Basque CyberSecurity Centre

PLAY Ransomware

BCSC-MALWARE-PLAY

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

PLAY, también conocido como ***PlayCrypt***, es un malware de tipo *ransomware* que afecta a sistemas Windows y que apareció por primera vez en julio de 2022, reportándose el primero caso de forma pública en agosto de 2022 cuando un periodista descubrió que el Poder Judicial de Córdoba (Argentina) fue víctima de un ataque por esta familia de *ransomware*.

Los actores detrás de PLAY siguen un modelo de doble extorsión, muy común en otras familias de *ransomware*, en el que tras el compromiso de la red de la víctima exfiltran cierta información sensible de la compañía previamente al cifrado de los equipos y utilizan la amenaza de publicar esta información como mecanismo para presionar a la víctima a pagar el rescate solicitado.

El vector de entrada preferido por los actores que operan este malware parece ser la explotación de equipos Fortinet SSL VPN o Exchange vulnerables a exploits de acceso remoto y el uso de credenciales válidas adquiridas en mercados ilegales.

PLAY está desarrollado en lenguaje Visual C++ y las muestras detectadas hasta la fecha no parecen presentar ofuscación mediante software de empaquetado. No obstante, sí que implementa ciertas técnicas como la ofuscación del flujo del programa mediante rutinas de salto y código basura, el cifrado de cadenas de caracteres u ocultación de las API utilizadas por el malware con el doble objetivo de dificultar la tarea de aplicar ingeniería inversa y evadir el *software* de detección antivirus.

Las técnicas y conjunto de herramientas utilizados por los operadores de PLAY coinciden en gran medida a las de otros actores como Hive o Nokoyawa, aunque el código de su *ransomware* sí es completamente diferente.

El sistema de cifrado de ficheros implementado por PLAY se basa en los algoritmos RSA y AES, los cuales son un tipo de cifrado robusto y la generación de las claves de cifrado aleatorias se realiza mediante librerías seguras con el objetivo de garantizar que los ficheros cifrados no puedan ser recuperados salvo que se pague el rescate.

Por todo esto, PLAY representa una amenaza significativa que está alcanzando un gran impacto por lo que debe ser tenida en especial consideración.

Análisis técnico

Flujo de infección

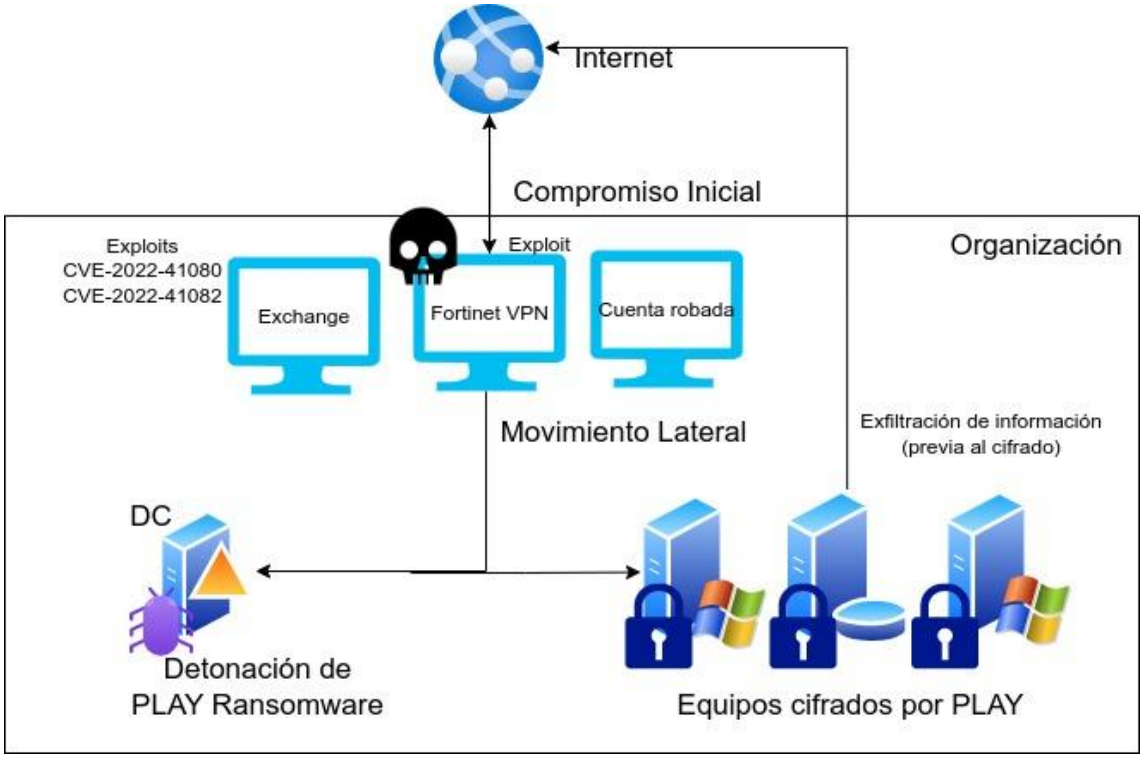


Ilustración 1: Flujo de infección de PLAY Ransomware.

El análisis de diferentes casos publicados que involucran a PLAY indica que sus actores utilizan tácticas similares a grupos como Hive o Nokoyawa, incluyendo nombres de archivos, rutas, herramientas y payloads como por ejemplo Nekto, Cobalt Strike, Coroxy, GMER o PsExec. No obstante, un comportamiento que distingue a PLAY del resto es el uso de AdFind, una herramienta de línea de comandos capaz de recopilar información del Directorio Activo.

Recientes incidentes relacionados con esta familia de *ransomware* muestran cierta preferencia de sus actores por obtener acceso inicial mediante el compromiso de cuentas válidas o explotación de sistemas Fortinet SSL VPN. Por otra parte, también se han detectado casos en los que se utiliza el exploit conocido como OWASSRF para explotar la vulnerabilidad CVE-2022-41080, que realiza una escalada remota de privilegios en servidores Exchange, o la CVE-2022-41082, que permite la ejecución remota de código y es el mismo error explorado en los ataques ProxyNotShell.

Tras esto, al igual que otros actores, aprovechan los binarios legítimos ya presentes en el sistema (LOLBins o Living Off the Land Binaries) como parte de su conjunto de herramientas para llevar a cabo el ataque. Los actores detrás de PLAY también llevan a cabo un sistema de doble extosión: en sus ataques, se realiza una exfiltración de datos previo a la ejecución del *ransomware*. Los

ficheros de las víctimas suelen ser comprimidos mediante herramientas como WinRAR para cargarlos posteriormente en sitios de compartición de archivos conocidos. El binario del *ransomware* se distribuye posteriormente mediante una política de grupo (GPO) y ejecutado por tareas programadas, PsExec o WMIC.

Portal de PLAY en la red TOR

Los actores de ***PLAY*** cuentan con un sitio en la red TOR para enumerar las organizaciones supuestamente afectadas por su *ransomware* y una cuenta atrás para la publicación de los datos recopilados por ellos en caso de no pagar el rescate demandado.

En este sitio hay además un portal donde se puede contactar al grupo, una sección de "Preguntas frecuentes" que describe ampliamente lo que ha hecho el grupo y los pasos que deben seguir las víctimas para restaurar sus datos.

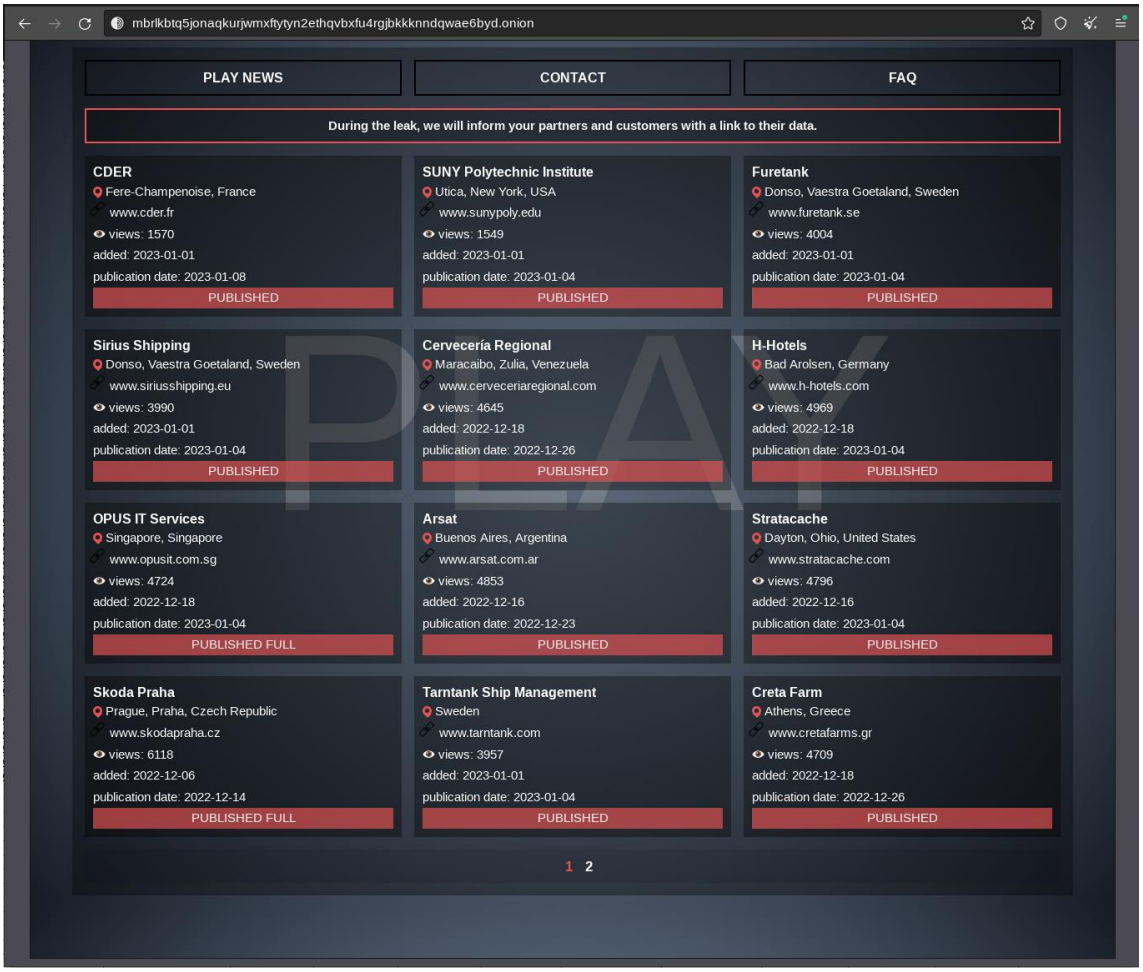


Ilustración 2: Sitio oficial de los actores de PLAY en la red TOR

Las direcciones actuales para acceder a este sitio son las siguientes:

hxxp://mbrlkbtq5jonaqkurjwmxftytyn2ethqvbxfu4rgjbkkknndqwae6byd.onion hxxp://k7kg3jqxang3wh7hnmaiokchk7qoebupfgoik6rha6mjpzwupwtj25yd.onion

Muestra analizada

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

006ae41910887f0811a3ba2868ef9576bbd265216554850112319af878f06e55

El binario está desarrollado con Visual C++ y no parece encontrarse empaquetado mediante ningún software de protección.

