Universidad de Buenos Aires
Facultades de Ciencias Económicas,
Cs. Exactas y Naturales e Ingeniería

Carrera de Especialización en Seguridad Informática

Trabajo Final Integrador

Tema
Ransomware

Título
Análisis de Cryptowall V3

Autor: Ing. Geovanny Soria Alava
Tutor del Trabajo Final: Mg. Juan Devincenzi

Año de Presentación
2018

Cohorte del Cursante
2015

DECLARACIÓN JURADA

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e Internacional de Propiedad Intelectual.

RESUMEN

Cryptowall es un tipo de ransomware que tiene por objetivo cifrar archivos en la computadora de la víctima, para luego exigir un pago a cambio del acceso a una clave o programa que le permita al usuario restablecer su información. Este tipo de malware es la amenaza más importante de la época actual, causante de millonarias pérdidas a usuarios domésticos y corporativos a nivel mundial.

Este trabajo final de especialización describe inicialmente el funcionamiento de dos conceptos asociados al éxito del ransomware: la red TOR y la moneda digital Bitcoin, responsables de la anonimidad de los servidores involucrados y la facilidad para el manejo de un sistema de pago internacional y anónimo.

Adicionalmente, se realiza el análisis de una muestra del ransomware Cryptowall versión tres, ejecutada en un ambiente aislado junto con la implementación de un falso servidor de Comando y Control que suplante las instrucciones requeridas por el malware.

Palabras clave: Cryptowall, Ransomware, malware, onion routing, TOR, Bitcoin, red anónima, análisis.

ÍNDICE

AGRADECIMIENTO

El autor de este documento expresa su más sincero agradecimiento a:

Mg. Juan Alejandro Devincenzi, Docente y tutor de la Especialización en Seguridad Informática, Facultad de Ciencias Económicas, Universidad de Buenos Aires, cuyos aportes fueron fundamentales para la elaboración del presente trabajo.

PhD. Juan Pedro Hecht, Docente y Coordinador Académico de la Maestría en Seguridad Informática, Universidad de Buenos Aires, por todas sus enseñanzas impartidas durante el posgrado.

Ing. Segundo Soria, Ing. Jenny Alava e Ing. Michelle Soria, por su valiosa ayuda y contaste estímulo a finalizar este proyecto.

NÓMINA DE ABREVIATURAS

TOR	The Onion Router
ISP	Internet Service Provider
I2P	Invisible Internet Project
P2P	Peer to peer
CW3	Cryptowall V3
C&C	Command and Control Server
C2	Command and Control Server
OP	Onion Proxy
OR	Onion Router
DS	Directory Service
HTTP	Hipertext Transfer Protocol
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
SSH	Secure Shell
TLS	Transport Layer Security
IAT	Import Address Table
TCP	Transmission Control Protocol
PEB	Process Environment Block
MD5	Message Digest Algorithm
IP	Internet Protocol

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Hoy en día, el internet es una herramienta indispensable para toda clase de usuario, millones de personas la utilizan como fuente de entretenimiento, comunicación, aprendizaje y comercio electrónico. Por consiguiente, desde hace mucho dejó de ser una tecnología reservada sólo para usuarios técnicos y se convirtió en una herramienta accesible para todo tipo de usuario, sin importar su edad o nivel de conocimiento.

La normalización del uso de una tecnología, además de ventajas, trae consigo una serie de dificultades. Por lo general, estas surgen por el crecimiento exponencial de usuarios que se encuentran expuestos a cualquier tipo de engaño, ya sea por falta de conocimiento o interés. Esta situación es aprovechada por los autores de *malware,* que, con el tiempo, enfocaron sus esfuerzos hacia el desarrollo de código malicioso que les permita obtener réditos económicos, tales como el *spyware, adware y ransomware.*

Durante los últimos años, el *ransomware* se ha convertido en uno de los principales problemas de la seguridad de la información. Estadísticas recientes reportadas por *Kaspersky Security Network* [1] muestran un incremento notable en el número de usuarios infectados alrededor del mundo, que incluso suele ser mucho menor que la cifra real.

En su definición más básica, un *ransomware* es un tipo de *malware* que tiene por objetivo cifrar o bloquear ciertos archivos en la computadora de la víctima, para luego proceder a exigir un pago a cambio del acceso a un programa o clave que le permita al usuario recuperar sus archivos.

En las primeras versiones de *ransomware,* el *malware* creaba ventanas que cubrían la totalidad de la pantalla, impidiendo el normal funcionamiento del sistema operativo. Estas ventanas contenían un comunicado que exigía el envío de un mensaje SMS *premium* a cambio del restablecimiento del sistema operativo. Para frenar la propagación de este

[1] Es un servicio que proporciona acceso a la base de conocimientos de *Kaspersky Lab* donde se encuentra información acerca de la reputación de los archivos, recursos de Internet y software.

malware, las autoridades respectivas junto con expertos en seguridad optaron por bloquear el sistema de pago provisto por las operadoras de telefonía móvil. Entonces, esta forma de extorsión pasó a ser cada vez menos rentable.

Con el tiempo, el *ransomware* evolucionó y se incorporaron nuevas tecnologías a la idea inicial. En lugar de bloquear el sistema operativo, el *malware* cifra los archivos de la víctima y el pago se realiza a través de transferencias *bitcoins.* La elección del *bitcoin* como medio de pago concede a los ciberdelincuentes la ventaja de poder recibir transferencias de forma parcialmente anónima. Adicionalmente, para potenciar aún más la anonimidad del ciberdelincuente y de cualquier servidor involucrado, la mayoría de familias de *ransomware* utilizan la red TOR para ocultar la dirección IP de origen.

Este documento se encuentra estructurado en tres partes. La primera, que abarca los capítulos del dos al cinco, explica detalladamente el funcionamiento de la criptomoneda *Bitcoin* y la arquitectura de la red TOR. En la segunda parte, que comprende los capítulos seis y siete, se analiza una muestra específica del *ransomware Cryptowall* 3, para lo cual, se efectúa la captura y descifrado de paquetes que viajan entre el *malware* y el servidor *Command and Control (C&C).* Las capturas de pantalla y los datos obtenidos son el resultado de la ejecución del *malware* en un entorno controlado junto con la implementación de un falso servidor C&C que suplante la comunicación entre cliente y servidor. Finalmente, en el capítulo ocho se establecen las conclusiones obtenidas a lo largo del desarrollo de este trabajo.

2. REDES ANÓNIMAS [1]

En la actualidad, donde la comunicación a través de internet es cada vez más frecuente, resulta indispensable encontrar un medio que permita comunicarnos de manera anónima y confidencial. Se entiende por confidencial a toda información que sea accesible únicamente por personas

autorizadas y se logra a través del uso de cifrado simétrico[2] o asimétrico[3]. Por el contrario, una comunicación anónima resulta un aspecto más complejo de garantizar, sobre todo porque continuamente estamos expuestos al rastreo de información por parte de empresas de publicidad o incluso por nuestro propio ISP[4], hechos que afectan la privacidad de la información.

La privacidad de la información no solo consiste en asegurar la confidencialidad del mensaje, sino también, debe garantizar la anonimidad del origen y el destino de la comunicación, dado que, si son interceptados podrían poner en riesgo la identidad y ubicación de los usuarios. Por lo tanto, para evitarlo, se requiere que la información viaje por la red de manera anónima y segura.

Como respuesta a este inconveniente surgió lo que actualmente se conoce como redes anónimas. Básicamente, es un tipo de red que redirige el tráfico de internet entre varios *proxies*[5] con el fin de esconder el origen y el destino del mensaje. La principal diferencia entre las redes anónimas y las tradicionales es el método de ruteo de los paquetes de red en sus respectivas arquitecturas.

Si bien existen suficientes razones para promover la utilización de redes anónimas, estas han sido desplazadas por el uso malintencionado que los ciberdelincuentes le han otorgado, como, por ejemplo, para compartir contenido ilegal entre usuarios o para realizar ataques que involucren una posterior extorsión de la víctima, como sucede con el caso de los *ransomware*. A pesar del buen o mal uso que se les pueda dar, este tipo de redes ha alcanzado una gran popularidad. Hoy en día, existen diversas soluciones que proveen cierto grado de privacidad a sus usuarios, entre las

[2] También llamada criptografía de clave privada o secreta, es un método criptográfico que utiliza la misma clave para cifrar y descifrar mensajes.
[3] Conocida también como criptografía de clave pública, es un método criptográfico que usa un par de claves para el envío de mensajes, las cuales pertenecen a la misma persona que ha enviado el mensaje. Una de las claves, es pública y se la puede entregar a cualquier persona.
[4] *Internet Service Provid*er, empresa que brinda conexión a internet a sus clientes.
[5] Servidor, programa o dispositivo que actúa como intermediario entre la comunicación de un cliente y un servidor.

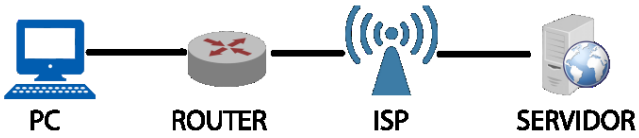
más utilizadas están TOR, I2P[6] y Freenet[7].

3. ONION ROUTING [1] [2]

Es un modelo centralizado formado por un conjunto de dispositivos llamados *onion routers* (OR), los cuales se sitúan entre los emisores y receptores para encaminar el tráfico. Este enrutado tiene como objetivo preservar la privacidad de la comunicación, tanto del origen y destino del mensaje como su contenido.

En una arquitectura de red tradicional se produce un enrutamiento directo. A grandes rasgos, esto es, del ordenador al router; del router al ISP; y por último del ISP al servidor de destino.

Imagen 1 – Enrutado Tradicional



La desventaja en este tipo de arquitectura es que es posible interceptar los paquetes de datos en un punto intermedio, y, por lo tanto, a través de los *headers* conocer el origen y destino del mensaje.

Onion Routing implementa un enrutamiento que permite enviar los paquetes por un camino no directo, esto es, a través de varios nodos aleatorios logrando una anonimidad en la conexión.

4. *The Onion Router* (TOR) [2] [3] [4]

Es un proyecto de software libre cuyo objetivo es el desarrollo de una red de baja latencia, en donde el intercambio de mensajes entre los usuarios y el uso de cualquier aplicación sean de carácter privado. Esta característica se obtiene a través del modelo de enrutamiento *Onion Routing,* tecnología que proporciona un alto nivel de anonimato tanto a nivel web como en

[6] *Invisible Internet Project*, proyecto que busca proveer comunicación segura y anónima a través de túneles unidireccionales.
[7] Plataforma *peer-to-peer* para la publicación y distribución de información propensa a la censura.

aplicaciones que usen otros tipos de protocolos, como el correo electrónico y la mensajería instantánea. Precisamente esta propiedad la distingue del resto de las alternativas actuales que no proveen soporte para otros tipos de aplicaciones o en su defecto lo gestionan de una manera compleja para el usuario.

Adicionalmente, TOR ofrece la posibilidad de crear "servicios ocultos" entre los usuarios de la red, lo que permite proveer servicios (FTP[8], SSH[9], etc.) manteniendo oculta la dirección IP y sin la necesidad de poseer una dirección pública. Estos servicios se representan mediante direcciones *onion*.

Imagen 2 – Dominio Onion

A pesar de que TOR proporciona más seguridad y privacidad que una red tradicional, como cualquier sistema, no es infalible. Existen múltiples métodos que permiten romper el anonimato de los usuarios. Sam DeFabbia-Kane, en su tesis *"Analyzing the Effectiveness of Passive Correlation Attacks on the Tor Anonymity Network"*, menciona varios vectores de ataques. Uno de estos se basa en un análisis de correlación *end-to-end*, en donde el atacante controla el primer y el último router del circuito virtual y emplea variables, como el tiempo de llegada y el tamaño del paquete, para correlacionar el flujo de datos entre los routers y así romper el anonimato de TOR.

4.1. COMPONENTES Y FUNCIONAMIENTO [1] [5]

El esquema de *onion routing* inicia cuando el emisor de la comunicación establece una conexión con el *proxy* de aplicación, cuya función es convertir los mensajes de un protocolo especifico a un formato genérico para que pueda ser interpretado por los nodos. TOR, a diferencia

[8] *File Transfer Protocol,* protocolo de red para la transferencia de archivos entre sistemas.
[9] *Secure Shell,* protocolo que tiene por objetivo conceder a un usuario el acceso seguro a un sistema remoto.

de otras implementaciones, no requiere un *proxy* de aplicación por cada protocolo de aplicación (HTTP[10], SMTP[11], etc.) sino que utiliza un solo protocolo genérico (SOCKS) para establecer conexiones con cualquier tipo de aplicación soportada.

El emisor, a través del *proxy* de aplicación envía un mensaje a un dispositivo llamado *Onion Proxy (OP)* y este es quien decide si acepta o rechaza la solicitud. En caso de rechazarla, envía un código de error al *proxy* de aplicación, cierra la conexión y queda a la espera de la próxima solicitud. Por el contrario, si llega a aceptar la solicitud, el *onion proxy* se conecta al primer *onion router* de la red, luego, crea una estructura llamada *onion* y define el camino aleatorio que debe seguir hasta su destino. Para lograr aquello, debe existir un intercambio de información previo entre cada nodo de la red y el *Onion Proxy.* Dependiendo del tipo de implementación de la red, este intercambio de información puede darse mediante consultas a un determinado sitio web o a través de un servidor de directorio, como sucede con la red TOR.

La eficiencia del modelo de "enrutamiento cebolla" se debe, en gran parte, a la colaboración de organizaciones e individuos que donan su ancho de banda y poder de procesamiento. Esto significa que cada nodo de la red es simplemente el dispositivo de un usuario que, según su posición dentro del circuito, asume uno de los tres roles existentes en la arquitectura: nodo de entrada, nodo intermedio, o nodo de salida.

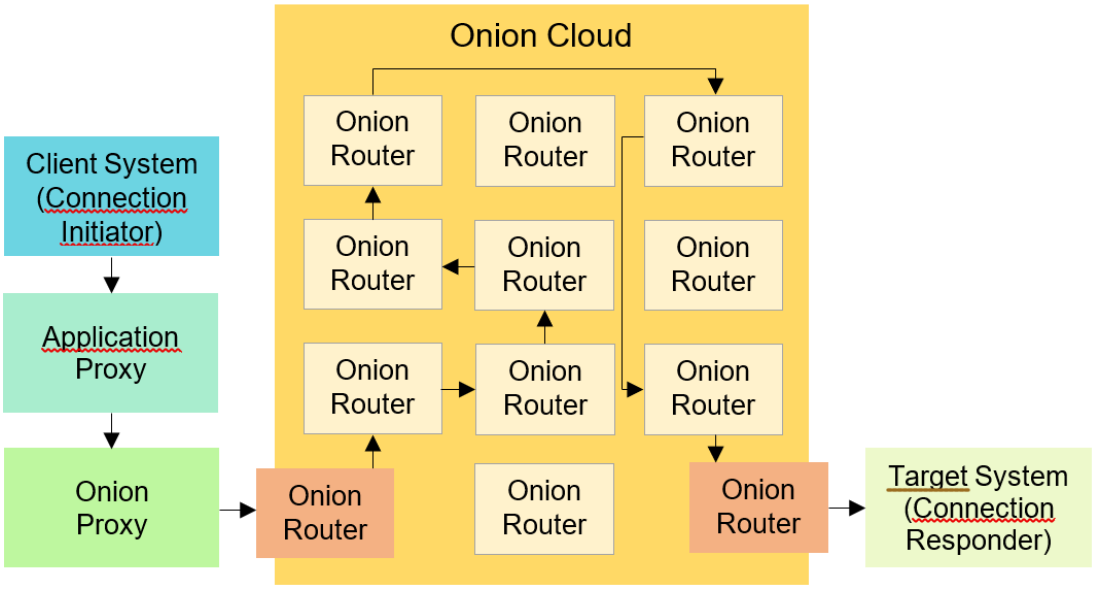
El nodo de entrada es el único router de la *onion network* que conoce la dirección IP del dispositivo iniciador de tráfico y es elegido de entre una lista predefinida de enrutadores especiales llamados *node guard.* La elección de los nodos intermedios y del nodo de salida se realiza de una forma probabilística ponderada, lo que significa que la probabilidad que se elija un router específico es proporcional a su ancho de banda anunciado. La función del nodo intermedio es reenviar la información cifrada al siguiente nodo del circuito, mientras que el nodo de salida es el responsable de remitir el tráfico

[10] *Hypertext Transfer Protocolo*, protocolo de comunicación que permite la transferencia de información en la *Word Wide Web.*
[11] *Simple Mail Transfer Protocol*, protocolo para transferencia simple de correo electrónico.

a su destino final por lo que es el único nodo que conoce la dirección IP del receptor del mensaje.

Una vez que el onion proxy obtiene la lista de los nodos disponibles, define la ruta aleatoria que la estructura *onion* debe seguir hasta el destino.

