De la práctica a la teoría de seguridad a través de retos

Juan A. Pereira Varela
Dpto. Lenguajes y Sistemas Informáticos
Universidad del País Vasco
Pº Manuel de Lardizabal, 1. 20018
juanan.pereira@ehu.es

Resumen

Tradicionalmente, la enseñanza de los conceptos teóricos de seguridad informática suele preceder a las pruebas prácticas. Por otra parte, estas prácticas, debido probablemente a la falta de ejemplos reales (pocas empresas desean publicitar los fallos de seguridad de sus productos) pueden pecar de ser demasiado académicas y artificiales.

Actualmente, para invertir esa situación, podemos aprovecharnos de la base de datos que conforman las miles de aplicaciones de calidad profesional junto a su código fuente ofrecidas bajo licencias libres. Éstas aplicaciones incluyen un histórico documentado de vulnerabilidades y errores de seguridad sufridos en el tiempo, junto con la documentación y el código fuente de los parches aplicados para corregirlas.

Este trabajo muestra las experiencias de trabajar con un entorno de (anti)seguridad basado en versiones antiguas de aplicaciones libres con vulnerabilidades conocidas. Se plantea que los alumnos comiencen cada tema teórico de la asignatura con un reto o desafío: romper la seguridad de una aplicación que se sabe insegura. Se pasa después a exponer las técnicas de ataque, documentar las vulnerabilidades y la forma de parchear el sistema y a analizar cómo se podría haber evitado el error a priori. Finalmente, se construye la aproximación teórica general al problema y su solución.

1. Introducción

A pesar de los esfuerzos en la docencia universitaria y no universitaria de la asignatura de seguridad informática, cada día surgen nuevas vulnerabilidades en la seguridad de los sistemas, muchas de ellas, basadas en patrones conocidos.

Tal y como informa la empresa IBM ISS [3] en el año 2007, se reportaron más de 6.000 vulnerabilidades de seguridad en aplicaciones conocidas.

Por otra parte, los conocimientos prácticos de seguridad de los alumnos de ingeniería informática de la Facultad de este trabajo, son limitados. Por poner un ejemplo, de 60 alumnos (ingeniería técnica y superior) que cursaban la asignatura de Seguridad Informática[1], sólo 1 conocía los conceptos básicos de técnicas de ataque web como SQL Injection[2] o Cross Site Scripting, cuando muchos de ellos ya habían programado aplicaciones web, incluso algunos de forma profesional.

[1] Facultad de Informática de San Sebastián, cursos 06/07 y 07/08

[2] Tipo de ataque contra webs soportadas por una base de datos en el que el atacante ejecuta código SQL no autorizado valiéndose de vulnerabilidades debidas a código inseguro en el servidor.

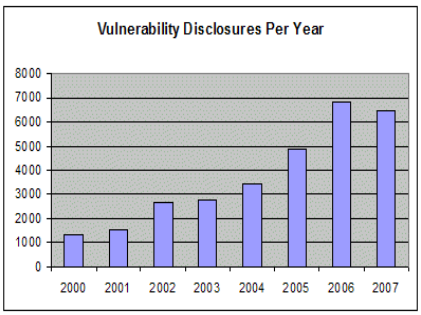

Figura 1. Vulnerabilidades descubiertas anualmente [3]

En el temario anterior al curso 06/07, la asignatura disponía de apartados para hablar de seguridad en entornos web y aplicaciones de escritorio, sin embargo, los ejemplos que se presentaban tenían carácter académico. En concreto nunca se mostraba código ni problemas de seguridad relacionados de aplicaciones reales. Se esperaba que los alumnos fueran capaces de ver el concepto teórico (bien marcado y recortado en un ejemplo de diapositiva) y acto seguido (al terminar la carrera) se pusieran a trabajar con aplicaciones reales de cientos de miles de líneas, escritas hace tiempo y/o con poca o ninguna documentación. Con ese bagaje se espera que el alumno recién licenciado sepa cómo añadir nuevas funcionalidades o corrija errores en dichas aplicaciones reales, siempre de forma segura, es decir, sin generar vulnerabilidades. Esta aproximación no parece acertada.