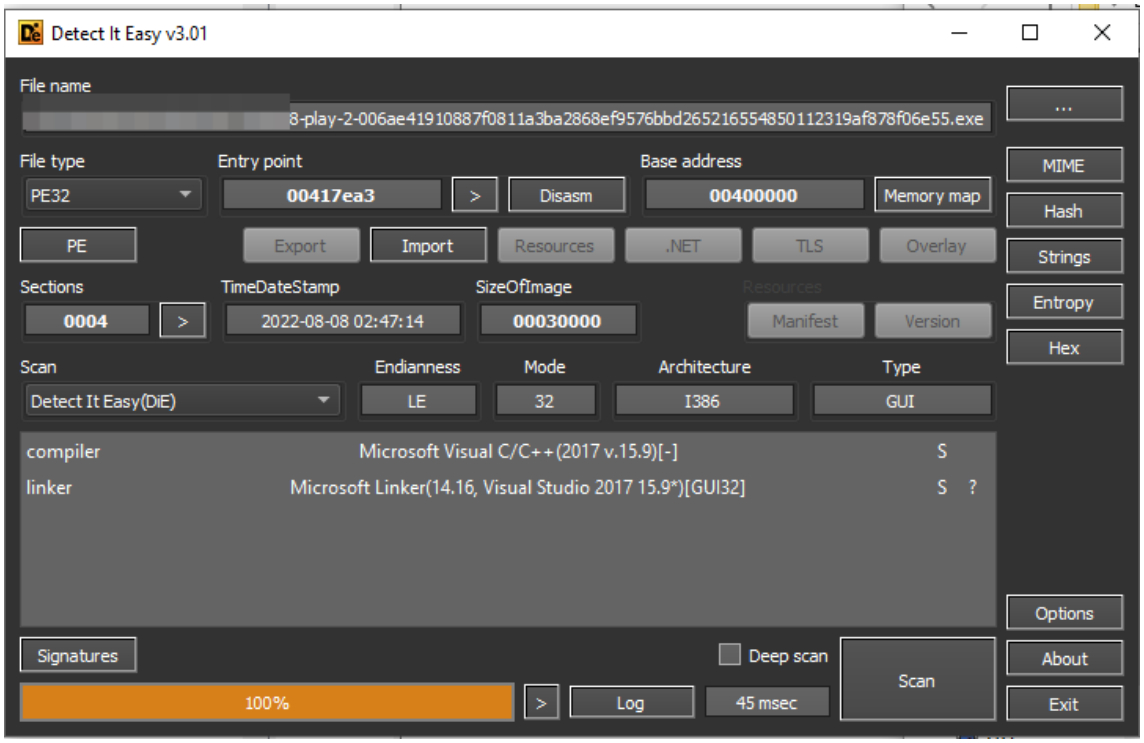


Ilustración 3: análisis de la muestra de PLAY en el software Detect It Easy (DIE)

Protecciones Anti-Análisis: flujo del programa

Tras desensamblar el código del binario, se puede comprobar que la mayor parte del código ensamblador no tiene mucho sentido. La rutina principal WinMain no contiene una declaración de retorno clara y aparecen bytes basura entre el código válido.

```
; int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)
_WinMain@16     proc near               ; CODE XREF: __scrt_common_main_seh(void)+F3↓p

hInstance       = dword ptr  8
hPrevInstance   = dword ptr  0Ch
lpCmdLine       = dword ptr  10h
nShowCmd        = dword ptr  14h

                push    ebp
                mov     ebp, esp
                push    ebx
                push    esi
                push    edi
                sub     esp, 0Ch
                mov     ax, 5AE5h
                cmp     ax, 5834h
                jg      short loc_4142E9
                add     esp, 183h

loc_4142E9:                             ; CODE XREF: WinMain(x,x,x,x)+11↑j
                add     esp, 0Ch
                call    sub_4142F5
                pop     edi
_WinMain@16     endp ; sp-analysis failed

; ---------------------------------------------------------------------------
                db 0A0h
                db  93h
                db  99h

; =============== S U B R O U T I N E =======================================


sub_4142F5      proc near               ; CODE XREF: WinMain(x,x,x,x)+1C↑p
                add     dword ptr [esp+0], 35h ; '5'
                retn
sub_4142F5      endp

; ---------------------------------------------------------------------------
                dw 3716h
                dd 8663A98Dh, 31268684h, 664EC078h, 0D497B3Eh, 52B226C7h
```

Ilustración 4: funcionalidad anti-decompilado en la rutina WinMain

Se puede observar que el flujo de la rutina WinMain llama a la función **sub_4142F5**. Tras regresar de esta, se observan bytes extraños que el desensamblador no ha sabido interpretar correctamente. Como resultado, IDA Pro muestra un código decompilado sin mucho sentido.

```
1 // positive sp value has been detected, the output may be wrong!
2 int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine,
3 {
4   sub_4142F5();
5   JUMPOUT(0x4142F2);
6 }
```

Ilustración 5: código decompilado de la rutina WinMain previo al parcheo

Examinando el código de la rutina **sub_4142F5**, se puede observar que al valor almacenado en el puntero de pila (registro ***esp***) se le suma 0x35 justo antes de retornar mediante la instrucción ***retn***.

Cuando se ejecuta una instrucción de tipo ***call*** como la de la llamada a la función **sub_4142F5**, internamente se añade a la pila la dirección a la que volver después del ***call*** y se salta a la dirección que esté siendo llamada. Cuando el flujo del programa salta a la función **sub_4142F5**, la dirección a la que volver se almacena en el punetero de pila es **0x4142F1**. Sin embargo, la función toma este valor y le añade 0x35 por lo que la función retornará a la dirección **0x414326** cuando se ejecute la instrucción ***retn***.

```
t:004142F5
t:004142F5 ; =============== S U B R O U T I N E =======================================
t:004142F5
t:004142F5
t:004142F5 sub_4142F5       proc near               ; CODE XREF: WinMain(x,x,x,x)+1C↑p
t:004142F5                  add     dword ptr [esp+0], 35h ; '5'
t:004142F9                  retn
t:004142F9 sub_4142F5       endp
t:004142F9
t:004142F9 ; ---------------------------------------------------------------------------
t:004142FA                  dd 0A98D3716h
t:004142FE                  dd 86848663h
t:00414302                  dd 0C0783126h
t:00414306                  dd 7B3E664Eh
t:0041430A                  dd 26C70D49h
t:0041430E                  dd 164E52B2h
t:00414312                  dd 0B4972D55h
t:00414316                  dd 396F2573h
t:0041431A                  dd 0A1FDFB65h
t:0041431E                  dd 0D99A80FEh
t:00414322                  dd 0D1492D69h
t:00414326 ; ---------------------------------------------------------------------------
t:00414326                  sub     esp, 0Ch
t:00414329                  mov     al, 7Ch ; '|'
t:0041432B                  mov     dl, 50h ; 'P'
t:0041432D                  cmp     al, dl
t:0041432F                  jg      short loc_414337
t:00414331                  add     esp, 15Bh
t:00414337
t:00414337 loc_414337:                              ; CODE XREF: .text:0041432F↑j
t:00414337                  add     esp, 0Ch
t:0041433A                  call    sub_41435B
t:0041433F                  sti
```

Ilustración 6: salto a código oculto ahora revelado

Esta técnica se repite a lo largo de todo el código del programa y se conoce como Programación Orientada al Retorno (ROP). Como se puede observar, su uso desvía el flujo regular del programa y, de esta forma, PLAY puede eludir la mayoría de análisis estáticos basados en desensamblado y decompilación como los de IDA Pro, dificultando así enormemente la tarea de análisis.

Para tratar de hacer frente a esta técnica de ofuscación, es necesario recorrer el código del programa, localizar mediante algún tipo de patrón como, por ejemplo, mediante expresiones regulares, estas rutinas y parchear todas las instrucciones ***call*** que tengan dicho comportamiento por un salto (***jmp***) a la instrucción calculada por la subrutina.

Tras aplicar este script, el código de la rutina principal WinMain cambia a una llamada a otra función que ahora sí tiene sentido y una instrucción de retorno adecuada.

```
1  // positive sp value has been detected, the output may be wrong!
2  int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lp
3  {
4    char v5[4]; // [esp+8Ch] [ebp-3Ch] BYREF
5    char v6[4]; // [esp+90h] [ebp-38h] BYREF
6    char v7[4]; // [esp+94h] [ebp-34h] BYREF
7    char v8[4]; // [esp+98h] [ebp-30h] BYREF
8
9    strcpy(v5, "nt");
10   strcpy(v6, "rs");
11   strcpy(v7, "i6");
12   strcpy(v8, "bN");
13   sub_446690();
14   return 1;
15 }
```

Ilustración 7: rutina WinMain ya parcheada

Protecciones Anti-Análisis: código basura

Además de la ofuscación del flujo del programa, PLAY también implementa en su código instrucciones de movimiento de registros aleatorias que no afectan a la funcionalidad principal del programa y únicamente sirven para distraer al analista.

```
.text:004169D0 loc_4169D0:                             ; CODE XREF: sub_416980+2D↑j
.text:004169D0                 movq    xmm0, ds:qword_42A084
.text:004169D8                 lea     esi, [ebp+var_39D8]
.text:004169DE                 mov     eax, ds:dword_42A07C
.text:004169E3                 mov     edi, 1Bh
.text:004169E8                 movq    [ebp+var_3A18], xmm0
.text:004169F0                 movq    xmm0, ds:qword_42A090
.text:004169F8                 movq    [ebp+var_3A30], xmm0
.text:00416A00                 movq    xmm0, ds:qword_42A09C
.text:00416A08                 movq    [ebp+var_3A24], xmm0
.text:00416A10                 movups  xmm0, ds:xmmword_42A0A8
.text:00416A17                 mov     [ebp+var_39F8], eax
.text:00416A1D                 movzx   eax, ds:word_42A098
.text:00416A24                 movups  [ebp+var_3A58], xmm0
.text:00416A2B                 mov     [ebp+var_3A28], ax
.text:00416A32                 movups  xmm0, ds:xmmword_42A0C0
.text:00416A39                 mov     eax, ds:dword_42A0A4
.text:00416A3E                 mov     [ebp+var_3A19], 0
.text:00416A45                 movups  [ebp+var_3A90], xmm0
.text:00416A4C                 mov     [ebp-3A1Ch], eax
.text:00416A52                 mov     eax, 38Ch
.text:00416A57                 movq    xmm0, ds:qword_42A0D0
.text:00416A5F                 movq    [ebp+var_3A80], xmm0
.text:00416A67                 movups  xmm0, ds:xmmword_42A0DC
.text:00416A6E                 mov     [ebp+var_39DD], 4Dh ; 'M'
.text:00416A75                 mov     [ebp+var_39E1], 61h ; 'a'
.text:00416A7C                 movups  [ebp+var_3A74], xmm0
.text:00416A83                 mov     [ebp+var_39F0], eax
.text:00416A89                 movq    xmm0, ds:qword_42A0EC
.text:00416A91                 movq    [ebp+var_3A64], xmm0
.text:00416A99                 movq    xmm0, ds:qword_42A0F8
.text:00416AA1                 movq    [ebp+var_3A40], xmm0
.text:00416AA9                 nop     dword ptr [eax+00000000h]
.text:00416AB0
.text:00416AB0 loc_416AB0:                             ; CODE XREF: sub_416980+140↓j
.text:00416AB0                 mov     ecx, esi
.text:00416AB2                 call    sub_416DF0
.text:00416AB7                 add     esi, 218h
.text:00416ABD                 sub     edi, 1
.text:00416AC0                 jnz     short loc_416AB0
.text:00416AC2                 lea     esi, [ebp+var_150]
.text:00416AC8                 mov     edi, 1Bh
0015DDD0 004169D0: sub_416980:loc_4169D0 (Synchronized with Hex View-1)
```