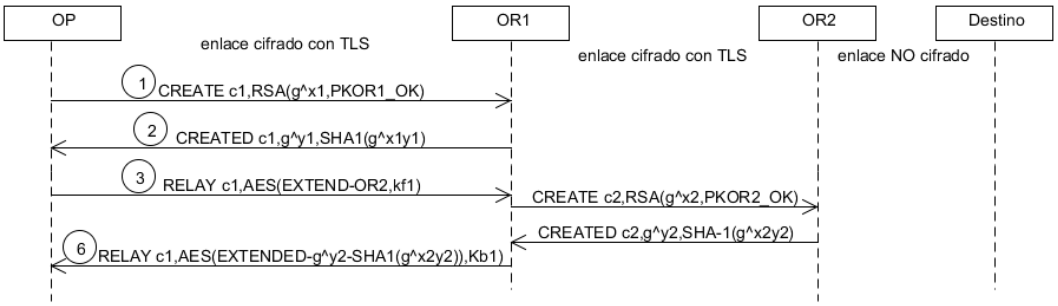
Imagen 3 - Componentes TOR [6]



4.2. INICIALIZACIÓN DEL CIRCUITO Y ENVÍO DE DATOS [7]

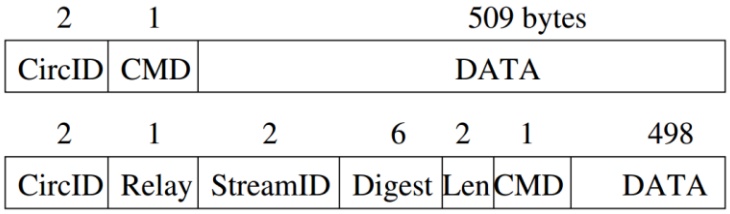
Después que el *onion proxy* define el camino aleatorio, construye cada circuito de esa ruta de forma incremental. En este contexto, un circuito corresponde al camino que existe entre dos dispositivos, sea este un OR u OP. La creación de los circuitos tiene como objetivo establecer un medio de comunicación seguro entre el *onion proxy* y cada *onion router*. No obstante, existe el inconveniente de que el *onion proxy* se conecta únicamente con el nodo de entrada de la red anónima, por lo que se debe emplear un enfoque telescópico para negociar una clave compartida con cada dispositivo de la red.

Imagen 4 – Construcción incremental circuito [8]



La unidad de comunicación de la red TOR es una estructura de 512 bytes llamada *cell* que se envía a través de conexiones TLS[12] prestablecidas que conectan a los routers entre sí. Según su función, una célula puede ser de control (*control cell*) o de retransmisión (*relay cell*) y almacena dos tipos de campos: *header* y *payload.* El campo encabezado contiene un identificador de circuito –*circID*– y un comando que define la función del *payload.* Los comandos principales pueden ser *padding, create o destroy* si se trata de una célula de control, o *relay* si es una célula de retransmisión.

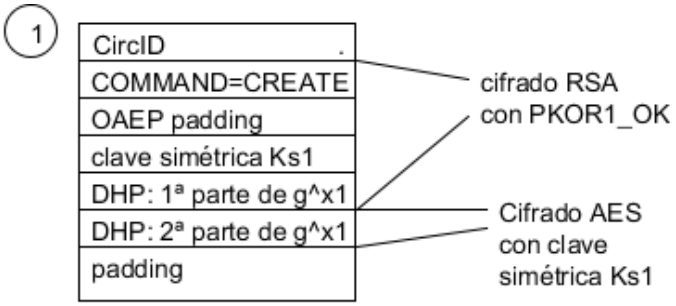
Imagen 5 – Estructura "Control Cell" y "Relay Cell" [9]



La creación de un nuevo circuito se da cuando el *onion proxy* mediante una conexión TLS, envía una célula de control al nodo de entrada de la *onion network.* En ella se define el *circID* y el comando requerido para su creación: "*create command".* En el *payload* de esta estructura se almacena el cálculo de la variable que inicia el proceso de intercambio de claves del algoritmo Diffie-Hellman[13] (g_{x1}). Este dato se encuentra cifrado con la clave pública del *entry node.*

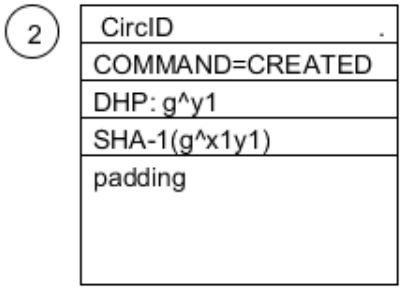
[12] *Transport Layer Security*, es un protocolo criptográfico que proporciona comunicación segura por la red.
[13] Protocolo de establecimiento de claves que se da a través de un canal inseguro y de manera anónima entre entidades que no han tenido un contacto previo.

Imagen 6 – Célula create [10]



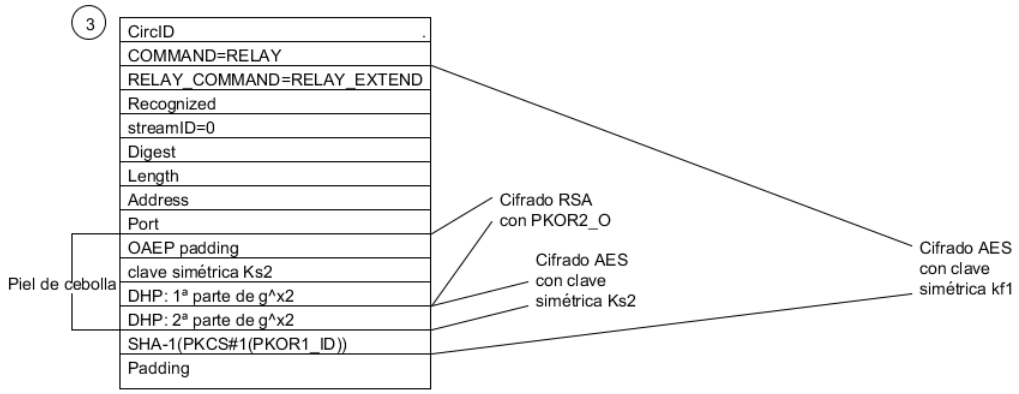
El nodo de entrada, al recibir la célula de creación, la descifra con su clave privada y crea una estructura –*created cell*– que actúa como acuse de recibo en la comunicación. En el *payload* de esta estructura se almacena el cálculo de la variable requerida para finalizar el proceso de intercambio de claves del algoritmo de Diffie-Hellman(g_{y1}). Al finalizar esta instancia, el *onion proxy* y el *entry node* han establecido una clave compartida que se usa para generar dos claves simétricas que cifren posteriormente la comunicación en cada sentido. (*forward key-backward key).*

Imagen 7 – Célula created [11]



El siguiente paso consiste en extender el circuito, de tal forma que sea posible establecer una clave compartida entre el *onion proxy* y el nodo intermedio. Para esto, el OP crea una célula de retransmisión del tipo –*relay extend*– donde se indica la dirección del siguiente nodo de la ruta y se adjunta la variable g_{x2} cifrada con la clave pública del nodo intermedio. Consecutivamente, esta estructura es cifrada con el *forward key* generado anteriormente y se la envía al *entry node*.

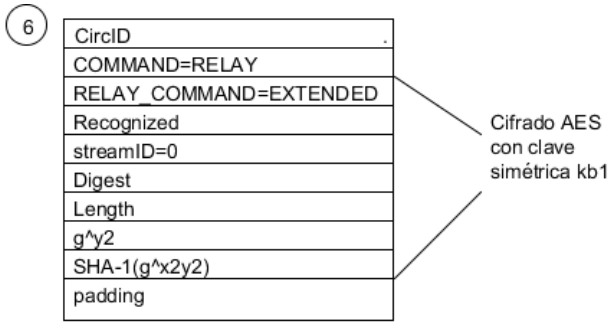
Imagen 8 – Célula relay extend [12]



Cuando el nodo de entrada recibe la solicitud de extender el circuito, crea una nueva estructura llamada *create cell* y adjunta el contenido cifrado de la variable g_{x2}, el cual es enviado a la dirección especificada en el *relay extend,* es decir al nodo intermedio. Este nodo debe responder la solicitud al *entry node* con una estructura *created cell* donde se adjunte la variable g_{y2}.

El nodo de entrada recibe la célula *created*, obtiene el contenido de su *payload* y lo agrega a una nueva estructura –*relay extended*– que es enviada al *onion proxy.* De esta forma el OP y el nodo intermedio han establecido una clave compartida que permite generar dos claves simétricas entre el *onion proxy* y el nodo intermedio, usada para cifrar posteriormente los datos en ambos sentidos de la comunicación.

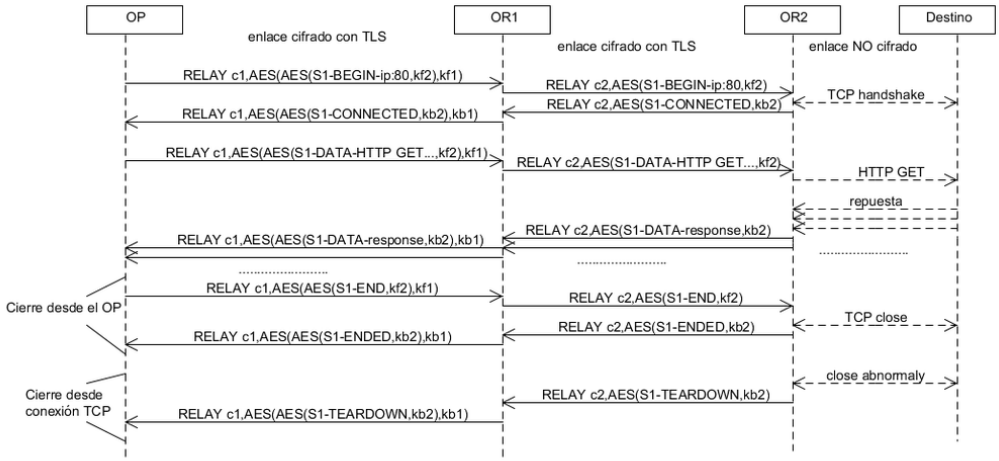
Imagen 9 – Célula relay extended [13]



De forma similar se continúa el proceso hasta alcanzar el último nodo del circuito. Al finalizar todo el proceso, el *onion proxy* comparte individualmente con cada *onion router* dos claves simétricas (*forward key* y *backward key*), utilizadas para cifrar las estructuras *relay_begin, relay_data, relay_connected y relay_end*.

Establecido el circuito, el OP crea y envía una célula de retransmisión de tipo *relay begin* al nodo de salida. La mencionada estructura, contiene la dirección IP y el puerto del destino final de la comunicación, por lo que se encuentra protegida por varias capas de cifrado simétrico. Cuando un nodo recibe la estructura, descifra la capa más exterior y la envía al siguiente salto del circuito. De esta forma, únicamente el nodo de salida conoce la dirección del servidor de destino.

Imagen 10 – Conexión anónima al servidor [14]



Al recibir la estructura *relay begin*, el *exit node* obtiene la dirección y el puerto de destino e intenta establecer una conexión TCP[14] con el *host*. En el caso de que no sea posible establecer la conexión, el nodo de salida devuelve una célula del tipo *relay end.* En caso contrario, establece la conexión y devuelve una célula *relay connected* al nodo anterior, y este a su vez, lo propaga hacía atrás hasta llegar al *onion proxy.*

Para finalmente comunicarse de forma anónima con un servidor, los *request* y los *response* son encapsulados dentro de una célula llamada *relay data*. Si los datos viajan desde el *onion proxy* hacia el servidor, estos son cifrados iterativamente con el *forward-key* de cada *router* siguiendo el orden de la ruta. Por el contrario, si los datos viajan en sentido contrario, el *payload* se cifra iterativamente con el *backward-key* a medida que atraviesa cada

[14] Protocolo de control de transmisión, garantiza que los datos serán entregados a su destino sin errores y en el mismo orden en el que se transmitieron.

salto. Al llegar al *onion proxy*, la *data* es descifrada con su respectiva clave anteriormente generada y compartida.

5. BITCOIN [15] [16]

La moneda *bitcoin* (BTC o ₿), es una forma relativamente reciente de dinero electrónico descentralizado y criptográfico, creada como una alternativa a las monedas fiduciarias[15]. Durante años se desconoció la verdadera autoría de esta moneda digital y su creación originalmente fue atribuida al seudónimo Satoshi Nakamoto por lo expuesto en el artículo *Bitcoin: A Peer-to-Peer Electronic Cash System*. Recientemente, el científico computacional australiano Craig Wright se atribuyó aquel seudónimo y por ende la autoría de la moneda digital.

Desde su aparición en el año 2009, el *bitcoin* se ha extendido, pero sin llegar a ser completamente aceptado. Grandes empresas como Microsoft, WordPress y Dell han adoptado el *bitcoin* como medio de pago, lo que supone, en un futuro, una influencia positiva en la adopción general del *bitcoin*. Por el contrario, el gobierno suele oponerse a su aceptación por la dificultad que les representa establecer políticas impositivas sobre las transacciones efectuadas por este medio. Adicionalmente, otro impedimento que ha limitado su expansión reside en el mal uso que se le ha dado, estos son, apuestas, extorsión y ciberdelincuencia; acciones respaldadas por la facilidad que provee *Bitcoin* para realizar transacciones internacionales y anónimas.

El término *bitcoin*, a más de ser usado para representar la unidad monetaria, también es utilizado para referirse al protocolo y a la red P2P[16] que lo sustenta. Generalmente se usa la palabra *Bitcoin* con mayúscula cuando se habla de la red o del protocolo, y se usa el vocablo *bitcoin* en minúsculas cuando se hace referencia a la unidad monetaria.

[15] Moneda que se basa en la fe o confianza de la comunidad, no está respaldada por un bien físico sino por una promesa de pago por parte de la entidad emisora.
[16] Red de ordenadores en la que todos o algunos aspectos funcionan sin clientes ni servidores fijos. Cada nodo actúa simultáneamente como cliente y servidor.

Bitcoin es una tecnología compleja que requiere diversos componentes para su funcionamiento. En términos generales, estos comprenden: software para su utilización, protocolo de comunicación y usuarios que le den valor a la moneda o que sean parte de la red, colaborando con el poder de procesamiento de su hardware.

Para comercializar *bitcoins* se requiere un programa cliente que permita enviar y recibir transferencias, llamado *wallet*. Estas aplicaciones pueden ser nativas (BitcoinCore, Electrum) y son ejecutadas en ordenadores y dispositivos móviles; o también, pueden ser aplicaciones webs e incluso pueden llegar a estar contenidos en algún tipo especial de hardware. Independientemente del tipo de monedero, el objetivo principal de los mismos es generar una o muchas direcciones *bitcoins* para el envío o recepción de transferencias. El monedero o *wallet* está asociado a una clave privada que el programa cliente utiliza para firmar las transacciones, garantizando de esta forma su integridad.

Imagen 11 – Dirección bitcoin [17]

1DNocez6uyHi8WfBswM3tjne2vJA15KfgH

Al igual que en el mundo real, una transacción es una transferencia de valores, pero esta se da entre monederos *bitcoins*. Desde una perspectiva muy amplia, cuando una persona “A” transfiere un *bitcoin* a una persona “B”, lo que hace realmente es asociar ese *bitcoin* con la clave pública de “B” y posteriormente firmarlo con la clave privada de “A”, garantizando que la transacción la realizó el propietario del *bitcoin*. Sin embargo, para que las transacciones realizadas pueden ser confirmadas, deben ser validadas y retransmitidas por todos los nodos de la red.

Bitcoin se estructura como una arquitectura de red entre pares, lo que significa que no hay un servicio centralizado o una jerarquía dentro de la red,

por lo que los equipos que conforman la red, llamados nodos, se interconectan entre sí. Aunque todos los nodos de la red *Bitcoin* son iguales para el sistema, estos pueden asumir diferentes roles, dependiendo de la funcionalidad elegida por el usuario. Un nodo *Bitcoin* puede estar en una de estas tres categorías (Oro y finanzas, 2014):

1. Los que sólo emiten transacciones (*wallet).*
2. Los que transmiten transacciones.
3. Los que transmiten y minan transacciones.

De estas tres categorías, los nodos encargados de minar transacciones son los que requieren mayor capacidad computacional y se los conoce como “mineros”. Estos constituyen una pieza fundamental del sistema debido a que son los encargados de expedir nuevas unidades de *bitcoins,* pero se encuentran limitados a una cantidad total máxima de 21 millones de unidades. (Oro y finanzas, 2014).

Inicialmente, la minería de *bitcoins* empezó realizándose con CPU[17] estándares y a medida que se incrementaba la dificultad de generar criptodivisas fue necesario más poder de procesamiento, por lo que se emigró al uso de tarjetas gráficas. Posteriormente, el *mining* se llegó a profesionalizar a través del empleo de equipos diseñados especialmente para estos fines (FPGA’s[18], ASIC’s[19]), lo que resultaba en un alto costo de inversión. Para hacer frente a esta evolución tecnológica, los mineros independientes suelen agruparse y trabajar conjuntamente al combinar todo su poder de procesamiento, concepto definido como *mining pools*.

5.1. FUNCIONAMIENTO Y COMPONENTES [18] [11] [19] [20] [21] [22] [23] [24] [25] [26] [27]

El sistema Bitcoin requiere de componentes que suplanten el proceso que implicaría contar con una autoridad central. A continuación, se enumera

[17] Unidad central de procesamiento, es un *hardware* dentro de una computadora u otro dispositivo programable que interpreta las instrucciones de programa mediante la realización de operaciones aritméticas básicas, lógicas y de entrada y salida.
[18] Equipos especialmente diseñados para programar su poder de procesamiento en actividades específicas.
[19] Circuito integrado desarrollado para aplicaciones específicas.

cada componente y en líneas posteriores se detalla su funcionamiento e importancia dentro de la red.

