-of-duqu-part-two-23/>
5. The Mystery of Duqu: Part Three <https://securelist.com/blog/incidents/31486/the-mystery-of-duqu-part-three-9/>
6. The Mystery of Duqu: Part Five <https://securelist.com/blog/incidents/31208/the-mystery-of-duqu-part-five-6/>
7. The Mystery of Duqu: Part Six (The Command and Control Servers) <https://securelist.com/blog/incidents/31863/the-mystery-of-duqu-part-six-the-command-and-control-servers-36/>
8. The Mystery of Duqu: Part Ten <https://securelist.com/blog/incidents/32668/the-mystery-of-duqu-part-ten-18/>
9. The Mystery of Duqu Framework Solved <https://securelist.com/blog/research/32354/the-mystery-of-duqu-framework-solved-7/>
10. The Duqu Saga Continues <https://securelist.com/blog/incidents/31442/the-duqu-saga-continues-enter-mr-b-jason-and-tvs-dexter-22/>



[Securelist](#), the ressource for Kaspersky Lab experts' technical research, analysis and thoughts



[Kaspersky Lab B2C Blog](#)



[Kaspersky Lab security news service](#)



[Eugene Kaspersky Blog](#)



[Kaspersky Lab B2B Blog](#)



[Kaspersky Lab Academy](#)

Kaspersky Lab, Moscow, Russia  
[www.kaspersky.com](http://www.kaspersky.com)

All about Internet security:  
[www.securelist.com](http://www.securelist.com)

Find a partner near you:  
[www.kaspersky.com/buyoffline](http://www.kaspersky.com/buyoffline)

Kaspersky Lab HQ

39A/3 Leningradskoe Shosse  
Moscow, 125212  
Russian Federation

[More contact details](#)

Tel: +7-495-797-8700  
Fax: +7-495-7978709

Follow us



[Twitter.com/Kaspersky](https://twitter.com/Kaspersky)



[Facebook.com/Kaspersky](https://facebook.com/Kaspersky)



[Youtube.com/Kaspersky](https://youtube.com/Kaspersky)