Ilustración 8: ejemplo de código basura entre código válido en PLAY

Esto hace que el código decompilado se vea mucho más extenso de lo que en realidad es y sea más difícil seguir su funcionalidad. Además, este tipo de instrucciones no son fáciles de parchear como en el caso de la ofuscación anterior, ya que el código válido se encuentra incrustado entre medias del código basura por lo que se opta por simplemente tratar de ignorar su presencia y centrarse en las instrucciones que sí que puedan tener sentido.

```
v0 = v38;
v1 = 27;
*(_QWORD *)v21 = 0x7044577768646735i64;
strcpy(v18, "upsvYgElq");
v19 = 0x6C005A00310045i64;
v26 = 1937008998;
v20 = 122;
v33 = 77;
v31 = 97;
v27 = 908;
do
{
  ((void (__thiscall *)(char *, _DWORD))sub_416DF0)(v0, *(_DWORD *)&v21[6]);
  v0 += 536;
  --v1;
}
while ( v1 );
v2 = v39;
v3 = 27;
do
{
  ((void (__thiscall *)(char *))sub_4116C0)(v2);
  v2 += 8;
  --v3;
}
while ( v3 );
v22[0] = 0;
v22[1] = 27;
v23 = v38;
((void (__thiscall *)(int *))sub_409860)(v22);
v4 = 119;
```

Ilustración 9: ejemplo de código basura decompilado

Protecciones Anti-Análisis: API Hashing

Al igual que ocurre con otras familias de *ransomware* moderno, PLAY ofusca sus llamadas a API del sistema mediante la técnica de API hashing. La función encargada de esto toma como parámetro un valor *hash* y la dirección de una DLL. La rutina itera sobre la tabla de exportaciones de la librería para obtener el nombre de cada una de las entradas. Para cada nombre de API, el malware llama a la función **sub_40F580** con el nombre y el valor 1 como parámetros y, al resultado de esta función se le suma el valor constante **0x4E986790** para formar el valor *hash* final. Si se encuentra una coincidencia, se devuelve la dirección de la API para poder ser llamada posteriormente por el malware.

```
while ( 1 )
{
  api_name = (const char *)(v15 + *v12);
  v36 = v14 + v13;
  v17 = v26;
  v5 += v27 - 1;
  produced_hash = xxHash32((int)api_name, strlen(api_name), 1u) + 0x4E986790;
  if ( BYTE1(v21[0]) && (_WORD)v33 )
  {
    v33 = v5 - 1;
  }
  else
  {
    v17 = v9 + v36;
    LOWORD(v33) = v9 + v35 + 97;
  }
  if ( produced_hash == target_hash )
    break;
  if ( v17 )
    v5 = BYTE2(v21[1]) + 1;
  ++v31;
  v12 = v30 + 1;
  v19 = v29 + 1 < v24;
  v13 = v36;
  v14 = v34;
  ++v29;
  v15 = v32;
  ++v30;
  if ( !v19 )
    return 0;
}
```

Ilustración 10: API Hashing en PLAY

La función de hashing contiene gran cantidad de constantes únicas que permiten identificar al algoritmo como xxHash32 con una semilla, *seed*, con valor 1 y a cuyo resultado se le añade la constante comentada anteriormente.

```
unsigned int __fastcall xxHash32(int a1, unsigned int a2, int a3)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v27 = a1;
  v3 = 0;
  v4 = a2;
  if ( a1 )
  {
    if ( a2 < 0x10 )
    {
      v12 = a3 + 0x165667B1;
    }
    else
    {
      v6 = a3;
      v22 = a3 + 0x24234428;
      v7 = a3 + 0x61C8864F;
      v23 = a3 - 0x7A143589;
      v26 = a1 + 3;
      v25 = a1 + 2;
      v8 = a1 + 3;
      v24 = a1 + 1;
      v21 = v4 >> 4;
      do
      {
        v4 -= 0x10;
        v22 = 0x9E3779B1
            * __ROL4__(
                v22
              - 0x7A143589
              * (*(unsigned __int8 *)(v3 + v27) | ((*(unsigned __int8 *)(v24 + v3) | ((*(unsigned __int8 *)(v25
                13);
```

Ilustración 11: función de hash xxHash32

Conociendo esto, es posible implementar un script que automáticamente resuelva todas las API que el malware va a utilizar y se aplique un renombrado en IDA Pro que facilite la tarea de análisis.

```
HIBYTE(v30) = LOBYTE(v18[1]) + 1;
LOWORD(v28) = 114;
LoadLibraryA = mw_resolve_API(v4, -1098775628, 0);
v27 = 8274;
v29 = 114 * v19[4];
VirtualAlloc = mw_resolve_API(v4, 587706791, 1);
LOWORD(v1) = 3766;
v26 = v1;
VirtualFree = mw_resolve_API(v4, 430125299, 1);
v17 = v19[2] * v29;
v20 = v27;
FindFirstFileW = mw_resolve_API(v4, 745928694, 1);
v28 = (v28 - 1);
v22 = v28;
FindNextFileW_0 = mw_resolve_API(v4, -984644163, 1);
FindClose_0 = mw_resolve_API(v4, -1756832492, 1);
v23 = 77;
CreateFileA = mw_resolve_API(v4, -2134016500, 1);
CreateFileW_0 = mw_resolve_API(v4, -972994119, 1);
ReadFile = mw_resolve_API(v4, 2119526180, 1);
if ( v24 - v25 == 2 )
{
```

Ilustración 12: resolución de API mediante script automático

Protecciones Anti-Análisis: cifrado de cadenas de caracteres

La gran mayoría de cadenas de caracteres de PLAY son descifradas en tiempo de ejecución. El algoritmo de descifrado parece bastante complejo, teniendo además en cuenta la inserción de múltiple código basura por lo que se ha optado por la ejecución dinámica para extraer las cadenas de caracteres utilizadas en cada caso.

```
{
  *Destination = 0;
  mw_decrypt_string(&unk_42CA34, 6u, v15, v16, &v22);// "-d"
  v3 = wcscmp(v2[1], &v22);
  if ( v3 )
    v3 = v3 < 0 ? -1 : 1;
  if ( !v3 )
  {
    mw_decrypt_string(&unk_42B968, 0xAu, v15, v16, Source);// "\\?\"
    wcscpy_s(Destination, 0x104u, Source);
    v20 = 0;
    *Source = 0i64;
    wcscat_s(Destination, 0x104u, v2[2]);
    mw_decrypt_string(&unk_42CA3C, 4u, v15, v16, v23);
```

Ilustración 13: función de descifrado de cadenas en PLAY

Argumentos del programa

PLAY puede ejecutarse con o sin parámetros. A continuación, se indica el listado de parámetros que el operador le puede indicar al malware:

- **-mc**: ejecutar de forma normal, igual que si no se le pasara ningún parámetro
- **-d <disco>**: cifrar un disco en específico
- **-d <ruta>**: cifrar un fichero/carpeta específicos
- **-ip <ruta-de-recurso-compartido> <usuario> <contraseña>**: cifrar un recurso de red compartido

```
  mw_encrypt_target_path(Destination);
  return 1;
}
mw_decrypt_string(&unk_42CA48, 8u, v16, v17, &v22);// "-ip"
v5 = wcscmp(argv[1], &v22);
if ( v5 )
  v5 = v5 < 0 ? -1 : 1;
if ( !v5 )
{
  wcscpy_s(Destination, 0x104u, argv[2]);
  if ( v13 == 5 )
  {
    wcscpy_s(v14, 0x104u, argv[3]);
    wcscpy_s(v15, 0x104u, argv[4]);
    username = v14;
    password = v15;
  }
  else
  {
    password = 0;
    username = 0;
  }
  mw_ecrypt_network_shared_resource(Destination, username, password);
  return 1;
}
mw_decrypt_string(&unk_42CA40, 6u, v16, v17, &v23);// "-p"
v7 = wcscmp(argv[1], &v23);
if ( v7 )
  v7 = v7 < 0 ? -1 : 1;
if ( !v7 )
{
  wcscpy_s(Destination, 0x104u, argv[2]);
  mw_encrypt_target_path(Destination);
  return 1;
```

Ilustración 14: procesamiento de los argumentos pasados a PLAY

Inicialización de algoritmos criptográficos

Antes de comenzar el cifrado, PLAY inicializa y recupera los proveedores de algoritmos criptográficos. Primero llama a **BCryptOpenAlgorithmProvider** para cargar e inicializar un proveedor de CNG para la generación de números aleatorios y **BCryptImportKeyPair** para importar su clave pública RSA codificada.

```
107  v6 = v59;
108  v7 = w_BCryptOpenAlgorithmProvider(BCRYPT_RNG_PROVIDER, RNG_str);
109  *v66 = 0i64;
110  if ( v7 )
111    return -2;
112  mw_decrypt_string(&PLAY_RSAPUBLICBLOB, 0x1Cu, v32, v40, RSAPUBLICBLOCK_str);
113  v9 = w_BCryptImportKeyPair(RSAPUBLICBLOCK_str, v8, v33);
114  memset(RSAPUBLICBLOCK_str, 0, sizeof(RSAPUBLICBLOCK_str));
115  if ( v9 )
116    return -7;
117  v11 = w_VirtualAlloc(v10, 516);
```

Ilustración 15: inicialización e importación de clave criptográfica pública RSA

A continuación, el malware llama a **VirtualAlloc** para reservar un buffer y almacenar 128 estructuras utilizadas para cifrar los ficheros. El tamaño de estas estructuras es 0x48 bytes y su contenido es el siguiente:

```
struct play_file_struct
{
    int struct_index;
    char *filename;
    int initialized_flag;
    int padding1;
    char *file_path;
    int file_marker[2];
    int chunk_count;
    int chaining_mode_flag;
    DWORD large_file_flag;
    HANDLE AES_provider_handle;
    HANDLE bcrypt_RNG_provider;
    HANDLE RSA_pub_key_handle;
    HANDLE file_handle;
    LARGE_INTEGER file_size;
    DWORD file_data_buffer;
    DWORD padding2;
};
```

A continuación, se detalla el significado de cada variable:

- **`struct_index`**: índice de la estructura en la lista global de estructuras
- **`filename`**: nombre del fichero a ser procesado
- **`initialized_flag`**: inicializado a 1 cuando la estructura es rellenada con un fichero a cifrar
- **`file_path`**: ruta del fichero a ser procesado
- **`file_marker`**: marcadores constantes para escribir en el pie de página del archivo que indica que se ha cifrado
- **`chunk_count`**: número de chunks a cifrar en el fichero
- **`chaining_mode_flag`**: inicializado a 1 para usar GCM o a 0 para usar CBC en el cifrado AES
- **`large_file_flag`**: inicializado a 1 cuando el fichero es grande
- **`AES_provider_handle`**: manejador al proveedor del algoritmo AES
- **`bcrypt_RNG_provider`**: manejador al proveedor del algoritmo RNG
- **`RSA_pub_key_handle`**: manejador a la clave pública RSA

- **`file_handle`**: manejador del fichero
- **`file_size`**: tamaño de fichero
- **`file_data_buffer`**: dirección del buffer virtual para leer el contenido del fichero

PLAY itera sobre esta lista de estructuras global y completa el campo correspondiente de cada estructura. Primero establece los marcadores FILE_MARKER_1 y FILE_MARKER_2 para rellenar el valor de la variable `file_marker`.

```
429AC8 arj15g           db 'rj15g',0
429ACE                  align 10h
429AD0 FILE_MARKER_1    db  96h                 ; DATA XREF: sub_4151D0+4C
429AD1                  db 0ABh
429AD2                  db 0CEh
429AD3                  db  54h ; T
429AD4                  db  93h
429AD5                  db  1Eh
429AD6                  db  55h ; U
429AD7                  db  82h
429AD8                  db  7Ch ; |
429AD9                  db  84h
429ADA                  db  21h ; !
429ADB                  db  1Ah
429ADC                  db  49h ; I
429ADD                  db  3Eh ; >
429ADE                  db  87h
429ADF                  db 0A7h
429AE0 FILE_MARKER_2    db 0FAh                 ; DATA XREF: sub_4151D0+4D
429AE1                  db  8Fh
429AE2                  db 0CFh
429AE3                  db 0F4h
429AE4                  db  35h ; 5
429AE5                  db 0EBh
429AE6                  db 0A4h
429AE7                  db  35h ; 5
429AE8 aRsa1            db 'RSA1',0             ; DATA XREF: w_BCryptImpor
429AED                  db ' ',0
```