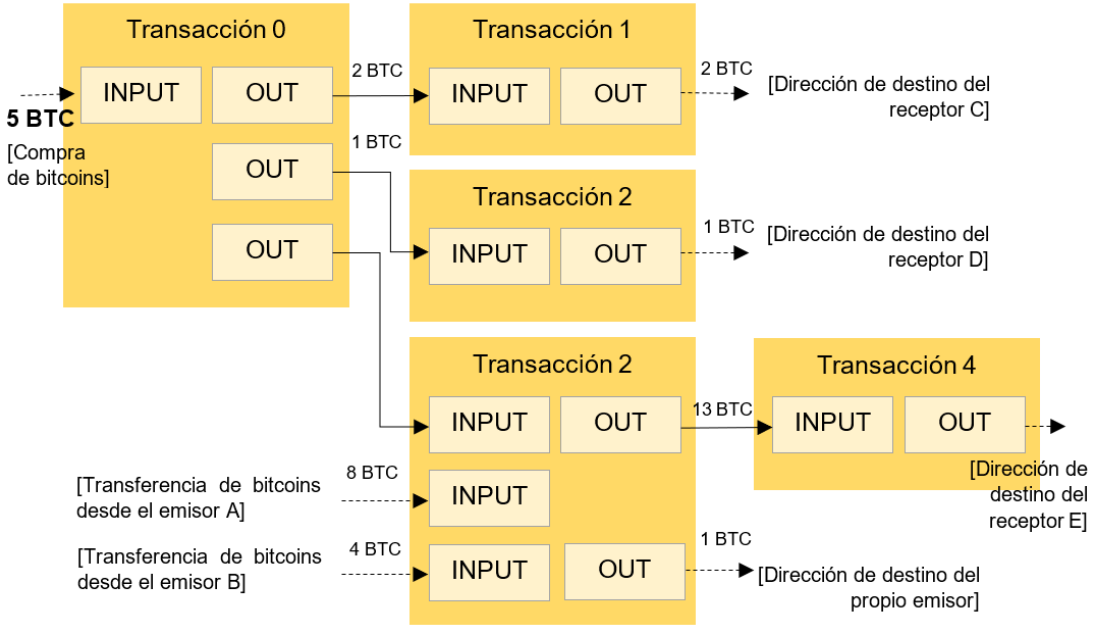
1. Transacciones
2. Bloques
3. *Proof of Work*[20] o prueba de trabajo
4. *Blockchain* o cadena de bloques

Una transacción *Bitcoin* es una transferencia de valores entre monederos, donde se asocia un determinado *bitcoin* o una fracción de este, a un nuevo propietario. Como resultado de la fluctuación constante de esta moneda digital y por consiguiente su incremento del poder adquisitivo –*actualmente 1 BTC = $13.411*–, surge una terminología que identifica una fracción de la moneda: un "Satoshi", que es la unidad más pequeña del bitcoin (1 bitcoin = 100'000.000 Satoshis).

En el sistema *Bitcoin*, una transacción consta de tres partes principales: *inputs, outputs* y el monto a transferir. Las entradas de una transacción son referencias a los valores disponibles que tiene el usuario para transferir y provienen de salidas –*no gastadas*– de transacciones anteriores. Es decir que, el número total de bitcoins disponibles de un usuario corresponde a la suma de todas las entradas de una transacción. Por otro lado, la salida de una transacción es una referencia de la asignación de cierta cantidad de bitcoins a un nuevo usuario. Evidentemente, en la mayoría de los casos, la cantidad disponible de bitcoins de un usuario no coincide con el monto de bitcoins que se desea transferir. En estos casos, se adiciona una salida extra a la transacción con la cantidad restante entre los bitcoins disponibles y los bitcoins a transferir menos el coste por concepto de comisión. El destino de esta salida será la dirección *bitcoin* del propio emisor.

[20] Sistema que requiere que el cliente del servicio realice algún tipo de trabajo que tenga cierto coste y que pueda ser verificado fácilmente en el lado del servidor.

Imagen 12 - Entradas y salidas de una transacción [28]



Las comisiones son aquellos valores que un minero obtiene por cada transacción cuando resuelve la *proof of Work* de su bloque personal. Estas tarifas equivalen a incentivos que los clientes incluyen en sus transacciones con el objetivo de que estas sean confirmadas rápidamente, porque generalmente, un minero dará prioridad a una transacción que le genera ganancias. Contrario a lo que se podría suponer, existen casos donde un minero o grupo de mineros, prefieren resolver su bloque personal "vacío", esto es, con una única transacción que contenga solamente el *coinbase* (recompensa que se obtiene al resolver la prueba de trabajo).

El *coinbase* a diferencia de las comisiones, representa la única forma que poseen los mineros para la creación de nuevas monedas digitales. No obstante, el diseño del sistema limita la cantidad máxima de monedas en el tiempo, fijando un tope de 21 millones de bitcoins. Para cumplir con este requisito, el precursor de la moneda digital implementó un concepto denominado *halving*, el cual hace referencia a la reducción a la mitad –*cada cierto tiempo*– del premio que reciben los mineros por completar un bloque de transacciones. Actualmente, el *coinbase* alcanza un valor de 12.5 bitcoins y se prevé que para el año 2140 se genere la última moneda bitcoin de la historia.

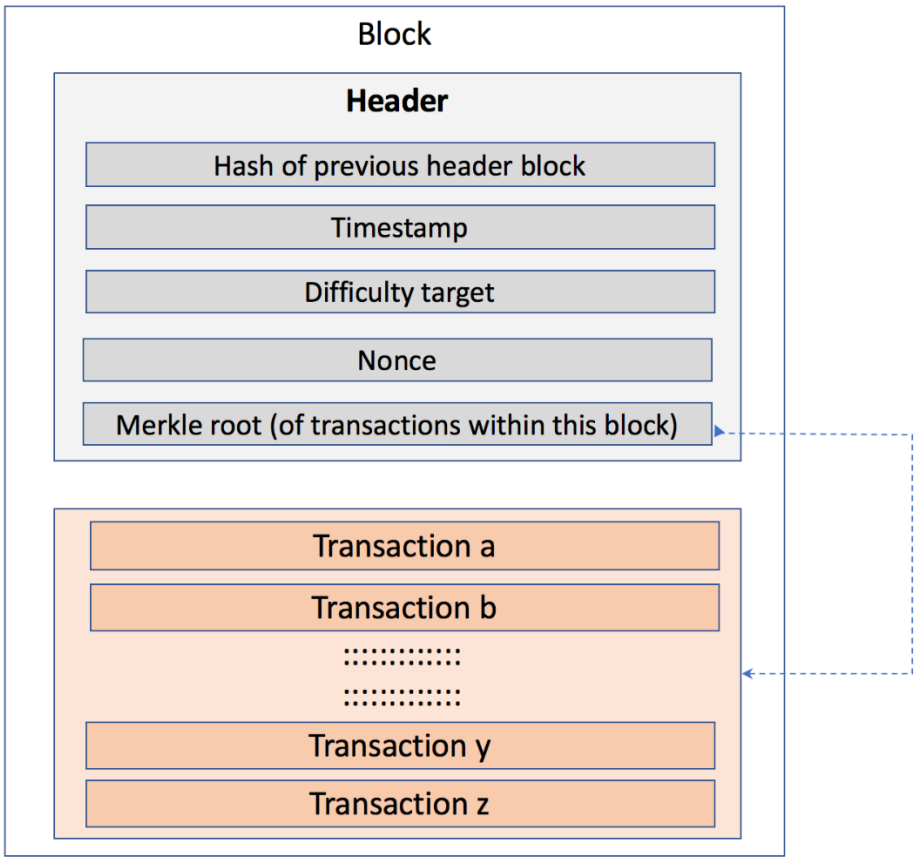
El ciclo de vida de una transacción inicia cuando el *wallet* del emisor firma con su clave privada la solicitud de transacción de bitcoins a una dirección de destino. Luego la transacción se propaga a través de la red, y cada nodo, de forma independiente, valida la transacción y la agrega a su bloque personal siguiendo la estructura organizativa de un árbol de Merkle[21]. Para cada nodo de la red Bitcoin, validar una transacción significa comprobar que esta cumple una serie de condiciones, entre las cuales, la más importante, se enfoca en asegurar que las entradas de esa transacción no hayan sido gastadas, es decir que el emisor disponga de la cantidad necesaria de *bitcoins* para efectuar la transferencia.

Un bloque *Bitcoin* es una estructura creada por cada nodo y contiene una o más transacciones ya validadas y organizadas en forma de árbol de Merkle. Además de las transacciones, cada bloque contiene un *header* con los siguientes campos:

1. Versión de protocolo que usa el bloque.
2. *Hash sha256* del encabezado del bloque anterior.
3. Raíz del árbol Merkle, representa el *hash* de todos los *hashes* de las transacciones del bloque.
4. *Timestamp,* que se genera cuando se crea el bloque.
5. *Bits,* valor objetivo del bloque.
6. *Nonce,* valor numérico que se incrementa cada vez que el nodo minero no logra resolver la prueba de trabajo del bloque.
7. Dificultad

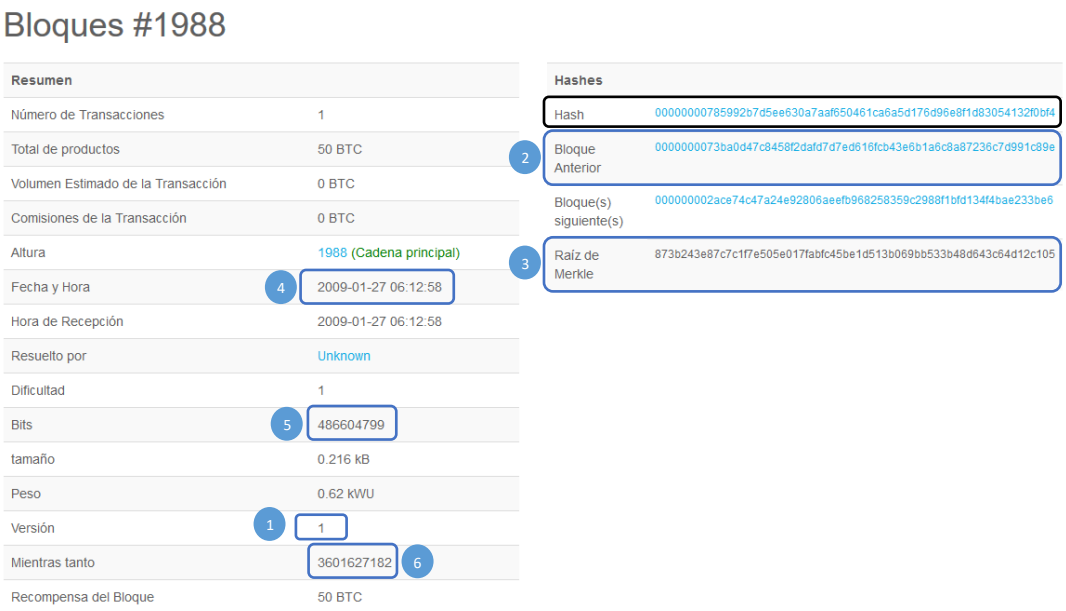
[21] Es una estructura de datos en forma de árbol de valores en donde cada nodo interior (nodos que no son los nodos finales) es el resultado de aplicar una función *hash* sobre el valor de los nodos hijos hasta llegar a un nodo raíz llamado Merkle *root* del árbol.

Imagen 13 - Estructura Bloque Bitcoin [29]



La prueba de trabajo consiste en generar un *hash* del *block header*, de manera que el valor creado cumpla la condición de ser menor a un "valor objetivo, dato que es calculado internamente por el sistema a partir del campo "dificultad" del *block header*, y se ajusta automáticamente para mantener la complejidad de resolver un bloque cada diez minutos. Resolver una prueba de trabajo implica concatenar los cinco primeros campos del encabezado de un bloque y luego concatenar ese resultado iterativamente con el campo *nonce* hasta lograr generar un *hash sha256* que cumpla la condición de ser menor que el "valor objetivo".

Imagen 14 – Bloque Bitcoin [30]



6. IMPLEMENTACIÓN LABORATORIO ANÁLISIS DEL MALWARE [31] [32]

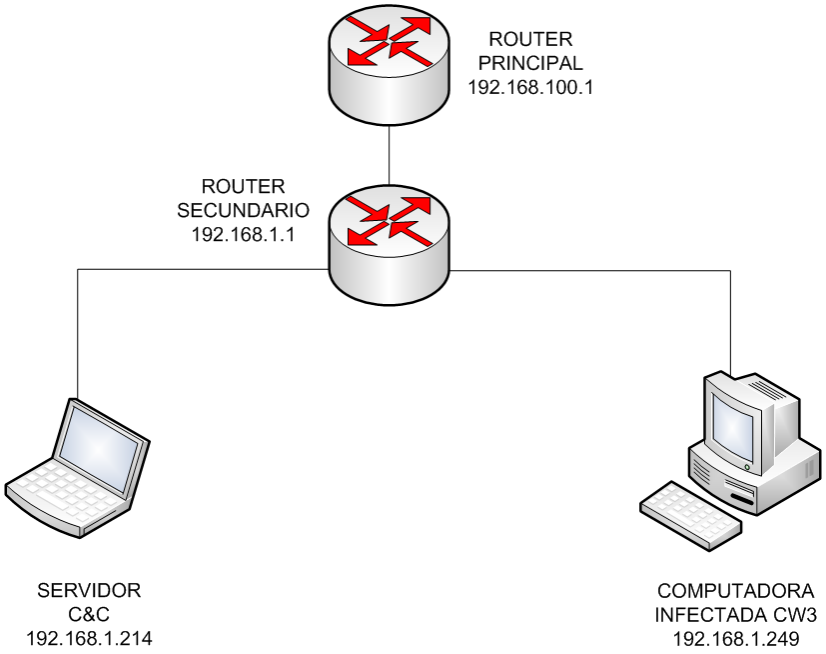
Analizar un *malware*[22] le exige a cualquier investigador el empleo de medidas de seguridad que impidan la propagación del código malicioso en la red. Por tanto, es necesario implementar un ambiente aislado de trabajo que facilite la recolección de datos y el análisis de la muestra. Con base a la anterior se ha definido la siguiente arquitectura de red:

Tabla 1 - Descripción de los dispositivos

Dispositivo	Dirección IP	Descripción
Router principal Huawei HG8245h	192.168.100.1	Dispositivo con conexión a internet
Router secundario TP-LINK WR741ND	192.168.1.1	Dispositivo con *firmware* OpenWrt[35]
PC víctima	192.168.1.249	Computadora de escritorio con sistema operativo Windows 8.1 PRO
Falso servidor Comando y Control[36]	192.168.1.214	*Netbook* con sistema operativo Windows 7

[22] Código o programa malicioso que tiene como objetivo dañar un sistema o causar un mal funcionamiento.

Imagen 15 – Arquitectura de red



Cryptowall 3 o CW3, es un tipo de *ransomware* que recibe instrucciones de un servidor *Command and Control* para llevar a cabo el proceso de cifrado de los archivos. Sin embargo, es muy común que a medida que surgen nuevas versiones del *malware*, los autores desactiven sus servidores C&C para evitar el análisis posterior de muestras. Por tal razón, en este trabajo se implementa un falso servidor C&C [anexo A] para que simule los *request y response* requeridos por el *ransomware*. Para esto es necesario habilitar y configurar el redireccionamiento de paquetes en el *router* secundario de la red, de manera que todas las peticiones que realice el *ransomware* hacia la WAN[23] sean redirigidas a un servidor web local que simule la comunicación entre el *malware* y el servidor. Esta característica conocida como *port-forwarding* está ausente en la mayoría de *routers* domésticos, de modo que se requirió la instalación previa del *firmware* OpenWrt.

Para configurar el redireccionamiento de paquetes en el router secundario se debe establecer una conexión remota a través de un cliente

[23] *Wide Area Network*, red de computadoras que une varias redes locales, aunque estas se encuentren separados físicamente.

SSH y acceder al archivo de configuración del *firewall*[24] localizado en el directorio /etc/config/firewall, posteriormente se debe adicionar una nueva regla de filtrado de paquetes. La imagen 16 muestra la regla de filtrado donde se establece que cualquier paquete cuyo origen sea la red LAN[25] y cuyo destino sea la red WAN sea redirigido a la dirección IP 192.168.1.214 donde se estará ejecutando el falso servidor C&C.

Imagen 16 – Configuración DNAT firewall

```
config 'redirect'
        option 'target' 'DNAT'
        option 'name' 'Redireccion'
        option 'src' 'lan'
        option 'dest' 'wan'
        option 'proto' 'tcp'
        option 'dest_ip' '192.168.1.214'
```

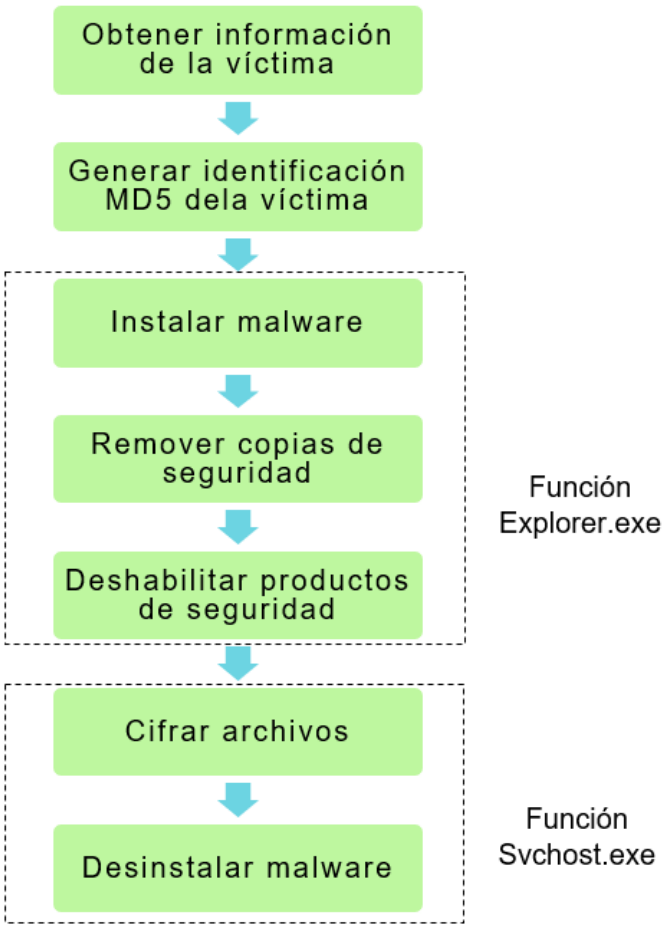
7. ANÁLISIS CRYPTOWALL V3 [13] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41]

Las observaciones descritas en esta sección están sustentadas en el documento *"Lucrative Ransomware Attacks: Analysis of the CryptoWall 3 Threat"* publicado por *Cyber Threat Alliance*[26]. No obstante, las imágenes y datos presentados corresponden a resultados de pruebas reales efectuadas en un entorno controlado con el objetivo de corroborar el fundamento teórico expuesto.

El análisis de *Cryptowall* 3 inicia con la descripción de un diagrama secuencial de la rutina, desde su infección hasta el cifrado de los archivos. Posteriormente el análisis se profundiza desde tres perspectivas: las instrucciones previas al cifrado (función explorer), la comunicación con el servidor (función svchost) y el cifrado de los archivos.

[24] *Software* o *hardware* que monitorea el tráfico entrante y saliente de una red y decide si permite o bloquea el acceso en función de un conjunto definido de reglas de seguridad.
[25] *Local Area Network*, red de computadoras que abarca un área física reducida.
[26] Organización sin fines de lucro cuyo objetivo es el intercambio de información de amenazas de seguridad para tomar medidas que permitan proteger a sus clientes.

Imagen 17 – Rutina de Cryptowall 3



La rutina del *malware* inicia al generar un *Import Address Table* (IAT), que es una estructura que contiene las direcciones virtuales y nombres de las librerías y funciones que requiere el *malware*. A cada dirección, librería y función se le aplica el algoritmo CRC32[27] para representarlo.

Para poder hacer uso de sus funciones, *Cryptowall* debe cargar previamente sus librerías en el *Process Environment Block*[28] (PEB). Al igual que en el IAT, los nombres de las librerías están representados por su respectiva cadena CRC32 y, por lo tanto, es posible comparar el nombre de cada función en el PEB con el nombre de cada función en el IAT. En caso de encontrar alguna coincidencia, el *malware* obtiene la dirección virtual de la librería y procede a cargar las funciones requeridas para el resto de la rutina.

[27] Código de detección de errores usado para detectar cambios accidentales de datos. Este método emplea la verificación por redundancia cíclica, que es un tipo de función hash.
[28] Estructura de datos que contiene parámetros asociados al proceso que se está ejecutando en el sistema.

La identificación unívoca de cada víctima se obtiene generando un *hash* MD5[29] a partir de la información recolectada de la computadora del usuario infectado. Esta información está compuesta por los siguientes datos concatenados y ofuscados:

1. Nombre de la computadora
2. Número de serie del volumen del disco
3. Información del Procesador
4. Versión del Sistema Operativo

Más adelante, este identificador MD5 es utilizado como nombre de un nuevo evento en el sistema para así garantizar la existencia de una sola instancia del malware.

El identificador MD5 junto con otros datos son almacenados dentro de una estructura que será enviada en una solicitud posterior al servidor remoto C&C. Los datos y sus respectivos métodos de obtención son detallados en la tabla posterior.

Tabla 2 – Datos adicionales

Dato	Método de obtención
Nombre de la campaña	Información agregada por el malware según la campaña de infección.
Longitud del nombre de la campaña	Se la obtiene a partir del registro anterior.
Identificador MD5 de la víctima	Generando el *hash* MD5 a partir de los siguientes datos concatenados: [ComputerName]-[VolumeSerial]-[ProcessorInformation]-[OS]
Versión del SO	Consultando el parámetro *OSMajorVersion* y *OSMinorVersion* del PEB.
Arquitectura del CPU	A través de un llamado a la función *ZwQueryInformationProcess(ProcessWow64Information)* Esta función retorna un tipo de dato *integer:* 1→32 bits 2→64 bits
Privilegios del usuario	En los sistemas operativos Windows Vista o superior, se hace una llamada a la rutina *ZwQueryInformationToken(tokeElevation)*, la cual retorna el nivel del privilegio del proceso en ejecución.

[29] Algoritmo de reducción criptográfico de 128 bits usado para comprobar la integridad de un archivo.

Para recibir instrucciones de ejecución, *Cryptowall* requiere comunicarse con un servidor C&C. Para esto, dispone de una lista de direcciones URL cifradas con el algoritmo RC4[30].

La solución de seguridad *TrendMicro* define a un "*Command and Control server"* como un servidor usado remotamente para enviar comandos maliciosos a una *Botnet*, o a una red comprometida de computadoras. Devincenzi (2011) conceptualiza a la *Botnet* como "una red formada por un conjunto de hosts de un tamaño determinado bajo el control centralizado de un tercero denominado *Bot Herder*".

Después de que el *ransomware* descifra y encuentra una dirección URL válida, identifica el proceso *explorer.exe* y genera una nueva instancia de este a través de una llamada a *ZwCreateSection*, función responsable de la creación y asignación de nuevos espacios de memoria. *Cryptowall* se copia a sí mismo en esta nueva instancia e inicia un *thread* que será el responsable de:

1. Instalación y persistencia
2. Borrado de copias de seguridad
3. Desactivación de servicios
4. Generación de una nueva instancia de *svchost.exe,* función que iniciará un nuevo *thread* responsable del:
    i. Cifrado de los archivos
    ii. Comunicación con el servidor
    iii. Desinstalación del malware al finalizar la rutina

Para un mejor entendimiento, la continuación del análisis es divido en dos partes. La primera parte, hace referencia a las instrucciones previas al cifrado de los archivos y se llevaran a cabo en la función *explorer.exe.* La segunda parte del análisis describirá el proceso de comunicación con el servidor C&C y el posterior cifrado de archivos. Estas últimas funciones se efectuarán en el proceso *scvhost.exe.*

[30] Esquema de cifrado de flujo (no basado en bloques) simétrico.

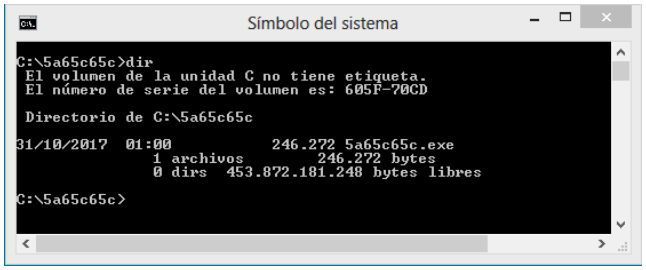
7.1. FUNCIÓN EXPLORER

Inicia con la generación de un IAT como requisito previo al uso de sus librerías y funciones. Posteriormente se genera un identificador unívoco para la víctima, para esto, se concatenan los siguientes datos: nombre de la computadora, número de serie del volumen, información del procesador y la versión del sistema operativo. A este texto concatenado se le aplica el algoritmo de reducción criptográfico MD5. A su vez, al resultado del proceso anterior, se le aplica el algoritmo CRC32. Finalmente, se utiliza este dato –*en minúsculas*– junto con el *System Drive*[31] para crear un *path* en el directorio raíz de la computadora de la víctima. En esta nueva carpeta se crea una copia temporal del *malware* con el mismo nombre de la carpeta que lo contiene.

Tabla 3 – Identificador de la computadora de la víctima

Hash MD5	0656DB2FADD6EC2A590AD75FCBDA1816
CRC32 (*hash* MD5)	5a65c65c
path	C:\5a65c65c\5a65c65c.exe

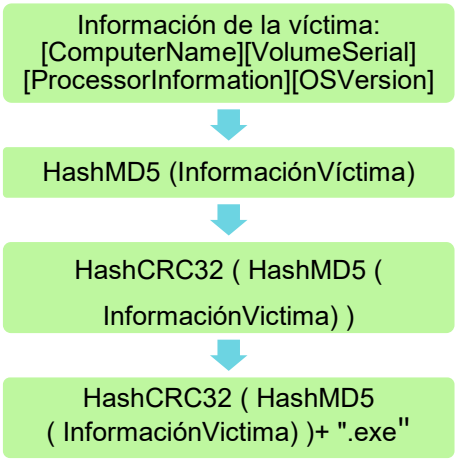
Imagen 18 – Directorio creado por malware



El resumen de la rutina mencionada se representa en el siguiente gráfico secuencial.

[31] Unidad de disco donde se encuentra instalado el sistema operativo principal.

Imagen 19 – Rutina creación directorio del malware

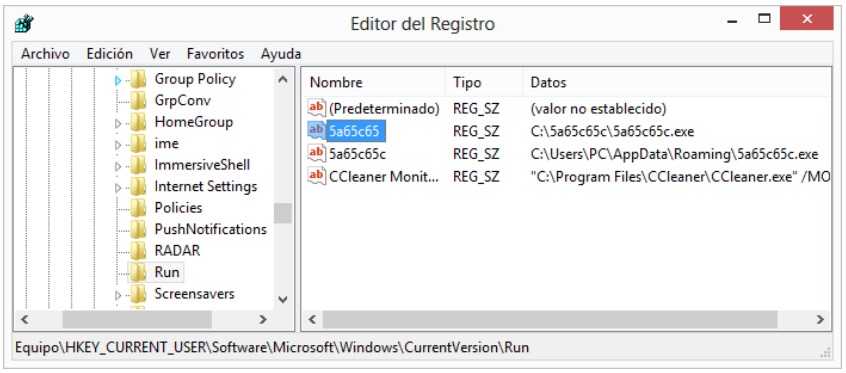


Creado el directorio, el malware modifica las siguientes claves de registro.

I. HKCU\Software\Microsoft\Windows\CurrentVersion\Run
II. HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce

En ambas claves de registro, se agrega un nuevo “valor de cadena” con su respectiva “información de valor”. Los datos agregados corresponden al CRC32 (hash MD5) de la víctima y al *path* que contiene el ejecutable.

Imagen 20 – Clave de registro creada



Estas modificaciones en el registro de Windows garantizan la persistencia del *malware*, al establecer su ejecución automática en cada inicio de sesión, aun cuando el usuario inicie en modo a prueba de fallos.

Para asegurar el mayor éxito posible en el proceso de extorsión, CW3 debe asegurarse que los datos a cifrar sean irrecuperables para el usuario,

por lo que, antes de cifrarlos, debe remover las *shadow copies*[32] y desactivar los productos de seguridad del sistema operativo. Este proceso se logra a través de llamadas a *WinExec*. Esta función forma parte del API[33] de *Windows* y es la responsable de ejecutar una determinada aplicación a través de la línea de comandos. Los comandos enviados a la función junto con sus respectivas descripciones son detallados en el recuadro siguiente.

Tabla 4 – Comandos ejecutados *WinExec*

Comando	Descripción
vssadmin.exe Delete Shadows/All/Quiet	Elimina todas las copias de seguridad de un volumen.
bcdedit/set {default} recoveryenabled No	Deshabilita la opción de reparación automática de Windows.
bcdedit/set {default} bootstatuspolicy ignoreallfailures	Establece por defecto la opción “Iniciar Windows normalmente”.

Adicionalmente, CW3 desactiva los servicios de seguridad de Windows.

Tabla 5 – Servicios de seguridad desactivados

Servicio	Descripción
wscsvc	*Security Center Service*
WinDefend	*Windows Defender Service*
wuauserv	*Windows Update Service*
BITS	*Background Intelligent Transfer Service*
ERSvc	*Error Reporting Service*
WerSvc	*Windows Error Reporting Service*

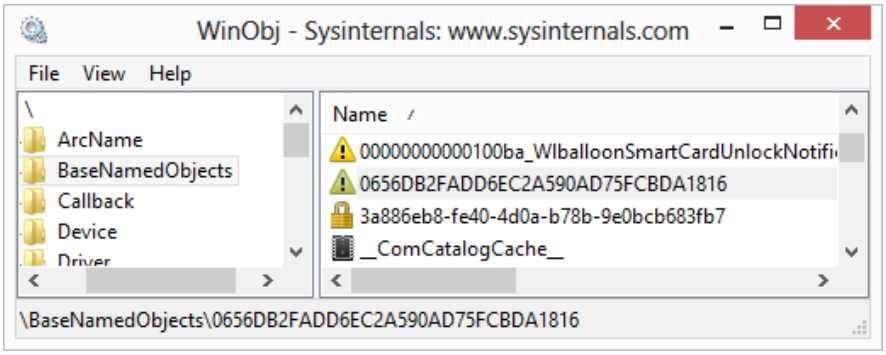
Finalmente, la función explorer.exe genera una nueva instancia del proceso svchost.exe y se inyecta a sí mismo.

[32] Tecnología incluida en Microsoft Windows que permite al usuario hacer copias de respaldo del contenido de su computadora.
[33] Conjunto de subrutinas, funciones y procedimientos que pone a disposición una biblioteca para ser utilizado por otro software.

7.2. FUNCIÓN SVCHOST

Al igual que la función *explorer.exe* se genera un IAT usando la misma técnica descrita en la sección anterior. Para garantizar que solo se ejecute una copia del *malware* a la vez, se crea un evento con el nombre del identificador MD5 de la víctima. Este evento es categorizado dentro del *namespace BaseNamedObjects* y gestionado por el Manejador de Objetos de Windows[34].

Imagen 21 – Evento creado en el Manejador de Objetos



Creado el evento, el *malware* procede a consultar la siguiente clave de registro.

HKU\[SID]\Software\[MD5-Key]\[char]k

De la clave de registro, el término [SID] corresponde al identificador de seguridad del usuario dueño del proceso actual, [MD5-KEY] es el identificador único de la víctima y [char] representa el tercer carácter del identificador MD5 de la víctima. Esta será usada más adelante para almacenar la clave pública RSA[35] proveída por el servidor C&C.

El servidor C&C se comunica con el *malware* a través de peticiones HTTP. *Cryptowall* cuenta con una lista de URL de distintos servidores de Mando y Control. Como no es posible garantizar la disponibilidad de una dirección URL en particular, CW3 procede a entrar en un *loop* infinito de *request* a cada URL con un *sleep time* de 15 segundos entre cada iteración

[34] Subsistema responsable del manejo de los recursos del sistema (físicos y lógicos).
[35] Sistema criptográfico de clave pública donde la seguridad del algoritmo radica en el problema de la factorización de números enteros. Se lo utilizada tanto para cifrar como para firmar digitalmente.

con el fin de descubrir una URL válida y disponible que permita recibir instrucciones del servidor.

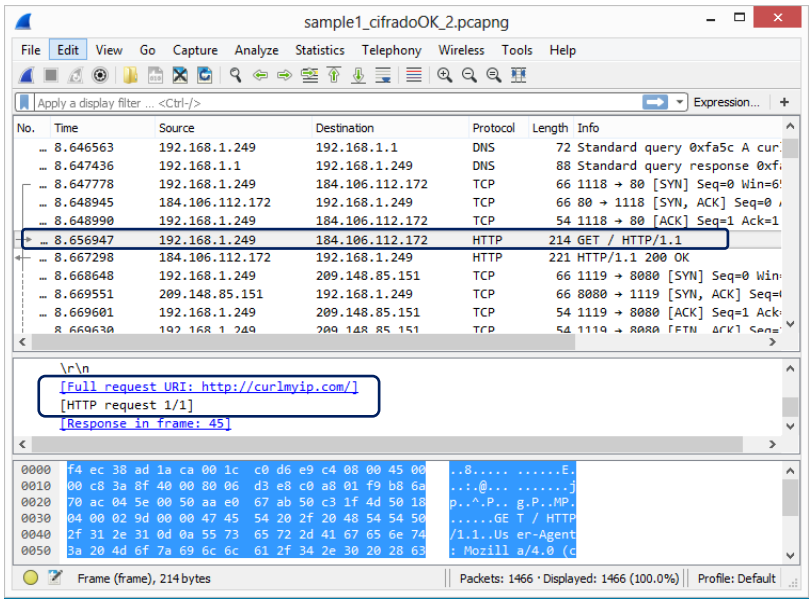
Antes de que el *malware* establezca cualquier comunicación con un servidor remoto, primero intenta adquirir la dirección IP externa del computador de la víctima. Esta información es obtenida consultando los siguientes enlaces.