Ilustración 16: marcadores de los archivos cifrados

A continuación, el malware asigna los manejadores de los algoritmos criptográficos RNG y AES, así como el manejador de la clave pública que serán utilizados para generar claves y vectores de inicialización (IV) AES aleatorios.

```
v56 += 2;
v22 = v63 - 98;
file_struct[5] = &FILE_MARKER_1;
file_struct[6] = &FILE_MARKER_2;
file_struct[3] = 1;
file_struct[2] = 0;
if ( v20 < 0x55 )
  v6 = 107 * v60;
v23 = v56 + v22;
v24 = v61;
*file_struct = struct_index_1;
v25 = v6 + v24;
file_struct[11] = &::BCRYPT_RNG_PROVIDER;
file_struct[12] = &BCRYPT_RSA_PUB_KEY_HANDLE;
v54 = v23 * v23;
v61 = v6 + LOBYTE(v51[1]) - 104;
mw_decrypt_string(&unk_42CA50, 8u, v35, v41, AES_str);// AES
v64 += v49[6];
++v63;
```

Ilustración 17: inicialización de estructurura de archivos

Comprobación de discos existentes

Antes de comenzar a iterar sobre los ficheros, PLAY necesita enumerar todos los volúmenes que la víctima tiene en su sistema mediante las API **FindFirstVolumeW** y **FindNextVolumeW**. Si el dispositivo no es una unidad de CD-ROM o de RAM, el malware llama a **GetVolumePathNamesForVolumeNameW** para devolver una lista de la letra asociada al disco y las rutas de carpetas montadas para ese volumen.

Si la lista se encuentra vacía, significa que el volumen no se encuentra montado en ninguna carpeta. PLAY llama a **GetDiskFreeSpaceExW** para comprobar si el espacio en el disco libre es mayor a 0x40000000. Si lo es, llama a **SetVolumeMountPointW** para intentar montarlo.

```
volume_path_name_len = 0;
FindFirstVolumeW = mw_assign_resolved_api(::FindFirstVolumeW);
find_volume_handle_1 = FindFirstVolumeW(volume_name, 260);
v5 = 3778;
v6 = 4;
find_volume_handle = find_volume_handle_1;
do
{
  ++v5;
  --v6;
}
while ( v6 );
if ( find_volume_handle_1 != INVALID_HANDLE_VALUE )
{
  v24 = 116;
  v23 = 104;
  v21 = 86;
  do
  {
    GetDriveTypeW = mw_assign_resolved_api(::GetDriveTypeW);
    drive_type = GetDriveTypeW(volume_name);
    if ( drive_type != DRIVE_CDROM && drive_type != DRIVE_RAMDISK )
    {
      GetVolumePathNamesForVolumeNameW = mw_assign_resolved_api(::GetVolumePathNamesForVolumeNameW);
      if ( !GetVolumePathNamesForVolumeNameW(volume_name, &volume_path_name, 520, &volume_path_name_len) )
        v23 = v22 + v19 + 1;
      v19 = -76;
      if ( volume_path_name_len <= 1 )
      {
        LODWORD(volume_free_space) = w_GetDiskFreeSpaceExW(volume_name);
        if ( volume_free_space > 0x40000000 )
        {
          sub_40A850(volume_name);
          v22 = 0;
        }
        v24 += 3;
        v21 += 3;
      }
      v23 -= 86;
    }
    FindNextVolumeW = mw_assign_resolved_api(::FindNextVolumeW);
  }
  while ( FindNextVolumeW(find_volume_handle, volume_name, 260) );
```

Ilustración 18: enumeración de volúmenes

Para cada volumen a ser montado, PLAY itera sobre los caracteres para encontrar una letra que pueda utilizar para llamar a **SetVolumeMountPointW** y montarlo.

Usando la misma técnica para iterar a través de todos los nombres de unidades posibles, PLAY llama a **GetDriveTypeW** para verificar el tipo de cada unidad. Evita cifrar la unidad de CD-ROM o el disco RAM. Si se trata de una unidad remota, el malware llama a **WNetGetUniversalNameW** para recuperar el nombre universal de la unidad de red.

```
{
  if ( v9 == DRIVE_REMOTE || v9 == DRIVE_NO_ROOT_DIR )
  {
    if ( v13 )
    {
      HIDWORD(v51) = v39[4] - v44;
      v16 = v38 + v51 - 2 + v47;
    }
    else
    {
      v16 = v11 + WORD1(v31) + 1;
    }
    v45 = v16;
    *&v35[2] = (v51 + v50);
    v47 = WORD3(v33);
    lpBufferSize = 260;
    v41 = *&v35[2];
    LOWORD(v40) = v51 + v50;
    error_code = w_WNetGetUniversalNameW(full_drive_path, &lpBufferSize, drive_remote_name_info);
    v12 = HIDWORD(v51) * v39[4];
    v44 = 1;
    v38 = v12;
    v50 = 5400;
```

Ilustración 19: procesamiento de unidades de red

La ruta de la unidad final que se cifrará se establece en el nombre universal o el nombre de conexión de la unidad de red, según cuál exista.

```
{
  ++Destination[0];
  v65 = v49 + v33 + 1;
  goto LABEL_32;
}
v42 = *(&v31 + 2) - v33 - 1;
WORD1(v40) = WORD1(v31) - v33 - 1;
drive_remote_name_to_encrypt = (full_drive_name[2] + 536 * *full_drive_name);
if ( drive_remote_name_info.lpUniversalName )
{
  wcscpy_s(drive_remote_name_to_encrypt, 0x104u, drive_remote_name_info.lpUniversalName);
  LOBYTE(v50) = v45 * v39[2];
}
else
{
  wcscpy_s(drive_remote_name_to_encrypt, 0x104u, drive_remote_name_info.lpConnectionName);
}
v15 = v67;
v58 += 114;
v62 += 105;
```

Ilustración 20: obtención del nombre de la unidad de red

Si la unidad es una unidad normal, su nombre sigue siendo el mismo. A cada unidad válida se le agrega su nombre a la lista de nombres de unidades que se recorrerán y cifrarán.

Recorrido de directorios recurisvo

Para comenzar a recorrer unidades, PLAY itera a través de la lista de nombres de unidades anterior y genera un hilo con **CreateThread** para recorrer cada unidad en el sistema.

```
v32 = 119;
do
{
  drive_index = 0;
  if ( drive_count > 0 )
  {
    v9 = v23;
    drive_path = v39;
    do
    {
      *drive_path = v9;
      *(drive_path + 1) = 0;
      if ( v4 <= 1u )
      {
        v29 = v21[3] + 1503;
        LOWORD(v7) = v7 - 1;
        v28 = v7;
      }
      threadHandle = w_CreateThread(mw_recursive_traverse, drive_path);
      v4 = v32;
      thread_handles[drive_index] = threadHandle;
      v9 += 536;
      v7 = v28;
      ++drive_index;
      drive_path += 8;
    }
    while ( drive_index < drive_count_cpy[0] );
    v6 = v27;
```

Ilustración 21: generación de subprocesos de recorrido transversal de unidades

Antes de procesar una unidad, el malware extrae el siguiente contenido de la nota de rescate antes de colocarlo en la carpeta de la unidad. Este es el único lugar donde se coloca la nota de rescate en lugar de en todas las carpetas como otros *ransomware*. Las cadenas descifradas para añadir al contenido de la nota son las siguientes:

PLAY teilightomemaucd@gmx.com

```
full_drive_path = result;
if ( result )
{
  *result = 0;
  wcscpy_s(result, 0x7FFFu, *drive_path_1);
  v11 = 0;
  mw_string_decrypt_2(&unk_42B974, 0x1F, v16, v17, &ransom_note_content);// PLAY
                                                    // teilightomemaucd@gmx.com
  mw_drop_ransom_note(full_drive_path, ransom_note_content);
  v9 = 0;
  v10 = 0;
  ransom_note_content = 0i64;
  v8 = 0i64;
  drive_path_2 = *(drive_path_1 + 4);
  v11 = 0;
  v5 = mw_recursive_traverse_2(drive_path_2);
```

Ilustración 22: escritura de la nota en disco

Para comenzar a enumerar, el malware llama a **FindFirstFileW** y **FindNextFileW** para enumerar subcarpetas y archivos. Comprueba específicamente para evitar el procesamiento de las rutas del directorio actual y principal "." y "..".

```
wcscat_s(drive_find_path, 0x7FFFu, &Source);
FindFirstFileW = mw_assign_resolved_api(::FindFirstFileW);
findFileHandle = FindFirstFileW(drive_find_path, &find_file_data);
findFileHandle_cpy = findFileHandle;
if ( findFileHandle != -1 )
{
  mw_remove_last_char(drive_find_path);
  v24 = parent_dir_str;
  v26 = 109;
  do
  {
    mw_decrypt_string(
      &unk_42B940,
      4u,
      *&find_file_data.cFileName[220],
      *&find_file_data.cFileName[222],
      &curr_dir_str);
    v6 = wcscmp(find_file_data.cFileName, &curr_dir_str);// "."
    if ( v6 )
      v6 = v6 < 0 ? -1 : 1;
    v24 -= 2;
    v23 -= 2;
    LOWORD(v19) = v19 + 4;
    curr_dir_str = 0i64;
    v7 = WORD2(v19) + 13308;
    WORD2(v19) += 13308;
    mw_decrypt_string(
      &unk_42C980,
      6u,
      *&find_file_data.cFileName[220],
      *&find_file_data.cFileName[222],
      &parent_dir_str);
    v8 = wcscmp(find_file_data.cFileName, &parent_dir_str);// ".."
    if ( v8 )
      v8 = v8 < 0 ? -1 : 1;
    v26 += v22;
    parent_dir_str = 0i64;
```

Ilustración 23: enumeración de ficheros

Si el archivo encontrado es un directorio, el malware verifica para evitar cifrar el directorio "Windows". Después de eso, concatena el nombre del subdirectorio con la ruta de búsqueda del archivo actual y recorre recursivamente el subdirectorio llamando a la función recursivamente en él.

```
parent_dir_str = 0i64;
v34 = 0;
if ( v6 && v8 )
{
  v23 = 204;
  if ( (find_file_data.dwFileAttributes & FILE_ATTRIBUTE_DIRECTORY) != 0 )
  {
    v9 = &unk_42C7E0;                   // directory
    v10 = 0;
    v11 = 99;
    while ( 1 )
    {
      *&find_file_data.cFileName[0xFB] = 0x34;
      if...
      mw_decrypt_string(v29, 0x68u, *&find_file_data.cFileName[247], *&find_file_data.cFileName[249], windows_str);// "Windows"
      v11 -= 2;
      v12 = wcscmp(windows_str, find_file_data.cFileName);
      if...
      if...
      v10 += 52;
      v9 += 52;
      if ( v10 >= 0x1A0 )
      {
        if ( wcscat_s(drive_find_path, 0x7FFFu, find_file_data.cFileName) )// Concatenar nombre del subdir al file find path
        {
          LOWORD(v19) = 22095 - v22;
          v22 = 120;
        }
        else
        {
          --v24;
          mw_recursive_traverse_2(sub_dir_path);// Recursividad para recorrer el subdirectorio
```

Ilustración 24: llamada recursiva para recorrer directorios

Si el archivo encontrado es un archivo normal, el malware verifica su nombre y su tamaño para ver si es válido para ser cifrado.

```
  }
}
else
{
  if ( mw_chek_filename(find_file_data.cFileName) || find_file_data.nFileSizeLow <= 5 )
  {
    v16 = v21;                          // Saltar
  }
  else
  {
    large_file_flag = (mw_check_large_file_extension)(find_file_data.cFileName);
    mw_process_file(
      drive_find_path,
      find_file_data.cFileName,
      find_file_data.nFileSizeHigh,
      find_file_data.nFileSizeLow,
      sub_dir_path,
      large_file_flag);
    v16 = 234 * v7;
    v26 = v20;
```

Ilustración 25: comprobación de ficheros

Si el nombre o extensión se encuentra en la lista siguiente o si el fichero es más pequeño de 6, PLAY evita cifrarlo:

.exe, .dll, .lnk, .sys, readme.txt, bootmgr, .msi, .PLAY, ReadMe.txt

```
mw_decrypt_string(&unk_42B944, 0x16u, v10, v11, v12);// ReadMe.txt
v1 = wcscmp(file_name, v12);
if ( v1 )
  v1 = v1 < 0 ? -1 : 1;
if ( v1 )
{
  v2 = mw_check_encrypted_extension(file_name);// ".PLAY"
  if ( !v2 )
    return 0;
  v3 = &EXTENSION_TO_AVOID_LIST;              // .exe, .dll, .lnk, .sys,
                                              // readme.txt, bootmgr, .msi
  v4 = 0;
  while ( 1 )
  {
    if ( v3 )
    {
      v5 = *v3;
      v14 = *(v3 + 4);
      v6 = *(v3 + 10);
      v13 = v5;
      v15 = v6;
    }
    else
    {
      v14 = 0;
      v13 = 0i64;
      v15 = 0;
      *_errno() = 22;
      _invalid_parameter_noinfo();
```

Ilustración 26: comprobación del nombre y extensión de fichero

PLAY también realiza una verificación adicional para ver si la extensión del archivo es la de los archivos grandes típicos para determinar su tipo de cifrado más adelante. El archivo se clasifica como grande si su extensión está en la siguiente lista:

mdf, ndf, ldf, frm

Nota de rescate

El contenido de la nota de rescate se encuentra almacenado como una cadena de caracteres codificada en el binario de PLAY. Ésta contiene la cadena "PLAY" así como la dirección de email del actor detrás del cifrado con la que debe ponerse en contacto la víctima para negociar el rescate. La nota se escribe con el nombre de fichero "ReadMe.txt".

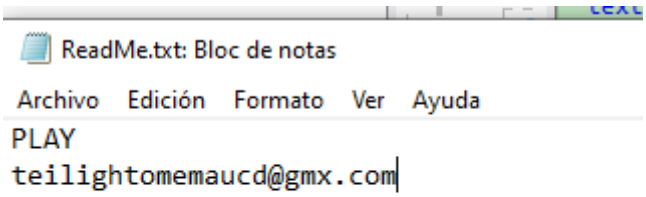


Ilustración 27: nota de rescate de la muestra de PLAY

Rellenando la estructura de ficheros

Para que cada archivo se cifre, PLAY primero completa la estructura del archivo con los datos apropiados sobre el archivo. Primero, comienza a recorrer la lista de estructuras de archivos globales para verificar si hay una estructura disponible para procesar el archivo.

```
if ( !*p_initialized_flag )
{
  file_struct = (v32 + 72 * v26[2]);
  file_struct->initialized_flags = 1;
  w_RtlLeaveCriticalSection(&CRITICAL_SECTION_1);
  return file_struct;
}
```

Ilustración 28: comprobación de estructuras disponibles en la lista de estructuras de archivos

Si no hay una estructura disponible en la lista global, PLAY llama a **Sleep** para que el hilo duerma y vuelve a verificar hasta que encuentra una. Una vez que se encuentra la estructura, el malware establece su campo `initialized_flag` en 1 y el campo de nombre de archivo en el nombre de archivo de destino. También rellena otros campos, como el tamaño del archivo, el indicador de archivo grande y el identificador del archivo.

```
file_struct_1 = mw_building_file_struct(folder_path, filename_1);
file_struct = file_struct_1;
file_struct_2 = file_struct_1;
if ( !file_struct_1 )
  return 0xFFFFFFFF;
wcscpy_s(file_struct_1->file_path, 0x7FFFu, folder_path);
wcscat_s(file_struct->file_path, 0x7FFFu, filename);
file_path = file_struct->file_path;
*(&file_struct->file_size.u.LowPart + 2) = a4;
*(&file_struct->file_size.QuadPart + 6) = a3;
*&file_struct->large_file_flag = a6;
file_struct->filename = filename_1;
FileAttributesW = w_GetFileAttributesW(file_path);
if ( FileAttributesW != INVALID_FILE_ATTRIBUTES && (FileAttributesW & FILE_ATTRIBUTE_READONLY) != 0 )
{
  new_file_attribute = (FileAttributesW ^ 1);
  file_name_1 = file_struct->file_path;
  SetFileAttributesW = mw_assign_resolved_api(::SetFileAttributesW);
  SetFileAttributesW(file_name_1, new_file_attribute);
  file_struct = file_struct_2;
}
```

Ilustración 29: rellenado de la estructura de ficheros a cifrar

Cifrado en subprocesos secundarios

Después de completar una estructura de archivos para un archivo específico, PLAY genera un hilo para comenzar a cifrar un archivo.

Si el archivo no se clasifica como un archivo grande, el malware calcula cuántos fragmentos necesita cifrar según el tamaño del archivo. El número de fragmentos cifrados es 2 si el tamaño del archivo es inferior o igual a 0x3fffffff bytes, 3 si el tamaño del archivo es inferior o igual a 0x27fffffff bytes y superior a 0x3fffffff bytes y 0 si el tamaño del archivo es igual a 0x280000000. Si el tamaño del archivo es superior a 0x280000000 bytes, la cantidad de fragmentos cifrados es 5.

```
int __thiscall mw_calculate_chunk_count(play_file_struct *this)
{
  DWORD lowPart; // edx
  LONG highPart; // esi

  lowPart = this->file_size.LowPart;
  highPart = this->file_size.HighPart;
  if ( (this->file_size.QuadPart - 0x5000001) <= 0x3AFFFFFE )
    return 2;
  if ( __PAIR64__(highPart, lowPart) - 0x40000001 <= 0x23FFFFFEi64 )
    return 3;
  if ( __SPAIR64__(highPart, lowPart) <= 0x280000000i64 )
    return 0;
  return 5;
}
```

Ilustración 30: cálculo de frafmentos a cifrar

El modo de cifrado predeterminado se establece a AES-GCM. Sin embargo, si el tamaño del archivo es mayor que 4025 veces su tamaño cifrado (que es el tamaño de fragmento 0x100000 multiplicado por el recuento de fragmentos), el modo de cifrado se establece en AES-CBC.

Esto se debe a que AES-GCM tiene peor rendimiento en comparación con AES-CBC[1]: AES-GCM es un cifrado más seguro que AES-CBC porque AES-CBC opera mediante XOR (OR eXclusivo) cada bloque con el bloque anterior y no se puede escribir en paralelo. Esto afecta el rendimiento debido a las matemáticas complejas involucradas que requieren el cifrado en serie.

Para el cifrado de archivos, PLAY ahora presenta una nueva estructura que representa el contenido del pie de página del archivo que se escribe en cada archivo cifrado.

```
struct file_footer_struct
{
  byte footer_marker_head[16];
  WORD last_chunk_size;
  WORD total_chunk_count;
  WORD large_file_flag;
  WORD small_file_flag;
  DWORD default_chunk_size;
  DWORD footer_marker_tail;
  QWORD encrypted_chunk_count;
  byte encrypted_symmetric_key[1024];
};
```

A continuación, se indica el significado de cada variable:

- `footer_marker_head`: primer índice del valor file_marker en la estructura de fichero
- `last_chunk_size`: tamaño del último fragmento al final del archivo
- `total_chunk_count`: número total de fragmentos que se cifrarán
- `large_file_flag`: inicializado a 1 si el archivo es más grande que 0x500000
- `small_file_flag`: inicializado a 1 cuando el tamaño del archivo alto es menor que 0
- `default_chunk_size`: 0x100000 bytes
- `footer_marker_tail`: xxHash32 hash de footer_marker_head. También el segundo índice en file_marker de file struct
- `encrypted_chunk_count`: número total de fragmentos cifrados con éxito
- `encrypted_symmetric_key`: BLOB con la clave AES cifrada por RSA

[1] https://helpdesk.privateinternetaccess.com/kb/articles/what-s-the-difference-between-aes-cbc-and-aes-gcm#:~:text=AES%2DGCM%20is%20a%20more,mathematics%20involved%20requiring%20serial%20encryption

Primero, PLAY lee 0x428 bytes al final del archivo para verificar el pie de página del archivo. Si el tamaño del archivo es inferior a 0x428 bytes, se garantiza que el archivo no está cifrado, por lo que el malware pasa a cifrarlo de inmediato.

Si los últimos 0x428 bytes se leen con éxito, el malware verifica si el *hash* xxHash32 de la cabeza del marcador de pie de página es igual a la cola del marcador de pie de página. Si lo son, se confirma que el pie de página del archivo es válido y el archivo ya está cifrado.

Si este no es el caso, PLAY verifica cada DWORD en el encabezado del marcador de pie de página y lo compara con los valores codificados en la estructura del archivo. Esto es para verificar si el pie de página del archivo no está cifrado, si el pie de página del archivo está escrito, pero no ha sido cifrado, o si el archivo ya está cifrado.

```
  v17 = 0;
  ReadFile = mw_assign_resolved_api(::ReadFile);
  v4 = ReadFile(file_handle, file_footer_buffer, 1064, &v17, 0);
  for ( i = 75; i < 0x4D; ++i )
    v13 *= 4185;
  if ( !v4 )
    return -1;
  footer_marker_tail = file_footer_buffer->footer_marker_tail;
  if ( w_xxhash32(file_footer_buffer) != footer_marker_tail )
    return 0;
  v7 = *file_struct_file_marker;
  if ( *file_struct_file_marker == *file_footer_buffer->footer_marker_head
    || (v8 = v7 < file_footer_buffer->footer_marker_head[0], v7 == file_footer_buffer->footer_marker_head[0])
    && (v9 = *(file_struct_file_marker + 1),
        v8 = v9 < file_footer_buffer->footer_marker_head[1],
        v9 == file_footer_buffer->footer_marker_head[1])
    && (v10 = *(file_struct_file_marker + 2),
        v8 = v10 < file_footer_buffer->footer_marker_head[2],
        v10 == file_footer_buffer->footer_marker_head[2])
    && (v11 = *(file_struct_file_marker + 3),
        v8 = v11 < file_footer_buffer->footer_marker_head[3],
        v11 == file_footer_buffer->footer_marker_head[3]) )
  {
    v12 = 0;                                    // 0 = No cifrado
  }
  else
  {
    v12 = v8 ? -1 : 1;                          // -1 = inválido
                                                // 1 = pie de fichero escrito pero no cifrado
  }
  return (v12 != 0) + 1;                        // 2 = Ya cifrado
}
```

Ilustración 31: comprobación del pie de fichero para comprobar su estado de cifrado

Cifrado de ficheros

Para cifrar un archivo desde cero, PLAY primero genera una clave AES para cifrar el archivo. Llama a **BCryptGenRandom** para generar un búfer aleatorio de 0x20 bytes. Según el modo de cifrado especificado en la estructura del archivo, el malware llama a **BCryptSetProperty** para configurar el cifrado correctamente con su manejador de proveedor AES.