1. http://ip-addr.es
2. http://myexternalip.com/raw
3. http://curlmyip.com

Para este caso particular, el *malware* no puede obtener la dirección IP externa de la PC de la víctima debido a que todas las peticiones hacia internet son redirigidas al servidor web local.

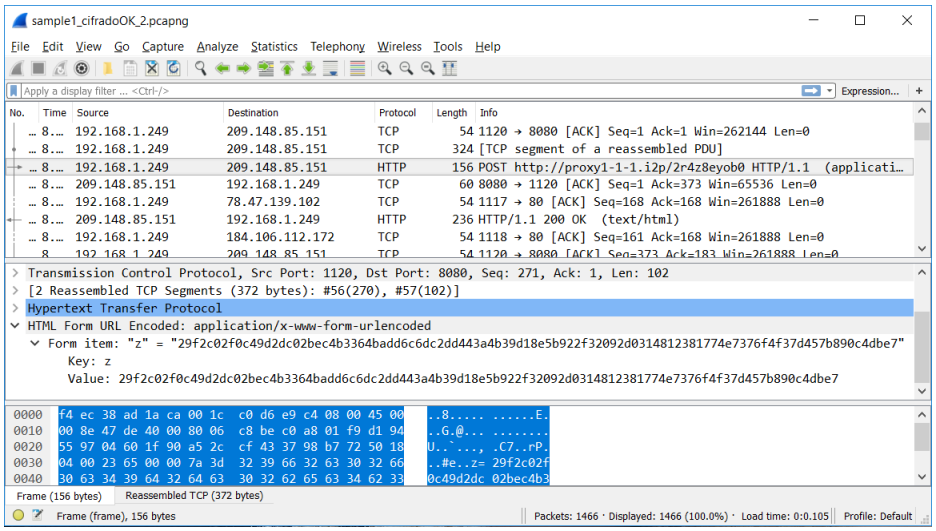
Adicionalmente, se aclara que los datos mostrados desde la imagen 24 hasta la imagen 30, son el resultado de la captura del flujo de tráfico que se genera en la máquina infectada. Por lo tanto, resulta correcto que en las imágenes se visualice como dirección de destino una IP pública, debido a que originalmente el *malware* utiliza una de las direcciones preconfiguradas en su *payload* para tratar de comunicarse con un servidor C&C; pero cuando este paquete, cuyo destino es la red WAN, llega al router secundario (imagen 16) la regla de filtrado lo redirecciona hacia el servidor web local.

Imagen 22 – Consulta IP externa



En este punto de la rutina comienza la comunicación entre el *malware* y el servidor C&C. Toda comunicación será realizada a través de peticiones HTTP *post*, donde la data enviada y recibida es cifrada utilizando el algoritmo RC4. Adicionalmente, por cada petición HTTP, el malware genera una única clave de descifrado que es proveída por medio de una variable POST[36].

Imagen 23 – Comunicación #1: cliente-servidor

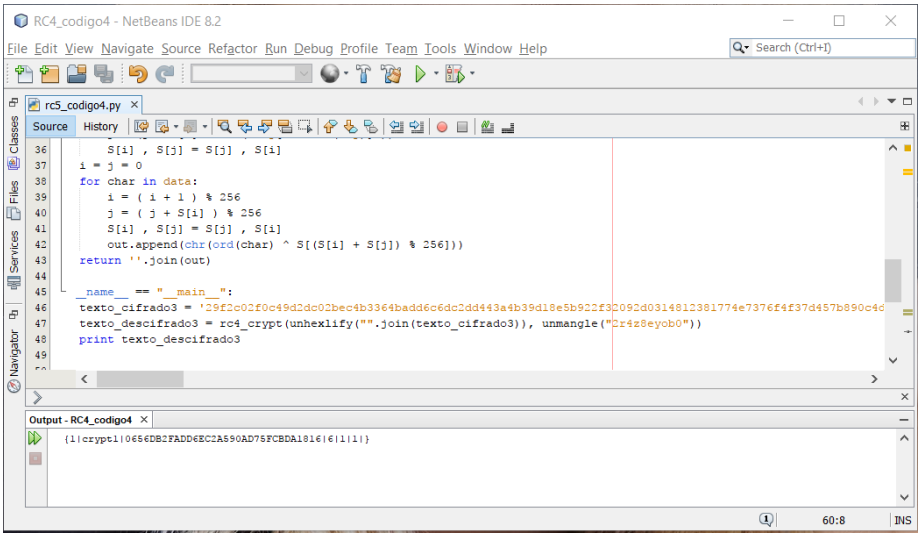


La imagen anterior muestra el inicio de la comunicación entre el *malware* y el servidor C&C. En cada *request*, el *malware* envía dos variables aleatorias. La primera, enviada a través de la URL, corresponde a la clave RC4 usada para cifrar el contenido del mensaje. La segunda, la variable "z" contiene las instrucciones enviadas al servidor.

Una vez que el servidor recibe el *request* debe obtener la clave RC4 para descifrar el contenido de la segunda variable recibida. Para esto, primero reordena los caracteres de la clave RC4 según su valor en código ASCII [anexo B]. El resultado obtenido es usado por el servidor remoto para descifrar el contenido del mensaje [anexo C].

[36] Es un método de *request* compatible con HTTP. Proporciona información adicional del cliente al servidor.

Imagen 24 – Mensaje descifrado comunicación #1



rc4_crypt(**unhexlify**("29f2c02f0c49d2dc02bec4b3364badd6c6dc2dd443a4b39d18e5b922f32092d0314812381774e7376f4f37d457b890c4dbe7"), **unmangle**("2r4z8eyob0"))

Mensaje **descifrado**:

{1|crypt1|0656DB2FADD6EC2A590AD75FCBDA1816|6|1|1|}

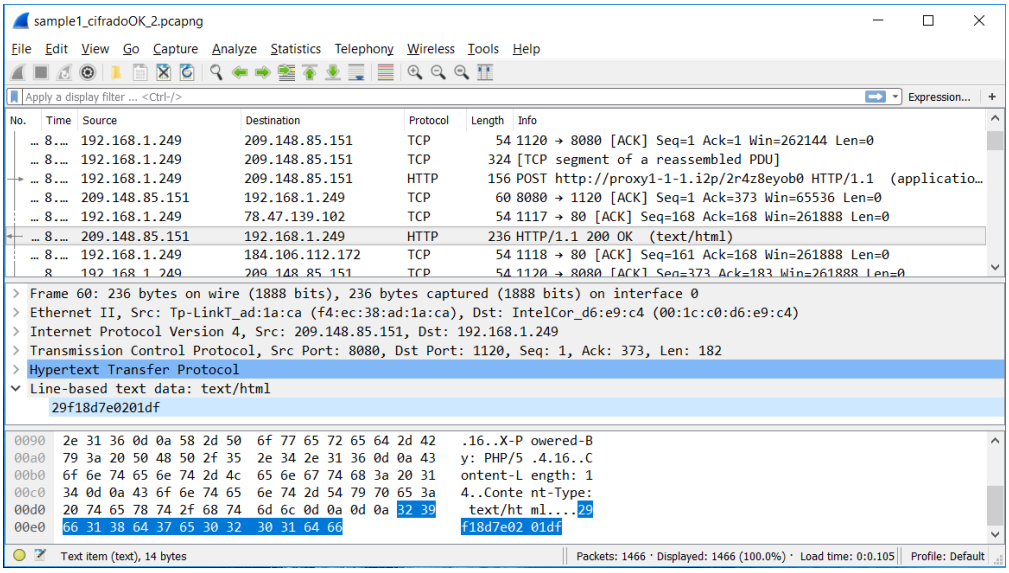
La función anterior retorna una lista de variables con los que el servidor identifica las características de la máquina de la víctima y la instrucción solicitada por el malware.

Tabla 6 – Variables descifradas

Descripción	Valor
Comando enviado	1
Código campaña	crypt1
Identificador MD5	0656DB2FADD6EC2A590AD75FCBDA1816
Versión SO	6
Arquitectura CPU	1
Privilegios del usuario	1
Dirección IP externa	N/A

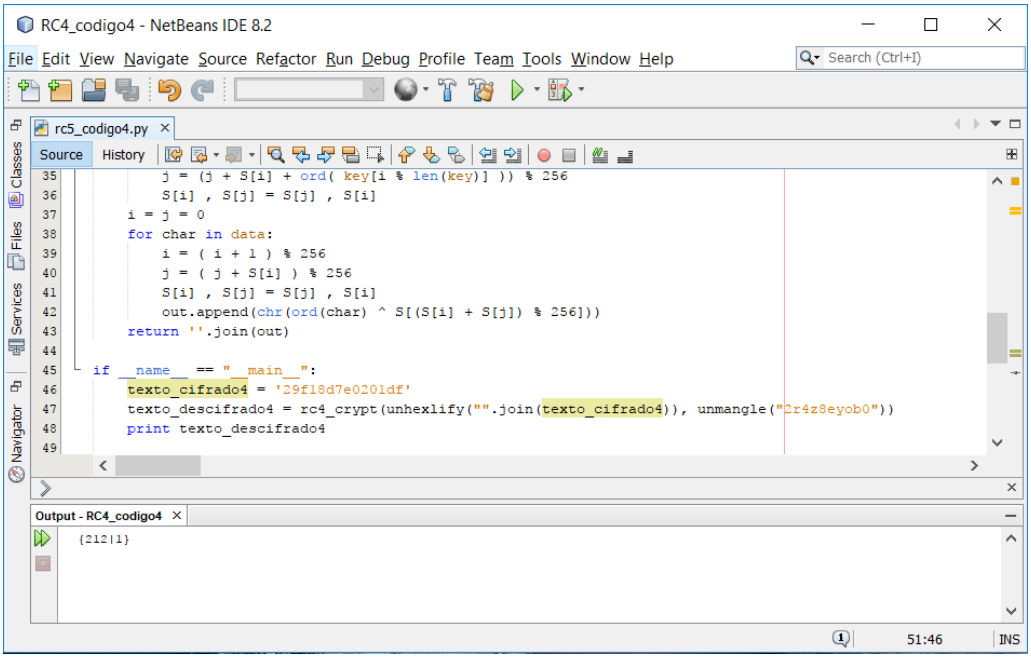
Ante esta petición, el servidor contesta con un mensaje cifrado con la clave RC4 previamente recibida.

Imagen 25 – Comunicación #2: servidor–cliente



El *malware* recibe la respuesta cifrada del servidor: 29f18d7e0201df, y la descifra con la clave RC4 obtenida con anterioridad.

Imagen 26 – Mensaje descifrado comunicación #2



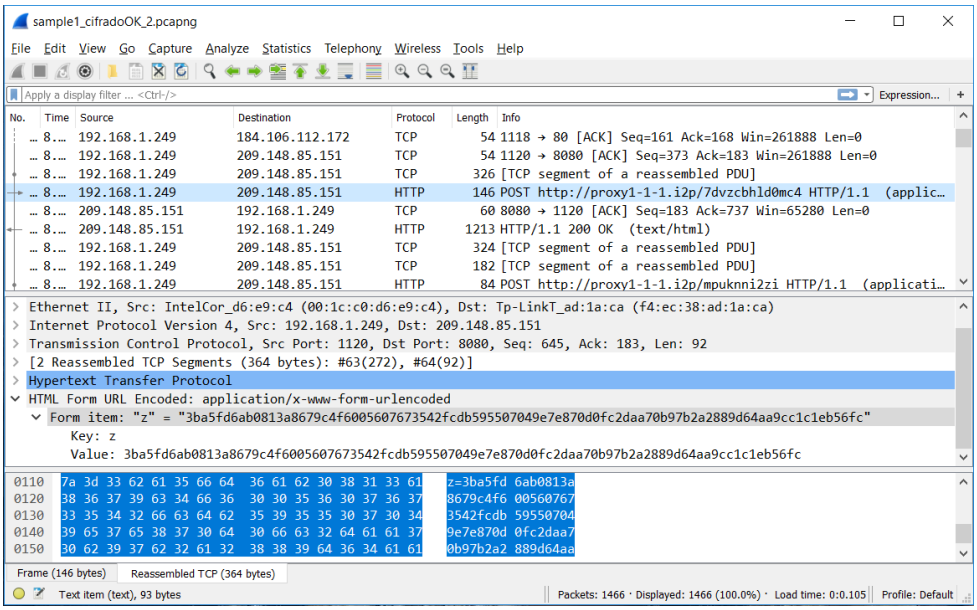
rc4_crypt(**unhexlify**(“29f18d7e0201df”), **unmangle**(“2r4z8eyob0”))

Mensaje **descifrado**:

{212|1}

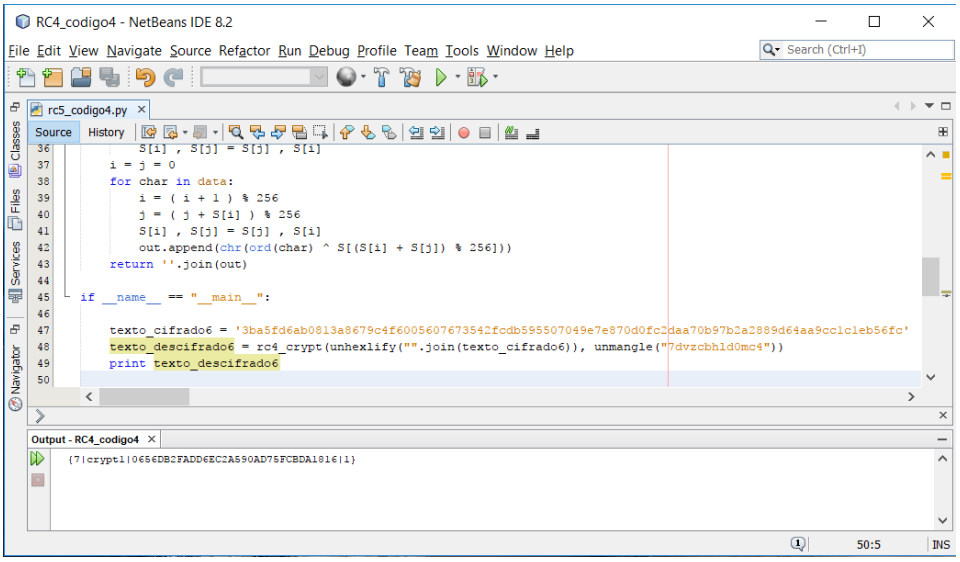
Del mensaje descifrado obtenido, el primer parámetro corresponde a la cantidad de segundos máximos que tiene el malware para completar el resto de sus acciones. Si este valor es mayor de 1000 o no es proveído, CW3 usará un tiempo por defecto de 120 segundos. Esta condición es establecida por los autores del *malware* como estrategia para cuando exista un *release* del *malware* o una actualización de la lista de sus servidores C&C. En el caso de que el *malware* no logre finalizar su rutina en el tiempo especificado, CW3 envía una nueva solicitud al servidor. Si la condición anterior no se cumple, el *malware* continúa su comunicación al enviar un nuevo *request*, solicitando la clave pública RSA.

Imagen 27 – Comunicación #3: cliente-servidor



Esta comunicación puede ser descifrada utilizando la función *rc4_crypt(texto_cifrado, claveRC4)*.

Imagen 28 – Mensaje descifrado comunicación #3

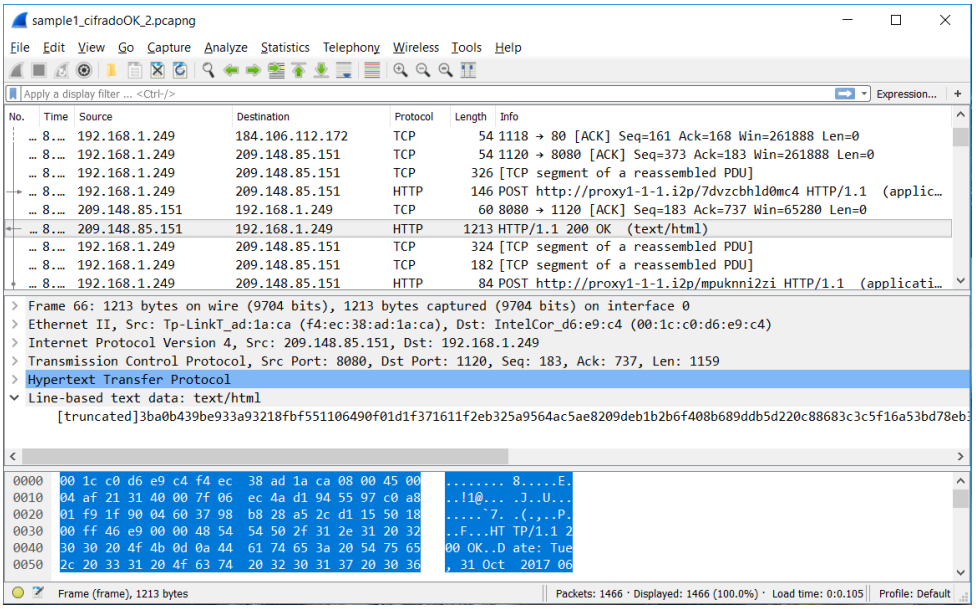


rc4_crypt(**unhexlify**(“3ba5fd6ab0813a8679c4f6005607673542fcdb595507049e7e870d0fc2daa70b97b2a2889d64aa9cc1c1eb56fc”),**unmangle**(“7dvzcbhld0mc4”))

Mensaje **descifrado**:
{7|crypt1|0656DB2FADD6EC2A590AD75FCBDA1816|1}

El servidor *Command and Control* recibe esta petición e interpreta el comando 7 como una solicitud de clave RSA, por lo que envía un *response* con la clave RSA a la maquina infectada.