A continuación, se llama a **BCryptGenerateSymmetricKey** en el búfer de 0x20 bytes generado aleatoriamente para generar el manejador de clave AES.

```
if ( chaining_mode_flag )
{
  mw_decrypt_string(&unk_42B8F4, 0x20u, qword_42D720, SHIDWORD(qword_42D720), bcrypt_property);// ChainingModeGCM
  v95 = 1277;
  mw_decrypt_string(&unk_42B914, 0x1Au, qword_42D720, SHIDWORD(qword_42D720), property_name_1);// ChainingMode
  v99 = (v87[0] + v87[9] + 9907);
  v115 = v87[0] + v87[9] + 9907;
  v96 = v127[0] + 9977;
  v23 = (w_BCryptSetProperty)(bcrypt_property, 64, 0);
  memset(property_name_1, 0, sizeof(property_name_1));
  v24 = v127[2];
  v127[2] = v87[6] - 3;
  v25 = v24 + 11;
  v87[6] -= 3;
  BYTE2(v93) = v25 + 22;
  HIBYTE(v93) = v127[1] + 3;
  memset(bcrypt_property, 0, sizeof(bcrypt_property));
  v89[2] += 2;
  if...
  v26 = v115;
  v27 = v87[3] + 9907;
  v127[0] = v88[7];
LABEL_26:
  v99 = v27;
  v29 = 3710;
  if...
  HIBYTE(v115) = ++v109[7];
  if ( w_BCryptGenerateSymmetricKey(0, 0, Src) )
  {
    ++v89[3];
    ++v89[7];
    v22 = -4;
  }
```

Ilustración 32: generación del manejador de clave AES

A continuación, para almacenar la clave AES en la estructura del pie de página del archivo, PLAY llama a **BCryptExportKey** para exportar la clave AES a un blob de claves de 0x230 bytes. También llama a **BCryptGenRandom** para generar aleatoriamente un IV de 0x10 bytes y lo agrega después del blob de la clave.

Ilustración 33: exportación del blob de clave AES y el IV

Luego, llama a **BCryptEncrypt** para cifrar el blob de clave exportado y el IV utilizando el manejador de la clave pública RSA y escribe la salida cifrada en un búfer de 0x400 bytes. A continuación, este búfer se copia en el campo de clave encrypted_symmetric_key de la estructura del pie de página del archivo.

Ilustración 34: cifrado del blob de clave AES con la clave pública RSA

PLAY luego completa los otros campos del pie de página del archivo, como **`footer_marker_head`**, **`footer_marker_tail`**, **`small_file_flag`** y **`large_file_flag`**

con la información existente de la estructura del archivo. El tamaño de fragmento predeterminado también se establece en 0x100000 bytes.

```
small_file_flag = a1->file_size.HighPart < 0;
v19 = a1->file_size.HighPart <= 0;
LOBYTE(v81[4]) = v14;
WORD1(v54) = v17;
HIDWORD(liDistanceToMove) = liDistanceToMove - 1;
v105 = 0x74;
if ( small_file_flag || v19 && a1->file_size.LowPart <= 0x500000 )
{
  v13->small_file_flag = 1;                    // file_type -> small
  v89 = BYTE2(v98) + 103;
  v79 = (v82[4] + v103);
  HIWORD(v65) = v82[4] + v103;
  LODWORD(liDistanceToMove) = v13->large_file_flag;
  v106 = v100[1];
}
else
{
  ++HIBYTE(v81[0]);
  v13->large_file_flag = 1;                    // file_type -> large
  v99 = 3834;
  v106 = -89;
  v100[1] = -89;
  LODWORD(liDistanceToMove) = 1;
  v89 = BYTE1(v98) + 2951;
}
*(&v81[4] + 2) = 16i64;
v13->default_chunk_size = 0x100000;
*(&v81[2] + 2) = a1->file_size.QuadPart;
v48 = v49 + 1;
```

Ilustración 35: rellenado de la estructura del pie de fichero cifrado

Una vez que el pie de página del archivo está completamente lleno, el malware llama a **SetFilePointerEx** para mover el puntero del archivo al final del archivo y llama a **WriteFile** para escribir la estructura allí.

```
movzx   eax, al
mov     ecx, eax
mov     [ebp+var_4C], eax
imul    ecx, dword ptr [ebp+var_144]
mov     eax, [ebp+var_7C]
imul    edx, eax
sub     al, [ebp+var_138+0Ah]
inc     byte ptr [ebp+var_64+2]
add     al, cl
mov     dword ptr [ebp+var_144], ecx
mov     [ebp+var_138+2], al
movzx   eax, dx
add     eax, [ebp+var_4C]
add     eax, dword ptr [ebp+var_18C+2]
mov     word ptr [ebp+file_footer_1+2], ax
mov     al, cl
sub     al, byte ptr [ebp+var_64+3]
add     al, byte ptr [ebp+var_64]
mov     [ebp+var_60], al
lea     eax, [ebp+file_handle_1]
push    0
push    eax
mov     word ptr [ebp+file_footer_1], cx
mov     ecx, [esi+34h]
mov     dword ptr [ebp+liDistanceToMove+4], edx
mov     edx, [ebp+var_88]
push    428h
call    w_WriteFile
```

```
  v85->total_chunk_count = a5;
}
v84[1] = (v46 + v80);
v53[0] = v99 + 2;
if ( !w_SetFilePointerEx(2, SHIDWORD(liDistanceToMove), SHIDWORD(v21)) )// Situar puntero al final del fichero
{
  ++HIBYTE(v81[0]);
  v12 = -2;
  v100[1] = v106 + 1;
  goto LABEL_51;
}
v99 = HIBYTE(v81[4]);
++BYTE2(v91);
v51 = (v48 >> 1) * HIBYTE(v81[4]);
v53[2] = v51 + v88 - v53[10];
v92 = v91 + v51 - HIBYTE(v91);
*(&v81[5] + 2) = 0;
HIDWORD(liDistanceToMove) = v88 * v56;
if ( !w_WriteFile(1064, &file_handle_1) )
{
  LOWORD(v62) = v10 + v62;
  v12 = -3;
  ++v100[3];
  HIWORD(v57) = v66 - v57;
  goto LABEL_51;
}
BYTE1(v81[6]) = v48 % v80;
v22 = v87 + 1;
```

Ilustración 36: escritura del pie en el fichero cifrado

Si el tamaño del archivo es superior a 0x500000 bytes, PLAY solo cifra la primera y la última parte del archivo.

```
339     v74 = v87 + v54 + 87;
340   }
341   HighPart = file_struct_1->file_size.HighPart;
342   LowPart = file_struct_1->file_size.LowPart;
343   WORD1(v54) = v17;
344   v99 = v17;
345   if ( __SPAIR64__(HighPart, LowPart) > 0x500000 )// Tamaño mayor a 0x500000 = cifrar primer y último chunk
346   {
347     v31 = mw_encrypt_file(
348             bcrypt_sym_key_handle,
349             &crypt_IV,
350             file_struct_1->file_handle,
351             file_struct_1->file_data_buffer,
352             0x100000,
353             0,
354             &chunk_count_flag,
355             &chunk_write_offset_from_end,
356             0,
357             0);
358     v32 = HIWORD(v57) - 4760;
359     HIWORD(v57) -= 4760;
360     LOWORD(v62) = v62 + 182;
361     if ( v31 <= 0 )
362     {
363       v12 = -6;
364       LOBYTE(v81[2]) = 77 * v106;
365       LOWORD(v65) = v103 - v100[4];
366       goto LABEL_51;
367     }
```

Ilustración 37: cifrado del primer y último fragmento de archivos grandes

La función de cifrado consta de una llamada **ReadFile** para leer los fragmentos de datos en el búfer en la estructura del archivo, una llamada **BCryptEncrypt** para cifrar el archivo usando el identificador de clave AES y el IV generado. Una vez que finaliza el cifrado, el malware llama a **WriteFile** para escribir la salida cifrada en el archivo, así como el índice del fragmento que se cifra en el pie de página del archivo. Esto se usa potencialmente para realizar un seguimiento de cuántos fragmentos se han cifrado en el caso de que se produzcan daños o interrupciones.

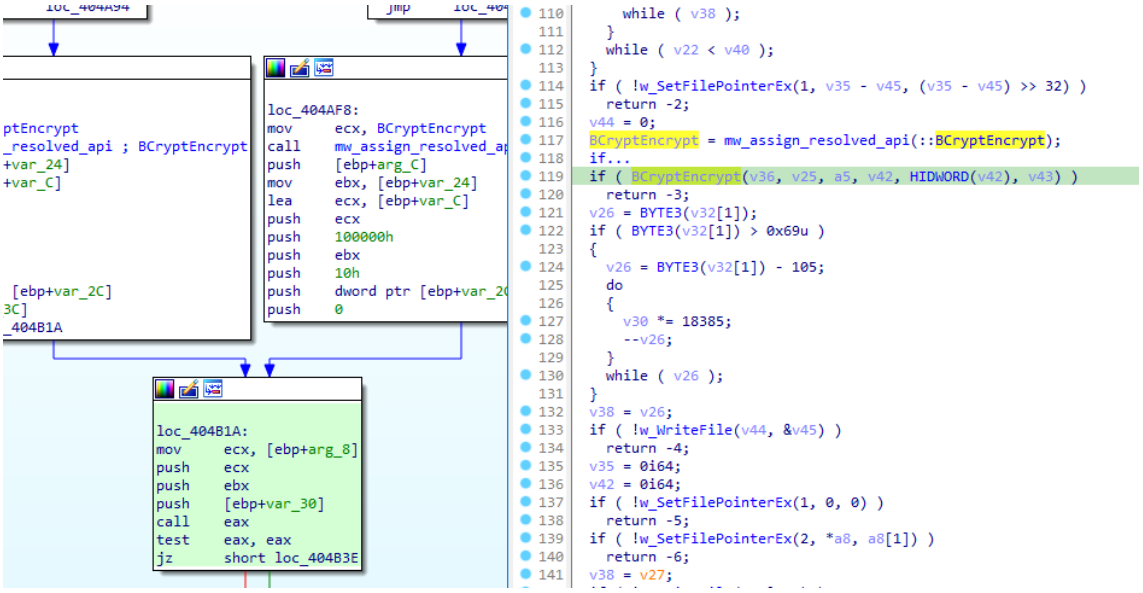


Ilustración 38: función de cifrado de la información

Si el tamaño del archivo es más pequeño que el tamaño de fragmento predeterminado de 0x100000 bytes, el malware cifra todo el archivo.

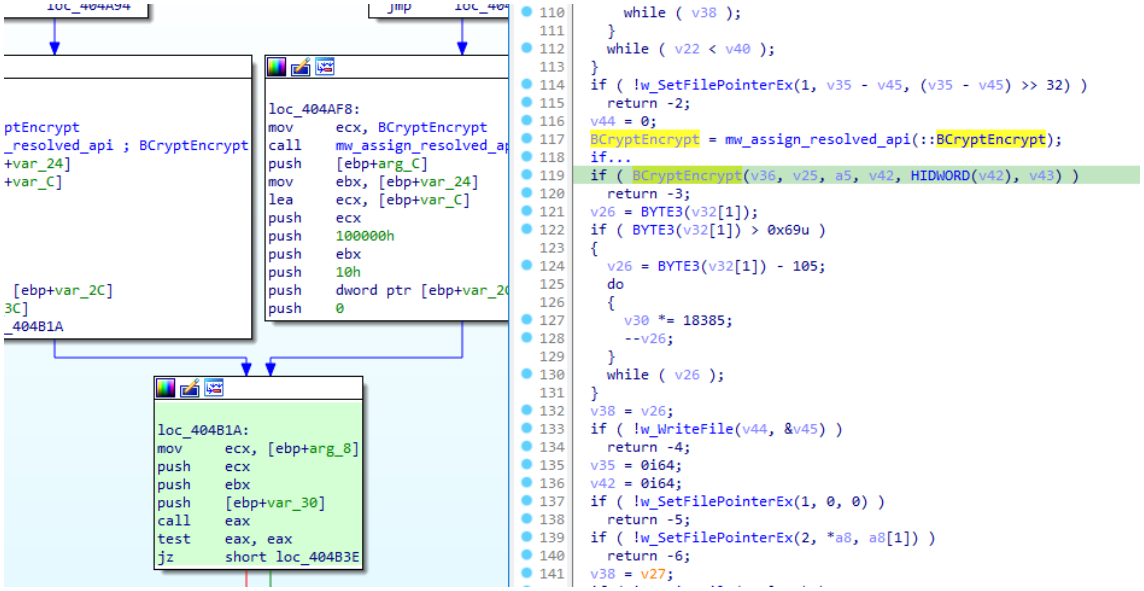


Ilustración 39: cifrado completo de archivos pequeños

Si el tamaño del archivo está entre 0x100000 y 0x500000, el malware lo cifra en fragmentos de 0x100000 bytes hasta que llega al final del archivo.

```
498  v99 = 0;
499  while ( 1 )
500  {
501    v27 = mw_encrypt_file(
502            v107,
503            &v112,
504            file_struct_1->file_handle,
505            file_struct_1->file_data_buffer,
506            0x100000,                          // cifrar chunks de 0x100000 bytes
507            0,
508            &chunk_count_flag,
509            &chunk_write_offset_from_end,
510            0,
511            0);
512    v28 = v103 - 1;
513    v103 = v28;
514    LOBYTE(v98) = v28;
515    BYTE1(v81[0]) = v105;
516    if ( v27 <= 0 )
517      break;
518    v90[12] = v89 * v106;
519    WORD1(v62) = v100[4];
```

Ilustración 40: cifrado de archivos de tamaño medio

Finalmente, después de cifrar el archivo, el malware cambia su extensión a ".PLAY" llamando a **MoveFileW**.

```
72    v11 = 2 * v7 + 13;
73    encrypted_filename = w_VirtualAlloc(v7, 4);
74    if ( !encrypted_filename )
75      return -1;
76    wcscpy_s(encrypted_filename, v11, file_path_1);
77    wcscat_s(encrypted_filename, v11, encrypted_extension);// ".PLAY"
78    *encrypted_extension = 0i64;
79    LODWORD(v24) = 0;
80    MoveFileW = mw_assign_resolved_api(::MoveFileW);
81    if ( !MoveFileW(file_path_1, encrypted_filename) )
82      return -2;
83    LODWORD(v24) = v15;
84    w_VirtualFree();
85    if ( v19 > 0x67Du )
86    {
87      v16 = 73;
88      v17 = v19 - 1661;
```

Ilustración 41: cambio de extensión a ".PLAY"

No obstante, hay un pequeño error en el código que hace que siempre cambie la extensión de un archivo a pesar de si el cifrado es exitoso o no debido al valor de retorno de la función de cifrado de archivos.

```
20    v2 /= chunk_count;
21  if ( v2 > 0xFB9 )
22    file_struct->chaining_mode_flag = 0;
23  encrypt_result = (mw_w_encrypt_file)(file_struct);
24  file_handle = file_struct->file_handle;
25  CloseHandle = mw_assign_resolved_api(CloseHandle_0);
26  CloseHandle(file_handle);
27  if ( encrypt_result )                    // Fallo: encrypt_result nunca vale 0
28    mw_set_encrypted_extension(file_struct->file_path);
29  w_RtlEnterCriticalSection(&CRITICAL_SECTION_1);
30  file_struct->initialized_flags = 0;
31  w_RtlLeaveCriticalSection(&CRITICAL_SECTION_1);
32  return encrypt_result;
33 }
```

Ilustración 42: fallo en el código de cambio de extensión

MITRE ATT&CK			
Initial Access	T1133	External Remote Services	**M1032: Multi-factor Authentication** Use strong two-factor or multi-factor authentication for remote service accounts to mitigate an adversary's ability to leverage stolen credentials, but be aware of [Multi-Factor Authentication Interception](https://attack.mitre.org/techniques/T1111) techniques for some two-factor authentication implementations.
			M1042: Disable or Remove Feature or Program Disable or block remotely available services that may be unnecessary.
			M1030: Network Segmentation Deny direct remote access to internal systems through the use of network proxies, gateways, and firewalls.

			M1035: Limit Access to Resource Over Network Limit access to remote services through centrally managed concentrators such as VPNs and other managed remote access systems.
	T1078	Valid Accounts	**M1027: Password Policies** Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. (Citation: US-CERT Alert TA13-175A Risks of Default Passwords on the Internet) When possible, applications that use SSH keys should be updated periodically and properly secured.
			M1017: User Training Applications may send push notifications to verify a login as a form of multi-factor authentication (MFA). Train users to only accept valid push notifications and to report suspicious push notifications.
			M1013: Application Developer Guidance Ensure that applications do not store sensitive data or credentials insecurely. (e.g. plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage).
			M1018: User Account Management Regularly audit user accounts for activity and deactivate or remove any that are no longer needed.

			M1026: Privileged Account Management Audit domain and local accounts as well as their permission levels routinely to look for situations that could allow an adversary to gain wide access by obtaining credentials of a privileged account. (Citation: TechNet Credential Theft) (Citation: TechNet Least Privilege) These audits should also include if default accounts have been enabled, or if new local accounts are created that have not be authorized. Follow best practices for design and administration of an enterprise network to limit privileged account use across administrative tiers. (Citation: Microsoft Securing Privileged Access)
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.
	T1204	User Execution	**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)
		M1038: Execution Prevention Application control may be able to prevent the running of executables masquerading as other files.
		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.

Persistence	T1133	External Remote Services	**M1032: Multi-factor Authentication** Use strong two-factor or multi-factor authentication for remote service accounts to mitigate an adversary's ability to leverage stolen credentials, but be aware of [Multi-Factor Authentication Interception](https://attack.mitre.org/techniques/T1111) techniques for some two-factor authentication implementations.
			M1042: Disable or Remove Feature or Program Disable or block remotely available services that may be unnecessary.
			M1030: Network Segmentation Deny direct remote access to internal systems through the use of network proxies, gateways, and firewalls.
			M1035: Limit Access to Resource Over Network Limit access to remote services through centrally managed concentrators such as VPNs and other managed remote access systems.
	T1078	Valid Accounts	**M1027: Password Policies** Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. (Citation: US-CERT Alert TA13-175A Risks of Default Passwords on the Internet) When possible, applications that use SSH keys should be updated periodically and properly secured.

			M1017: User Training Applications may send push notifications to verify a login as a form of multi-factor authentication (MFA). Train users to only accept valid push notifications and to report suspicious push notifications.
			M1013: Application Developer Guidance Ensure that applications do not store sensitive data or credentials insecurely. (e.g. plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage).
			M1018: User Account Management Regularly audit user accounts for activity and deactivate or remove any that are no longer needed.
			M1026: Privileged Account Management Audit domain and local accounts as well as their permission levels routinely to look for situations that could allow an adversary to gain wide access by obtaining credentials of a privileged account. (Citation: TechNet Credential Theft) (Citation: TechNet Least Privilege) These audits should also include if default accounts have been enabled, or if new local accounts are created that have not be authorized. Follow best practices for design and administration of an enterprise network to limit privileged account use across administrative tiers. (Citation: Microsoft Securing Privileged Access)
			M1027: Password Policies Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. (Citation: US-CERT Alert TA13-175A Risks of Default Passwords on the Internet) When possible, applications that use SSH keys should be updated periodically and properly secured.

Privilege Escalation	T1078	Valid Accounts	**M1017: User Training** Applications may send push notifications to verify a login as a form of multi-factor authentication (MFA). Train users to only accept valid push notifications and to report suspicious push notifications.
			M1013: Application Developer Guidance Ensure that applications do not store sensitive data or credentials insecurely. (e.g. plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage).
			M1018: User Account Management Regularly audit user accounts for activity and deactivate or remove any that are no longer needed.
			M1026: Privileged Account Management Audit domain and local accounts as well as their permission levels routinely to look for situations that could allow an adversary to gain wide access by obtaining credentials of a privileged account. (Citation: TechNet Credential Theft) (Citation: TechNet Least Privilege) These audits should also include if default accounts have been enabled, or if new local accounts are created that have not be authorized. Follow best practices for design and administration of an enterprise network to limit privileged account use across administrative tiers. (Citation: Microsoft Securing Privileged Access)
	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.**

Defense Evasion	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.**
	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.**
			M1027: Password Policies Applications and appliances that utilize default username and password should be changed immediately after the installation, and before deployment to a production environment. (Citation: US-CERT Alert TA13-175A Risks of Default Passwords on the

	T1078	Valid Accounts	Internet) When possible, applications that use SSH keys should be updated periodically and properly secured.
			M1017: User Training Applications may send push notifications to verify a login as a form of multi-factor authentication (MFA). Train users to only accept valid push notifications and to report suspicious push notifications.
			M1013: Application Developer Guidance Ensure that applications do not store sensitive data or credentials insecurely. (e.g. plaintext credentials in code, published credentials in repositories, or credentials in public cloud storage).
			M1018: User Account Management Regularly audit user accounts for activity and deactivate or remove any that are no longer needed.
			M1026: Privileged Account Management Audit domain and local accounts as well as their permission levels routinely to look for situations that could allow an adversary to gain wide access by obtaining credentials of a privileged account. (Citation: TechNet Credential Theft) (Citation: TechNet Least Privilege) These audits should also include if default accounts have been enabled, or if new local accounts are created that have not be authorized. Follow best practices for design and administration of an enterprise network to limit privileged account use across administrative tiers. (Citation: Microsoft Securing Privileged Access)

Discovery	T1135	Network Share Discovery	**M1028: Operating System Configuration** Enable Windows Group Policy “Do Not Allow Anonymous Enumeration of SAM Accounts and Shares” security setting to limit users who can enumerate network shares.(Citation: Windows Anonymous Enumeration of SAM Accounts)
	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.**
Impact	T1486	Data Encrypted for Impact	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable cloud-delivered protection and Attack Surface Reduction (ASR) rules to block the execution of files that resemble ransomware. (Citation: win10_asr)
			M1053: Data Backup Consider implementing IT disaster recovery plans that contain procedures for regularly taking and testing data backups that can be used to restore organizational data.(Citation: Ready.gov IT DRP) Ensure backups are stored off system and is protected from common methods adversaries may use to gain access and destroy the backups to prevent recovery. Consider enabling versioning in cloud environments to maintain backup copies of storage objects.(Citation: Rhino S3 Ransomware Part 2)

Mitigación

Medidas a nivel de endpoint

El código de *PLAY* 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.