Imagen 29 – Comunicación #4: servidor–cliente



rc4_crypt(**unhexlify**("3ba0b439be933a93218fbf551106490f01d1f371611
f2eb325a9564ac5ae8209deb1b2b6f408b689ddb5d220c88683c3c5f16a53bd7
8eb300416c9d8ae3bb5062b77a87e06ea912c5d194d0451ccc67a5b4735b510
1aa67f563455871964571cf051cc44786088df691cf6962e5142a335973387a51
cad22274bc40b1a9fc9225a151ec28169cc6b73da30411c5d76556fe65faa2f06
f5b614f62f767254d15d4b4207fc00b4c593dd3396f250b51d76c6842fb049a09
0a80d65a48ee50686c5bac0af491523d4216cae2d6d12d763345b192e37eb5f
aa1591e3158cfc434e58664a14dd5fec2b3252357b0fbe7ec44b5fc089f2df56a1
b6f49a5c8bbfda886a847b2dc499d0677f2c21148e2266628e185ae3aa0878f0
2962d60b90b4b2217fda492ecae02ac684535eab4960a5dce5eea6f05846a1d
9e915f69764fb7d658cd65938653806195c254bb03ddbcec39fcb1b03dfff934d2
7e57f4ce8b4b4294b7edf9808efe4ceffb767bfacd9010a3c8c8d13a356827d85d
fa3aba40b53b1e359cbcf59a891b953d24ac4ac6fa957168d49a3fe9635cf22a3
dca0b711312aabf7839ed764bf83d36b5ac4a62ca076fe62503a0396ef28545f8
5aaa1b563161195105ca87c58d72700647da072d11bb9e91b460a9635f32601
822579272ac6d1b4b480c6c64d6b3f63953e55a248c1"),**unmangle**("7dvzcb
hld0mc4"))

Mensaje **descifrado**:

**{250|kpai7ycr7jxqkilp.onion|75a5|ES|-----BEGIN PUBLIC KEY----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAq6ZEGywe2w
C83CErmVhBgn89wi2lq8rQyYZZGCUyQr5cQirN32HX5n6MNcJDjB8uINSo
saHHGdCPUeOoetx9IM3TiXZwwSnteGR+gBry/C7dr3JSAWrnLE+TkeO6m
rQ8yUtaZ3ue7XSyvV457jLzLg1noHcLL/RXNKsP0MmbZW1yHvRuIsuy5X
W6clQqNMMmfAAT8y+UwePL7M0YG86qSP9QQFB3B+FdlRJ/8VlN3Sva1
JeXJbxGcbmowwtJEJVZtAuf9c7sDv5Kt1tlH8Z2VXjg4P2Dw3KUg/7pcfL18
SAEfsRBlONFvLNcrVdh0/W5aVh6/9djcnMiCV61CYkS7wlDAQAB-----END
PUBLIC KEY-----}|1}**

El mensaje descifrado es interpretado por el *malware* de acuerdo con la siguiente estructura:

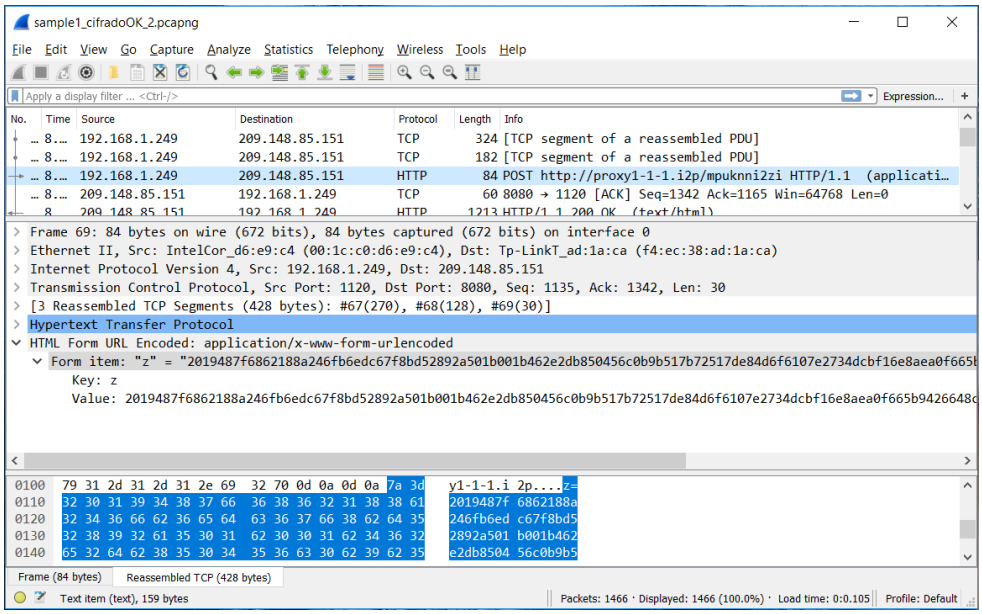
{[SleepTimer][TORSite][URI][VictimCountryCode][RSApublicKey]}

Donde, la dirección TOR y el identificador URI son usados por el *malware* para generar una dirección *onion* que permita a la víctima la realización de un pago a cambio de la clave que descifre sus archivos.

Cryptowall, maneja una "lista negra" de códigos de países. Estos códigos son usados para identificar a aquellos países que no deben ser infectados por el *malware*. A partir de la dirección IP externa de la víctima, el servidor retorna la representación CRC32 del código del país al cual pertenece y lo compara con los códigos de la *blacklist*. En caso de encontrar similitud, el *malware* procede a desinstalarse.

Además de los datos anteriores, el *malware* recibe la clave pública solicitada. A este dato, CW3 le aplica el algoritmo MD5 y se lo envía al servidor.

Imagen 30 – Comunicación #5: cliente–servidor



rc4_crypt(**unhexlify**("mpuknni2zi"),**unmangle**("2019487f6862188a246fb6edc67f8bd52892a501b001b462e2db850456c0b9b517b72517de84d6f6107e2734dcbf16e8aea0f665b9426648cada3e6dd4cada204418b727c1847d1362c498bc3964"))

Mensaje **descifrado**:

{7|crypt1|0656DB2FADD6EC2A590AD75FCBDA1816|2|2CB0852A68AD6AB66F1589A44FC7A879}

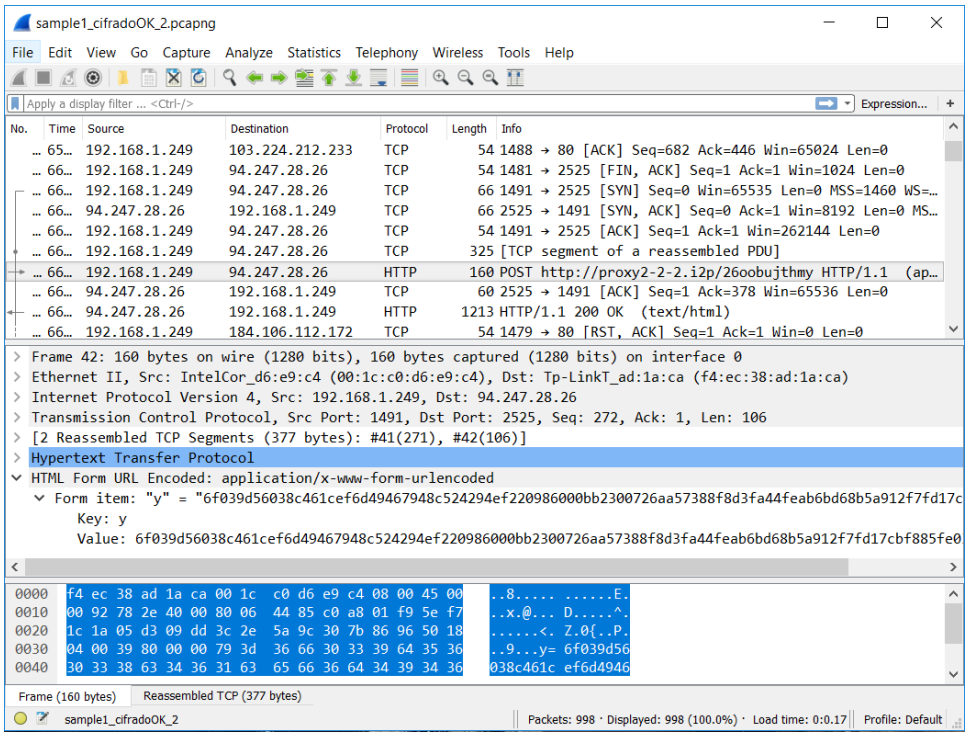
Tabla 7 – Datos descifrados comunicación #5

Datos cifrados	2019487f6862188a246fb6edc67f8bd52892a501b001b462e2db850456c0b9b517b72517de84d6f6107e2734dcbf16e8aea0f665b9426648cada3e6dd4cada204418b727c1847d1362c498bc3964				
Datos descifrados	{7|crypt1|0656DB2FADD6EC2A590AD75FCBDA1816|2|2CB0852A68AD6AB66F1589A44FC7A879}				
Datos separados	Comando	Código campaña	Identificador MD5	Sub comando	MD5 RSA Key
	7	Crypt1	0656DB2FADD6EC2A590AD75FCBDA1816	2	2CB0852A68AD6AB66F1589A44FC7A879

Una vez que el servidor recibe el *hash* de la clave pública RSA puede confirmar que existió integridad en la comunicación. Por lo tanto, procede a enviar un *response* con la *data* de una imagen PNG que contiene las instrucciones del procedimiento a seguir para efectuar el pago por medio de una dirección Bitcoin y descifrar los archivos.

En este punto de la rutina, empieza el cifrado de los archivos en la máquina de la víctima. Tanto la extensión del archivo como la cantidad de archivos cifrados dependerán de las características particulares del *ransomware*. Cifrados los archivos, el *malware* realiza una última comunicación con el servidor, informando la cantidad de archivos cifrados e inicia su desinstalación.

Imagen 31 – Comunicación #6: servidor-cliente



rc4_crypt(**unhexlify**("26oobujthmy"),**unmangle**("6f039d56038c461cef6d49467948c524294ef220986000bb2300726aa57388f8d3fa44feab6bd68b5a912f7fd17cbf885fe03875"))

Mensaje **descifrado**:

{7|crypt1|0656DB2FADD6EC2A590AD75FCBDA1816|3|all=41}

Finalmente, el servidor contesta con un acuse de recibo y termina la comunicación entre cliente y servidor.

7.3. CIFRADO DE ARCHIVOS

El proceso de cifrado de archivos inicia en el instante en que el *malware* solicita al servidor una clave pública RSA (imagen 30). Posteriormente, *Cryptowall* crea una clave de registro donde almacenará la lista de nombres y directorios de cada archivo cifrado cuando concluya la rutina de cifrado.

Tabla 8 - Clave de registro para archivos cifrados

Formato clave registro	HKCU\Software\[MD5-Key]\[Last-16-characters]
Clave de registro	HKCU\Software\0656DB2FADD6EC2A590AD75FCBDA1816 \011556789AABCDDF\

De la tabla descrita recientemente, el término [MD5-Key] corresponde al *hash* identificador de la computadora de la víctima y el término [Last-16-characters] hace referencia a los últimos 16 caracteres del [MD5-Key] reordenados ascendentemente.

A continuación, el *malware* procede a crear las siguientes claves de registro.

1. HKU\[SID]\Software\[MD5-Key]\[2-char]k
2. HKU\[SID]\Software\[MD5-Key]\[3-char]u
3. HKU\[SID]\Software\[MD5-Key]\[4-char]r
4. HKU\[SID]\Software\[MD5-Key]\[5-char]v

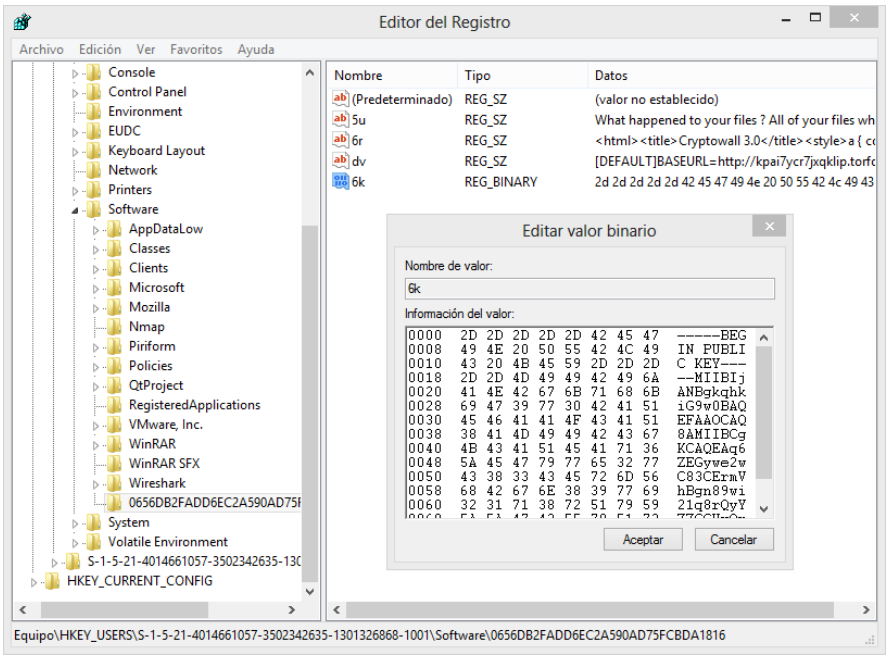
En las claves de registro mencionadas, el término [SID] hace referencia al identificador de seguridad del usuario dueño de la instancia actual del *malware*, [MD5-key] representa el identificador MD5 de la computadora de la víctima y, por último, el término [X-char] corresponde a un carácter minúsculo específico del [MD5 Key].

Tabla 9 - Claves de registro para instrucciones de cifrado

MD5 key: 0**656D**B2FADD6EC2A590AD75FCBDA1816	
Clave de registro 1	HKU\S-1-5-21-4014661057-3502342635-1301326868-1001\Software\0656DB2FADD6EC2A590AD75FCBDA1816**6**k
Clave de registro 2	HKU\S-1-5-21-4014661057-3502342635-1301326868-1001\Software\0656DB2FADD6EC2A590AD75FCBDA1816**5**u
Clave de registro 3	HKU\S-1-5-21-4014661057-3502342635-1301326868-1001\Software\0656DB2FADD6EC2A590AD75FCBDA1816**6**r
Clave de registro 4	HKU\S-1-5-21-4014661057-3502342635-1301326868-1001\Software\0656DB2FADD6EC2A590AD75FCBDA1816**d**v

Cada una de las claves mencionadas en la tabla 9 almacenará un valor de registro. En la primera clave de registro se almacenará la clave pública RSA enviada por el servidor C&C. Las demás, contendrán las instrucciones de descifrado de los archivos en diferentes formatos: TXT, HTML[37] y a través de una URL.

Imagen 32 - Claves de registro creadas



Después que la clave pública RSA ha sido almacenada en su respectiva clave de registro, *Cryptowall*, procede a importarla través de una llamada a la función *CryptImportPublicKeyInfo* del API *CryptoAPI.* A esta clave pública RSA se le aplica el *hash* MD5 y se lo envía al servidor C&C para verificar su integridad. Una vez que se haya verificado este dato, el

[37] *HyperText Markup Language,* lenguaje de marcado que se utiliza para el desarrollo de páginas de internet.

malware utiliza la función *GetLogicalDrivesStringsW* para obtener una lista de las unidades lógicas del equipo a excepción del CD-ROM. Por cada unidad lógica identificada, *Cryptowall* crea un *thread* que se encarga de cifrar todos los archivos del directorio raíz respectivo. El proceso de identificación de los archivos se efectúa a través de llamadas a las funciones *FindFirstFileW* y *FindNextFileW.*

Previo al cifrado, *Cryptowall* verifica que cada archivo cumpla las siguientes condiciones:

1. Que la representación CRC32 del nombre de la carpeta que contiene el archivo identificado no coincida con ningún elemento de la *blacklist* de directorios del malware.
2. Que la representación CRC32 del nombre del archivo identificado no concuerde con ningún elemento de la *blacklist* de archivos del malware.
3. Que la representación CRC32 de la extensión del archivo identificado se encuentre en la *whitelist* de extensiones de archivos del malware.