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

- 006ae41910887f0811a3ba2868ef9576bbd265216554850112319af878f06e55
- 0b4f893402b51880d0df703c21195aa0f50cbdd3a5b223cc333cd24f0167c16d
- 0ed328af77f2576071bfd543938fc01101daac01f216dc43bc091a8da4aff18d
- 176476f9d924d83343a51a90ade097d12b7594dc5dbca1771c440047dfbe81eb
- 2ab190542c3ec7b2b6e6d4bccce4c5d6a572f98c6bc89b014fea0c8fd6db6723
- 2e9126dfad03bdaf54f9b29ade42038c83f65ac7288376f45768901660f62d7b
- 3e6317229d122073f57264d6f69ae3e145decad3666ddad8173c942e80588e69
- 47c7cee3d76106279c4c28ad1de3c833c1ba0a2ec56b0150586c7e8480ccae57
- 608e2b023dc8f7e02ae2000fc7dbfc24e47807d1e4264cbd6bb5839c81f91934
- 73e19be4da76bb4e52cb82493c75690977fc3a5f589a9b47e834362545ef512a
- 8962de34e5d63228d5ab037c87262e5b13bb9c17e73e5db7d6be4212d66f1c22
- 957a6aee2437a5c4d31372af2f6bceb29e1c7a49d650fe207cefc624bf6bca82
- bb51255ec929ae1fb34981b8b988769027ee49e68c0958a4a2a76b59a0dc1cff
- cd16f6b2faaf801a0de5d5afc68ed50a5bde9b4905bb00fe5f408b265ec081e5
- e641b622b1f180fe189e3f39b3466b16ca5040b5a1869e5d30c92cca5727d3f0
- f054f373cead893f868fd9b4acc24f751afefbb80cf961e305f97741f952a641

Yara:

- Esta regla sirve para identificar las muestras de la familia *PLAY*, se trata de una yara procedente del analista Felix Bilstein:

YARA

```
rule win_play_auto {

    meta:
        author = "Felix Bilstein - yara-signator at cocacoding dot com"
        date = "2022-11-21"
        version = "1"
```

```
        description = "Detects win.play."
        info = "autogenerated rule brought to you by yara-signator"
        tool = "yara-signator v0.6.0"
        signator_config = "callsandjumps;datarefs;binvalue"
        malpedia_reference =
"https://malpedia.caad.fkie.fraunhofer.de/details/win.play"
        malpedia_rule_date = "20221118"
        malpedia_hash = "e0702e2e6d1d00da65c8a29a4ebacd0a4c59e1af"
        malpedia_version = "20221125"
        malpedia_license = "CC BY-SA 4.0"
        malpedia_sharing = "TLP:WHITE"

    /* DISCLAIMER
     * The strings used in this rule have been automatically selected
from the
     * disassembly of memory dumps and unpacked files, using YARA-
Signator.
     * The code and documentation is published here:
     * https://github.com/fxb-cocacoding/yara-signator
     * As Malpedia is used as data source, please note that for a given
     * number of families, only single samples are documented.
     * This likely impacts the degree of generalization these rules
will offer.
     * Take the described generation method also into consideration
when you
     * apply the rules in your use cases and assign them confidence
levels.
     */


    strings:
        $sequence_0 = { 8885affdffff 8a852cfdffff 8885b0fdffff
8a8529fdffff 8885aefdffff 668b8534fdffff 668985a2fdffff }
            // n = 7, score = 100
            //   8885affdffff        | mov                 byte ptr
[ebp - 0x251], al
            //   8a852cfdffff        | mov                 al, byte
ptr [ebp - 0x2d4]
            //   8885b0fdffff        | mov                 byte ptr
[ebp - 0x250], al
            //   8a8529fdffff        | mov                 al, byte
ptr [ebp - 0x2d7]
            //   8885aefdffff        | mov                 byte ptr
[ebp - 0x252], al
            //   668b8534fdffff      | mov                 ax, word
ptr [ebp - 0x2cc]
```

```
            //   668985a2fdffff       | mov                 word ptr
[ebp - 0x25e], ax

        $sequence_1 = { ff15???????? 83f809 7513 0fb645e9 0527130000 50
ff15???????? }
            // n = 7, score = 100
            //   ff15????????         |
            //   83f809               | cmp                 eax, 9
            //   7513                 | jne                 0x15
            //   0fb645e9             | movzx               eax, byte
ptr [ebp - 0x17]
            //   0527130000           | add                 eax, 0x1327
            //   50                   | push                eax
            //   ff15????????         |

        $sequence_2 = { 90 40 02ca 3dd5040000 72f6 8b5d08 }
            // n = 6, score = 100
            //   90                   | nop
            //   40                   | inc                 eax
            //   02ca                 | add                 cl, dl
            //   3dd5040000           | cmp                 eax, 0x4d5
            //   72f6                 | jb                  0xfffffff8
            //   8b5d08               | mov                 ebx, dword
ptr [ebp + 8]

        $sequence_3 = { 0fb6cd 83f968 7612 8a55da }
            // n = 4, score = 100
            //   0fb6cd               | movzx               ecx, ch
            //   83f968               | cmp                 ecx, 0x68
            //   7612                 | jbe                 0x14
            //   8a55da               | mov                 dl, byte
ptr [ebp - 0x26]

        $sequence_4 = { 3bc2 7f06 81c4c5010000 83c40c e8???????? 9e }
            // n = 6, score = 100
            //   3bc2                 | cmp                 eax, edx
            //   7f06                 | jg                  8
            //   81c4c5010000         | add                 esp, 0x1c5
            //   83c40c               | add                 esp, 0xc
            //   e8????????           |
            //   9e                   | sahf

        $sequence_5 = { 0fb685d2feffff 89856cfeffff 03d8 e8????????
8a85cbfeffff b9???????? fec0 }
            // n = 7, score = 100
```

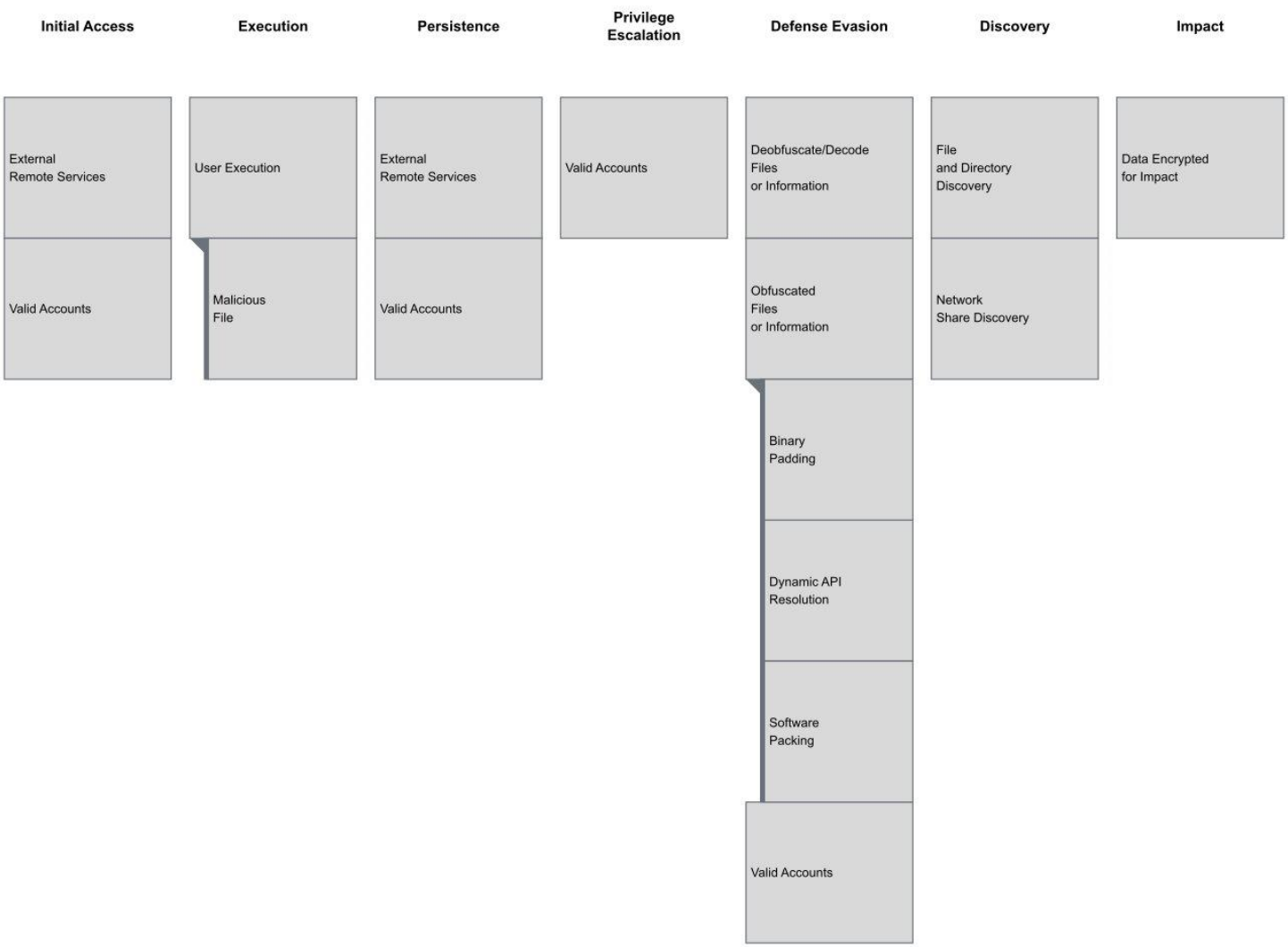
```
            //   0fb685d2feffff       | movzx               eax, byte
ptr [ebp - 0x12e]
            //   89856cfeffff         | mov                 dword ptr
[ebp - 0x194], eax
            //   03d8                 | add                 ebx, eax
            //   e8????????           |
            //   8a85cbfeffff         | mov                 al, byte
ptr [ebp - 0x135]
            //   b9????????           |
            //   fec0                 | inc                 al

        $sequence_6 = { 8955e8 0fb6543104 8b4df4 c1e208 0fb64c3104 0bd1
8b4df0 }
            // n = 7, score = 100
            //   8955e8               | mov                 dword ptr
[ebp - 0x18], edx
            //   0fb6543104           | movzx               edx, byte
ptr [ecx + esi + 4]
            //   8b4df4               | mov                 ecx, dword
ptr [ebp - 0xc]
            //   c1e208               | shl                 edx, 8
            //   0fb64c3104           | movzx               ecx, byte
ptr [ecx + esi + 4]
            //   0bd1                 | or                  edx, ecx
            //   8b4df0               | mov                 ecx, dword
ptr [ebp - 0x10]

        $sequence_7 = { 83042436 c3 aa 0030 85b7a34a892f 59 6d }
            // n = 7, score = 100
            //   83042436             | add                 dword ptr
[esp], 0x36
            //   c3                   | ret
            //   aa                   | stosb               byte ptr
es:[edi], al
            //   0030                 | add                 byte ptr
[eax], dh
            //   85b7a34a892f         | test                dword ptr
[edi + 0x2f894aa3], esi
            //   59                   | pop                 ecx
            //   6d                   | insd                dword ptr
es:[edi], dx

        $sequence_8 = { ce e0fd 9c 43 5b f5 ae }
            // n = 7, score = 100
            //   ce                   | into
            //   e0fd                 | loopne              0xffffffff
```

```
            //   9c                  | pushfd
            //   43                  | inc                 ebx
            //   5b                  | pop                 ebx
            //   f5                  | cmc
            //   ae                  | scasb               al, byte
ptr es:[edi]

        $sequence_9 = { 0f854c010000 b890000000 8b0c02 85c9
0f8430010000 837c020400 0f8425010000 }
            // n = 7, score = 100
            //   0f854c010000        | jne                 0x152
            //   b890000000          | mov                 eax, 0x90
            //   8b0c02              | mov                 ecx, dword
ptr [edx + eax]
            //   85c9                | test                ecx, ecx
            //   0f8430010000        | je                  0x136
            //   837c020400          | cmp                 dword ptr
[edx + eax + 4], 0
            //   0f8425010000        | je                  0x12b

    condition:
        7 of them and filesize < 389120
}
```

Referencias adicionales

- https://malpedia.caad.fkie.fraunhofer.de/details/win.play
- https://chuongdong.com/reverse%20engineering/2022/09/03/PLAYRansomware/
- https://www.trendmicro.com/en_us/research/22/i/play-ransomware-s-attack-playbook-unmasks-it-as-another-hive-aff.html
- https://www.sentinelone.com/labs/crimeware-trends-ransomware-developers-turn-to-intermittent-encryption-to-evade-detection/
- https://www.fortinet.com/blog/threat-research/ransomware-roundup-play-ransomware
- https://www.orangecyberdefense.com/global/blog/playing-the-game
- https://www.avertium.com/resources/threat-reports/an-in-depth-look-at-play-ransomware
- https://www.bleepingcomputer.com/news/security/rackspace-confirms-play-ransomware-was-behind-recent-cyberattack/

Apéndice A: Mapa de técnicas de ATT&CK



EUSKO JAURLARITZA GOBIERNO VASCO
GRUPO spri TALDEA