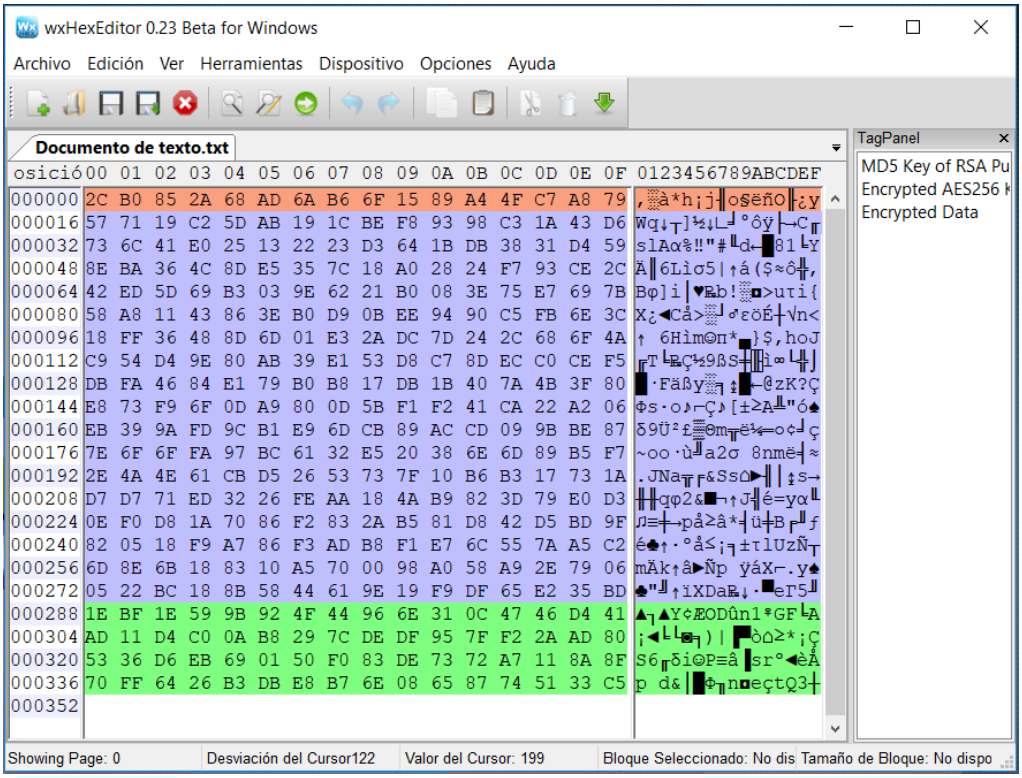
En caso de no cumplirse alguna de estas condiciones, el *malware* procederá a ignorar el archivo. De lo contrario, se modifican los atributos del archivo identificado a través de una llamada a la función *SetFileAttributes*. Esta función recibe dos parámetros: el *lpFileName* que es el nombre del archivo cuyos atributos se desean modificar y el *DwFileAttributes* que es una estructura que contiene una lista de atributos a modificar. Entre esos campos se encuentra el atributo *File_Attribute_Archive* que se utiliza para identificar con un bit a aquellos archivos que se van a remover o respaldar.

Por cada archivo identificado, *Cryptowall* realiza el siguiente procedimiento:

1. Crea un identificador de archivo que permita accederlo y asignarle el atributo FILE_ALL_ACCESS. Este atributo le concede todos los privilegios posibles de acceso.

2. Obtiene el *timestamp* [38] del archivo original. Este dato será usado para sobrescribir la información de *timestamp* del archivo después que el proceso de cifrado finalice.
3. Crea un nuevo archivo con el mismo nombre y extensión del archivo original y le adiciona una extensión aleatoria de tres caracteres. A este nuevo archivo, se le agrega al principio, el *hash* MD5 de la clave pública RSA seguido de la clave AES256 cifrada con la clave pública RSA, y al final, el contenido cifrado del archivo original.

Imagen 33 - Visualización hexadecimal archivo cifrado



Para cifrar los archivos de la víctima, CW3 genera una clave AES de 256 bytes a través de una llamada a la función *CryptGenKey* presente en la librería *CryptoAPI.* La elección de un algoritmo simétrico en lugar de un algoritmo asimétrico se fundamenta en que este último requiere mayor potencia de procesamiento para efectuar las operaciones de cifrado.

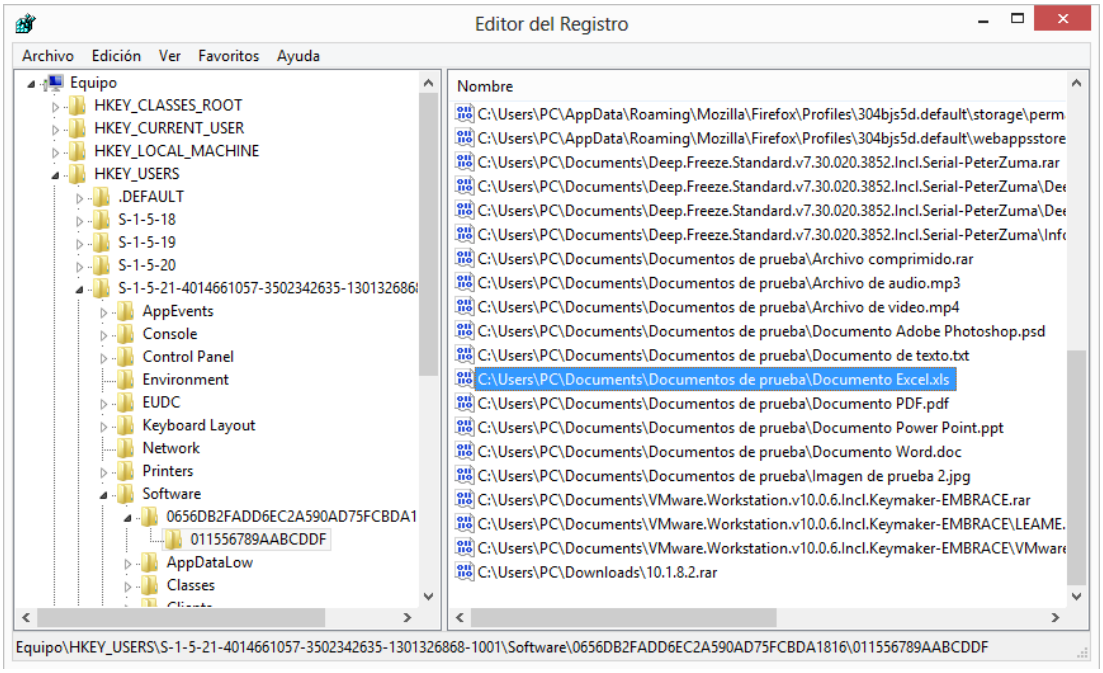
Finalizado el cifrado de archivos, la clave AES256 se cifra usando la clave pública RSA provista por el servidor C&C. Seguidamente, CW3

[38] Secuencia de caracteres que denota la hora y fecha en la que ocurrió un evento.

reemplaza cada archivo original por su equivalente archivo cifrado a través de la función *MoveFileExW*. Pero antes, sustituye el *timestamp* del archivo cifrado por el *timestamp* del archivo original a través de una llamada a la función *SetFileTime.*

Como se menciona al inicio de esta sección, *Cryptowall* crea una clave de registro para llevar un control de la cantidad de archivos cifrados por unidad de volumen. Es decir que, a medida que se cifra un archivo se crea un valor de cadena que identifica el directorio donde se almacena cada archivo cifrado.

Tabla 10 - Archivos cifrados



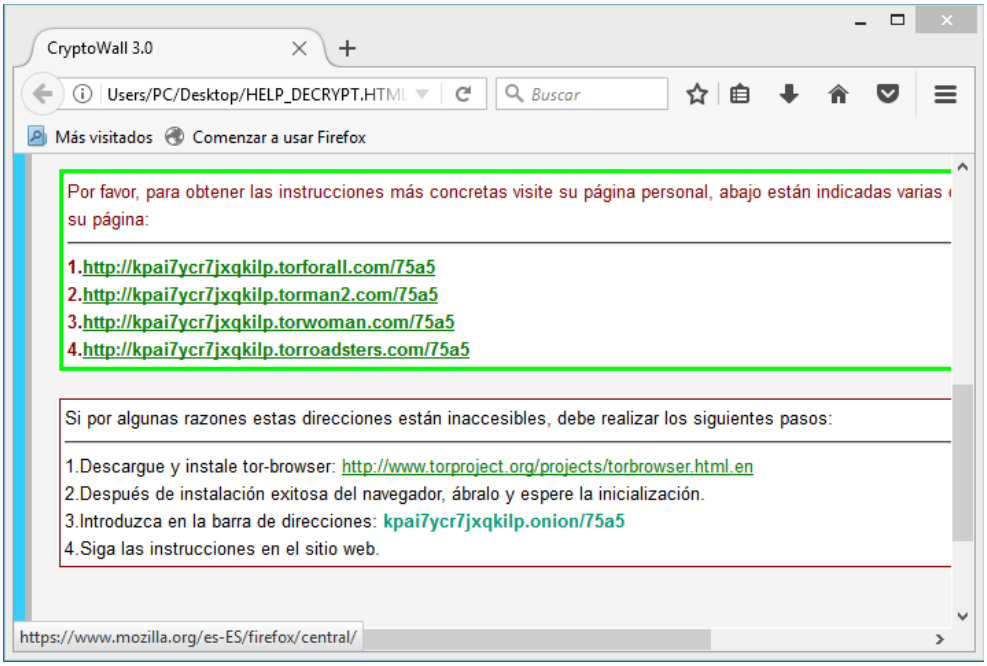
Por cada directorio que contenga archivos cifrados, a excepción del *Desktop*, *Cryptowall* crea cuatro archivos con instrucciones de descifrado: *help_decrypt.txt, help_decrypt.png, help_decrypt.html* y *help_decrypt.url.* La excepción mencionada previene que el usuario visualice archivos extraños dentro su área de trabajo antes que finalice el proceso de cifrado. Cuando la rutina finaliza, CW3 crea y abre los archivos a través de la función *ShellExecuteW.*

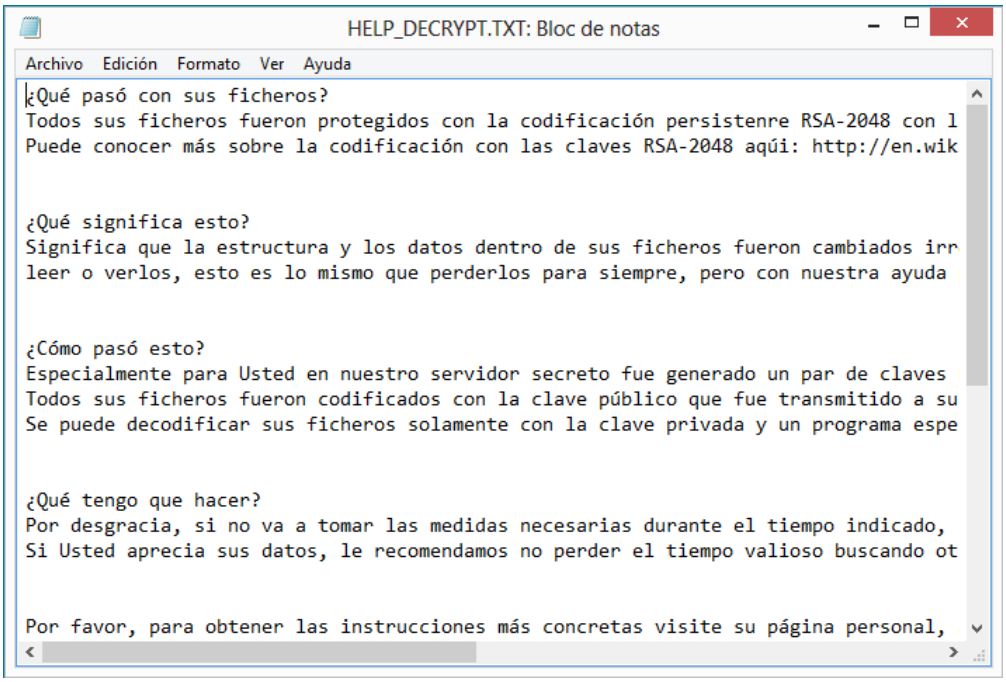
En los archivos de instrucciones creados, también se provee una dirección *onion* de la página web personal de la víctima, creada automáticamente por el *malware*; aquí se detalla el tiempo disponible para

efectuar el pago, la cantidad de archivos cifrados, el valor a pagar y la dirección *bitcoin* para la transferencia. El hecho de elegir la moneda *bitcoin* como medio de pago tiene por objetivo poder recibir grandes cantidades de dinero sin exponer su identidad y sin que exista la posibilidad de una impugnación del pago, como podría suceder con las transferencias tradicionales o con el uso de PayPal. Otra de las razones para la elección del *bitcoin* como medio de pago, radica en la fluctuación constante de la moneda, que, de seguir con la tendencia de los últimos años, significaría grandes inversiones a futuro.

Para dar por finalizado la rutina del *malware*, se consulta la cantidad de valores que tiene la clave de registro descrita en la tabla 8, a través de una llamada a la función *ZwQueryKey0*. El retorno de esta función equivale a la cantidad total de archivos cifrados por el *malware*, información que se notifica al servidor a través de un *request HTTP*. Para evitar el análisis posterior de las muestras, CW3 elimina todas las claves de registro creadas y procede a desinstalarse.

Imagen 34 – Instrucciones de descifrado





8. CONCLUSIONES

- El análisis de *ransomware* trae consigo un conjunto de dificultades previas. El acceso a una muestra actualizada y funcional resulta algo complejo de obtener para aquellas personas nuevas en el campo del análisis de *malware*. Generalmente, los investigadores que poseen muestras válidas son reacios a compartir el código malicioso, solicitando evidencias de experiencia en el análisis de *malware,* para así disminuir el riesgo de propagación. Otra de las complicaciones encontradas, previo al análisis de CW3, recae en la comunicación del *malware* con el servidor C&C. Si bien es cierto que *Cryptowall* dentro de su código almacena una lista de direcciones URL de posibles servidores; con el tiempo, todos estos servidores son desactivados por decisión propia de los autores o por exigencia de alguna autoridad. Con base a lo expuesto, por lo general resulta necesario implementar nuestro propio servidor web, de tal forma, que haga las funciones de un servidor *Command and Control*. Existen dos formas de implementarlo: dentro de la red local o adquiriendo un alojamiento web. Ambas alternativas requieren habilitar la característica *port-forwarding* en el router principal de la red local y crear una regla que

establezca la redirección de paquetes hacia el falso servidor. Esta característica está ausente en la mayoría de los routers domésticos, en cuyo caso se requiere la instalación de un *firmware* compatible *(OpenWRT).*

- *Cryptowall* es un tipo de *malware* que mejora constantemente. En sus primeras versiones*,* el proceso del cifrado de archivos se efectuaba solamente con criptografía asimétrica (RSA). Con la aparición de CW3, el proceso de cifrado de archivos combina algoritmos simétricos y asimétricos. Esto significa que además de generar un par de claves RSA, el malware crea una clave simétrica utilizando el algoritmo AES. A diferencia de la versión anterior, el cifrado de archivos ya no se efectúa con una clave pública RSA, sino que utiliza criptografía simétrica. La clave pública RSA es solamente usada para cifrar la clave AES luego de que finalice la rutina. La razón de este cambio se debe a que el cifrado simétrico es mucho más rápido y requiere menos poder computacional que un algoritmo de cifrado asimétrico. Considerando que a medida que surgen nuevas versiones del *ransomware*, aumenta la lista de tipos de archivos a cifrar, resulta necesario seleccionar un algoritmo que disminuya considerablemente el tiempo de cifrado.

- Desde sus inicios, *Cryptowall* es un tipo de *malware* bien diseñado. Además de combinar eficientemente la criptografía simétrica y asimétrica para el cifrado de archivos también utiliza un conjunto de técnicas para garantizar el éxito de la rutina. Entre las técnicas más utilizadas se encuentra la persistencia del *malware.* Para lograrlo, *Cryptowall* modifica ciertas claves del registro de Windows con la finalidad de que el *malware* trate de instalarse en cada inicio de sesión a pesar de que el usuario inicie en modo a prueba de fallos. Otra de las características de *Cryptowall* es que antes de ejecutar la rutina de cifrado, le solicita al servidor una actualización del *malware* o de la lista de servidores. A menudo, los investigadores de *malware* detectan servidores maliciosos en la red, por lo que es común que los

autores de CW3 deban cambiar constantemente la lista de servidores C&C o deban actualizar el *malware* para corregir errores detectados. Como medida de evasión de análisis, desde la versión 2 de *Cryptowall* se implementan técnicas defensivas para la detección de ejecución del *malware* en máquinas virtuales, como por ejemplo la detección de claves de registros exclusivas de sistemas virtuales, detección de nombres de servicios, entre otras.

- Es acertado afirmar que el *ransomware* es la amenaza más importante de los últimos tiempos. No solo es el causante de grandes pérdidas económicas a usuarios domésticos y corporativos, sino también es el responsable del deterioro de la imagen corporativa de reconocidas empresas a nivel mundial. Cada cierto tiempo surgen nuevas versiones de *Cryptowall*, lo que conduce a pensar que estaríamos frente a uno de los *malwares* más lucrativos de la historia. Estudios realizados por la *Cyber Threat Alliance* reportaron daños de aproximadamente $325 millones de dólares, causados en mayor porcentaje a usuarios domésticos de países más desarrollados. Esta segregación de posibles víctimas se debe estrictamente a dos factores. El primero, es el resultado del objetivo principal del *malware:* obtener beneficios económicos a cambio de proveer una clave de descifrado. Una víctima de un país desarrollado posee más poder adquisitivo y por lo tanto es más propenso a efectuar el pago. El segundo factor, se debe a que los usuarios domésticos, generalmente no acostumbran a tener una política de respaldos de archivos; a diferencia de las organizaciones que incluso pueden llegar a contar con un DRP[39], lo que evitaría tener que efectuar el pago para recuperar su información.

[39] Plan de recuperación ante desastres, es un proceso creado con el fin de que una organización pueda reiniciar sus operaciones en caso de un desastre natural o causado por humanos.

9. ANEXOS

9.1. ANEXO A – CÓDIGO FUENTE FALSO SERVIDOR C&C

Autor: Mukul Sabharwal

```
<?php
    /**
     * RC4Crypt 3.2
     *
     * RC4Crypt is a petite library that allows you to use RC4
     * encryption easily in PHP. It's OO and can produce outputs
     * in binary and hex.
     *
     * (C) Copyright 2006 Mukul Sabharwal [http://mjsabby.com]
     *     All Rights Reserved
     *
     * @link http://rc4crypt.devhome.org
     * @author Mukul Sabharwal <mjsabby@gmail.com>
     * @version $Id: class.rc4crypt.php,v 3.2 2006/03/10
     * @copyright Copyright &copy; 2006 Mukul Sabharwal
     * @license http://www.gnu.org/copyleft/gpl.html
     * @package RC4Crypt
     */

    /**
     * RC4 Class
     * @package RC4Crypt
     */
    class rc4crypt {
        /**
         * The symmetric encryption function
         *
         * @param string $pwd Key to encrypt with(can be binary of
         * hex)
         * @param string $data Content to be encrypted
         * @param bool $ispwdHex Key passed is in hexadecimal or not
         * @access public
         * @return string
         */

        public static function encrypt ($pwd, $data, $ispwdHex = 0)
        {
            if ($ispwdHex)
                $pwd = @pack('H*', $pwd); // valid input, please!

            $key[] = '';
            $box[] = '';
            $cipher = '';

            $pwd_length = strlen($pwd);
            $data_length = strlen($data);

            for ($i = 0; $i < 256; $i++)
            {
                $key[$i] = ord($pwd[$i % $pwd_length]);
                $box[$i] = $i;
            }
            for ($j = $i = 0; $i < 256; $i++)
            {
                $j = ($j + $box[$i] + $key[$i]) % 256;
```

```
                $tmp = $box[$i];
                $box[$i] = $box[$j];
                $box[$j] = $tmp;
            }
            for ($a = $j = $i = 0; $i < $data_length; $i++)
            {
                $a = ($a + 1) % 256;
                $j = ($j + $box[$a]) % 256;
                $tmp = $box[$a];
                $box[$a] = $box[$j];
                $box[$j] = $tmp;
                $k = $box[(($box[$a] + $box[$j]) % 256)];
                $cipher .= chr(ord($data[$i]) ^ $k);
            }
            return $cipher;
        }


        public static function decrypt ($pwd, $data, $ispwdHex = 0)
        {
            return rc4crypt::encrypt($pwd, $data, $ispwdHex);
        }
    }
?>

<?php
  function hextobin($hexstr)
  {
    $n = strlen($hexstr);
    $sbin="";
    $i=0;
    while($i<$n)
    {
      $a =substr($hexstr,$i,2);
      $c = pack("H*",$a);
      if ($i==0){$sbin=$c;}
      else {$sbin.=$c;}
      $i+=2;
    }
    return $sbin;
  }
?>

<?php
/**
* Crypt/decrypt strings with RC4 stream cypher algorithm.
*
* @param string $key Key
* @param string $data Encripted/pure data
* @see   http://pt.wikipedia.org/wiki/RC4
* @return string
*/
function rc4($key, $data)
{
  // Store the vectors "S" has calculated
  static $SC;
  // Function to swaps values of the vector "S"
  $swap = create_function('&$v1, &$v2', '
      $v1 = $v1 ^ $v2;
      $v2 = $v1 ^ $v2;
      $v1 = $v1 ^ $v2;
  ');
```

```
  $ikey = crc32($key);
  if (!isset($SC[$ikey])) {
      // Make the vector "S", basead in the key
      $S    = range(0, 255);
      $j    = 0;
      $n    = strlen($key);
      for ($i = 0; $i < 255; $i++) {
          $char  = ord($key{$i % $n});
          $j     = ($j + $S[$i] + $char) % 256;
          $swap($S[$i], $S[$j]);
      }
      $SC[$ikey] = $S;
  } else {
      $S = $SC[$ikey];
  }
  // Crypt/decrypt the data
  $n    = strlen($data);
  $data = str_split($data, 1);
  $i    = $j = 0;
  for ($m = 0; $m < $n; $m++) {
      $i        = ($i + 1) % 256;
      $j        = ($j + $S[$i]) % 256;
      $swap($S[$i], $S[$j]);
      $char     = ord($data[$m]);
      $char     = $S[($S[$i] + $S[$j]) % 256] ^ $char;
      $data[$m] = chr($char);
  }
  return implode('', $data);
}
?>

<?php
    function response() {
      foreach ($_POST as $name => $value) {
      //we get the uri, that is a rc4 key. Its necesary to order
      //bytes to decrypt coming data
      $rc4key_bytearray = unpack('C*',
      substr("$_SERVER[REQUEST_URI]", 22));
      asort($rc4key_bytearray);
      $rc4key_bytearray_str = implode(array_map("chr",
      $rc4key_bytearray));

      //here we have decrypted data coming from ransom
      $cwcommand = rc4crypt::decrypt($rc4key_bytearray_str,
      hextobin($value));

      //if the command starts with { after decryption its surely a
      //cryptowall
      if ($cwcommand[0]=='{')
      {
         if(strpos($cwcommand, "crypt")!=FALSE &&
         $cwcommand[1]=='1') //cryptowall first msg
            {
            //cryptowall msg1 communications example:
            //command:
            //{1|cw200|99DC835DFC77319C2176AB46302136BF|2|1|2|}
            //response:{212|1}
            //echo 'entro4';
            $premsgres = "{212|1}";

            //cryptowall
```

```
            //now, with the key, its necesary to encrypted the
            response and send it to the client
            $temp = bin2hex(rc4crypt::encrypt($rc4key_bytearray_str,
            $premsgres));

            //echo strlen($temp);

            $arrtemp = str_split($temp, 50);

            foreach ($arrtemp as $arrtempN) {
               echo $arrtempN;
            }
        }

       else if(strpos($cwcommand, "crypt")!=FALSE &&
       $cwcommand[1]=='7') //cryptowall second msg
       {

//cryptowall msg2 communications example:
//command:{7|cw200|99DC835DFC77319C2176AB46302136BF|4}
//response:{250|kpai7ycr7jxqkilp.onion|75a5|ES|-----BEGIN
PUBLIC KEY-----
//MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAq6ZEGywe2wC83CErmVhB
//gn89wi2lq8rQyYZZGCUyQr5cQirN32HX5n6MNcJDjB8uINSosaHHGdCPUeOoetx9
//IM3TiXZwwSnteGR+gBry/C7dr3JSAWrnLE+TkeO6mrQ8yUtaZ3ue7XSyvV457jLz
//Lg1noHcLL/RXNKsP0MmbZW1yHvRulsuy5XW6clQqNMMmfAAT8y+UwePL7M0YG86q
//SP9QQFB3B+FdlRJ/8VlN3Sva1JeXJbxGcbmowwtJEJVZtAuf9c7sDv5Kt1tlH8Z2
//VXjg4P2Dw3KUg/7pcfL18SAEfsRBIONFvLNcrVdh0/W5aVh6/9djcnMiCV61CYkS
//7wIDAQAB
//-----END PUBLIC KEY-----}

$msgres = "{250|kpai7ycr7jxqkilp.onion|75a5|ES|-----BEGIN PUBLIC
KEY-----
\r\n"."MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAq6ZEGywe2wC83CErm
VhB\r\n"."gn89wi2lq8rQyYZZGCUyQr5cQirN32HX5n6MNcJDjB8uINSosaHHGdCPUe
Ooetx9\r\n"."IM3TiXZwwSnteGR+gBry/C7dr3JSAWrnLE+TkeO6mrQ8yUtaZ3ue7XS
yvV457jLz\r\n"."Lg1noHcLL/RXNKsP0MmbZW1yHvRulsuy5XW6clQqNMMmfAAT8y+U
wePL7M0YG86q\r\n"."SP9QQFB3B+FdlRJ/8VlN3Sva1JeXJbxGcbmowwtJEJVZtAuf9
c7sDv5Kt1tlH8Z2\r\n"."VXjg4P2Dw3KUg/7pcfL18SAEfsRBIONFvLNcrVdh0/W5aV
h6/9djcnMiCV61CYkS\r\n"."7wIDAQAB\r\n"."-----END PUBLIC KEY-----}";

        //cryptowall
        //now, with the key, its necesary to encrypted the response
        //and send it to the client
        $temp = bin2hex(rc4crypt::encrypt($rc4key_bytearray_str,
        $msgres));

        //echo strlen($temp);

        $arrtemp = str_split($temp, 50);

        foreach ($arrtemp as $arrtempN) {
          echo $arrtempN;
        }
      }
    }
  }
}
?>

<?php response();?>
```

9.2. ANEXO B – CÓDIGO FUENTE PARA ORDENAR CARACTERES

```
from binascii import *

def unmangle(data_string):
      '''
      Takes a string of data and re-arranges it in order. This
      technique is used by CryptoWall v3 to create the actual RC4
      key used for decryption.
      '''
      return ''.join(sorted(list(data_string))).lstrip("\x00")
```

9.3. ANEXO C – CÓDIGO FUENTE PARA DESCIFRAR RC4

```
from binascii import *

def rc4_crypt(data, key):
   S = range(256)
   j = 0
   out = []

   for i in range(256):
      j = (j + S[i] + ord( key[i % len(key)] )) % 256
      S[i] , S[j] = S[j] , S[i]
      i = j = 0

      for char in data:
            i = ( i + 1 ) % 256
            j = ( j + S[i] ) % 256
            S[i] , S[j] = S[j] , S[i]
            out.append(chr(ord(char) ^ S[(S[i] + S[j]) % 256]))
      return ''.join(out)

      if __name__ == "__main__":
            texto_cifrado = "";
            clave_ordenada = "";
            texto_descifrado = "";
            encrypted = rc4_crypt(unhexlify("".join(texto_cifrado)),
            clave_ordenada);
            print texto_descifrado;
```

10. BIBLIOGRAFÍA

[1] Syverson, P., Goldschlag, D., and Reed, M. *Anonymous Connections and Onion Routing.* IEEE. (1997).

[2] Erkkonen, H., and Larsson, J. (s.f.). *Anonymous Networks.* Chalmers University, Computer communication and distributed systems, Maskingränd.

[3] DeFabbia-Kane, S. *Analyzing the Effectiveness of Passive Correlation Attacks on the Tor Anonymity Network.* Tesis, Wesleyan University, Departmental Honors in Computer Science, Connecticut. (2011).

[4] Dingledine, R., Mathewson, N., and Syverson, P., Tor: The Second-Generation Onion Router. (2004)

[5] Panchenko, A., and Renner, J., Path Selection Metrics for Performance-Improved Onion Routing. *Ninth Annual International Symposium on Applications and the Internet*. doi:10.1109/SAINT.2009.26. (2009).

[6] Morain, M., Titov, V., Verbuggen, W. (2005). Componentes TOR. [Figura]. Recuperado de http://ctvr.tcd.ie/undergrad/4ba2.05/group10/index.html

[7] Goldschlag, D., Reed, M., and Syverson, P., *Hiding Routing Information.* Naval Research Laboratory, Center For High Assurance Computer Systems, Washington. (1996).

[8] Fercufer. (2012). Intercambio de células para el establecimiento de circuito en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690389

[9] Dingledine, R., Mathewson, N., and Syverson, P. (2004). TOR Control Cell and Cell Structure. [Figura]. Recuperado de "*Tor: The Second-Generation Onion Router".*

[10] Fercufer. (2012). Célula *create* en intercambio de células para el establecimiento de circuito en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690390

[11] Fercufer. (2012). Célula *created* en intercambio de células para el establecimiento de circuito en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690391

[12] Fercufer. (2012). Célula *relay extend* en intercambio de células para el establecimiento de circuito en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690392

[13] Fercufer. (2012). Célula *relay extended* en intercambio de células para el establecimiento de circuito en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690393

[14] Fercufer. (2012). Envío de información en TOR. [Figura]. Recuperado de https://commons.wikimedia.org/w/index.php?curid=19690585

[15] Nakamoto, S., *Bitcoin: A Peer-to-Peer Electronic Cash System.* (2008)

[16] Oro y finanzas. (2014, Diciembre 30). *¿Qué es un nodo en bitcoin?* Recuperado de: https://www.oroyfinanzas.com/2014/12/que-es-nodo-bitcoin/

[17] ¿Cómo funciona Bitcoin? Direcciones públicas y claves privadas. [Figura]. Recuperado de https://www.queesbitcoin.info/como-utilizar-bitcoin/direcciones-bitcoin/

[18] Ruben, P., *Bitcoin: Identificar nodos mineros.* Tesis, Universidad Oberta de Catalunya, Sabadell. (2015).

[19] Blockchain: Las entrañas de bitcoin, transacciones, criptografía y ASIC's, http://blog.elevenpaths.com/2017/04/blockchain-iv-las-entranas-de-bitcoin.html (consultada el 18/01/2018)

[20] Huang, J. (2017). Bitcoin and Blockchain: The Blockchain Consensus. Recuperado de https://www.linkedin.com/pulse/bitcoin-blockchain-part-v-consensus-junbang-huang/ (consultada el 7/02/2018)

[21] Bitcoin difficulty, https://bitcoin.stackexchange.com/questions/5838/how-is-difficulty-calculated/5840#5840 (consultada el 9/12/2017)

[22] Bitcoin difficulty equation, https://bitcoin.stackexchange.com/questions/54171/a-problem-on-difficulty-equation/54173 (consultada el 9/12/2017)

[23] Explanation of difficulty retargeting. Recuperada de https://github.com/bitcoinbook/bitcoinbook/issues/206 (consultada el 8/12/2017)

[24] Block hashing algorithm. [Figura]. Recuperada de https://en.bitcoin.it/wiki/Block_hashing_algorithm (consultada el 9/12/2017)

[25] Hashcash. [Figura] Recuperada de https://en.bitcoin.it/wiki/Hashcash (consultada el 9/12/2017)

[26] Blockchain: Un modelo distribuido fiable de seguridad ¿La panacea?, http://blog.elevenpaths.com/2017/02/blockchain-ii-un-modelo-distribuido.html (consultada el 12/08/2017)

[27] Learn me a bitcoin: Difficulty, http://learnmeabitcoin.com/guide/difficulty (consultada el 17/4/2017)

[28] Bitcoin Project. (2014). Inputs and Outputs Bitcoin. [Figura]. Recuperada de https://bitcoin.org/en/developer-guide#block-chain-overview

[29] Kangas, E. (2017). Structure of a Bitcoin block. [Figura]. Recuperado de https://luxsci.com/blog/understanding-blockchains-and-bitcoin-technology.html

[30] Blockchain Luxembourg S.A. [Figura]. Recuperado de https://blockchain.info/block/00000000785992b7d5ee630a7aaf650461ca6a5d176d96e8f1d83054132f0bf4

[31] Getting Cryptowall and CryptoDefense working without C&C, https://vallejo.cc/2015/03/10/getting-cryptowall-and-cryptodefense-working-without-cc/ (consultada el 03/09/2017)

[32] Cyber Threat Alliance. Lucrative Ransomware Attacks: Analysis of the Cryptowall Version 3 Threat. (2015)

[33] Devincenzi, J.. *Técnicas y herramientas forenses para la detección de Botnets.* Buenos Aires. (2011)

[34] Anatomy of Cryptowall 3.0 Virus – A look inside ransomware code & tactics, https://www.sentinelone.com/blog/anatomy-of-cryptowall-3-0-a-look-inside-ransomwares-tactics/ (consultada el 22/07/2017)

[35] Infecting myself with Ransomware (Exploring Cryptwall), https://blogs.msmvps.com/mickyj/blog/2014/06/30/infecting-myself-with-ransomware-exploring-cryptowall/ (consultada el 03/08/2017)

[36] Kotov, V., and Mantej, R., Understanding Crypto-Ransomware: In-Depth Analysis of the Most Popular. (2014).

[37] A glimpse inside CryptoWall 3.0, https://blog.brillantit.com/cryptowall-3-0-traffic-analysis/ (consultada el 17/07/2017)

[38] Guess who's back again? Ransomware Cryptowall version 3.0, https://malware.dontneedcoffee.com/2015/01/guess-whos-back-again-cryptowall-30.html?q=cryptowall&view=classic (consultada el 5/06/2017)

[39] Ocultando entornos virtuales a malware y atacantes., http://www.hackplayers.com/2012/11/ocultando-entornos-virtuales-malware-y.html (consultada el 08/01/2017)

[40] Cryptowall Ransomware Threat analysis, https://www.secureworks.com/research/cryptowall-ransomware (consutada el 03/01/2018)

[41] Resume Malspam sending Cryptowall 3.0, http://www.malware-traffic-analysis.net/2015/06/04/index.html (consultada el 10/012/2018)