2. De la teoría a la práctica

En general, la bibliografía de la asignatura de Seguridad Informática sigue una estructura clásica de mostrar una base teórica de cada tema a tratar y pasar después, en su caso, a proponer ejercicios prácticos. Es una aproximación razonable y como tal, es la que se sigue en muchos de los centros en los que se imparte esta asignatura. Sin embargo, nuestra contribución aboga por cambiar la dirección de esta estrategia, es decir, comenzar planteando un caso práctico, llamativo, que atraiga la atención del alumno, para pasar a continuación a intentar resolverlo con los conocimientos actuales y darse cuenta de la necesidad de la teoría. Comprobar de forma empírica que no es posible resolver el ejercicio hasta obtener las claves teóricas que nos permitan continuar.

3. De la práctica a la teoría: una propuesta de proceso

Siguiendo las directrices que se muestran en este trabajo, la asignatura cambió el planteamiento anterior en el curso 06/07. Se presentaba a los alumnos, desde el primer momento, aplicaciones con fallos de seguridad conocidos, y se planteaba como reto, su estudio, explotación, análisis de las pautas de recuperación y prevención. Para ello, nos basamos en el uso de aplicaciones de software libre de amplio uso, como pueden ser el sistema de creación de blogs Wordpress [9], el sistema de gestión de contenidos TikiWiki [7] o la aplicación para generación de foros online phpBB [6]. El reto planteaba la explotación de un fallo de seguridad conocido en una versión antigua de alguna de las aplicaciones mencionadas (y otras), siguiendo el *proceso* que documentamos a continuación.

3.1. Propuesta de reto

Con una semana o varios días de antelación a la presentación de un nuevo tema o aspecto práctico, se presenta una aplicación con una vulnerabilidad conocida (o un texto cifrado, o un fichero ejecutable susceptible de ser crackeado, etc.) y se

desafía al alumnado a que explote dicho fallo. En caso de ser una aplicación vulnerable, ésta puede estar instalada en una máquina remota preparada a tal efecto o bien en una máquina virtual que se reparte al alumnado (ver 5. Entorno de competición). Aquél que resuelva antes el reto (rompa la seguridad de lo planteado) y exponga pautas de solución, gana el campeonato y obtiene, como recompensa , por ejemplo, *x* décimas de punto para la evaluación final. El resto de alumnos participantes que consiga superar la prueba recibirá también unas décimas, pero en cantidad menor que *x*.

3.2. Búsqueda de recursos *black-hat*

Para resolver el desafío, es necesario que los alumnos realicen una labor de búsqueda en Internet, especialmente de los recursos conocidos como black-hat (páginas de anti-seguridad), donde encontrarán información sobre las vulnerabilidades conocidas de la aplicación propuesta así como scripts para explotar dicho agujero de seguridad (*exploits*).

3.3. Uso de *exploits*

El uso de exploits es trivial, basta con tener conocimientos rudimentarios de informática para poder ejecutarlos y lanzarlos contra la aplicación vulnerable. Sin embargo, para superar el reto, no basta con ejecutar el exploit. Es necesario que el alumno explique por qué y cómo funciona dicho exploit, en qué tipo de vulnerabilidad se basa y cómo corregir dicha vulnerabilidad.

3.4. Análisis supervisado del código fuente

En clase, y con la ayuda de aquellos alumnos que hayan superado el reto, se procede a explicar y a analizar el código de la aplicación vulnerable. Se aprovecha la ocasión para explicar cómo llegar hasta las líneas que contienen el error en la aplicación, partiendo del análisis del exploit. Esta operación no es trivial cuando tratamos con aplicaciones con miles o millones de líneas de código (a la sazón, las que analizamos en este tipo de retos). Hay que aprender a escudriñar esas líneas, deshechar lo que no afecte a la vulnerabilidad bajo estudio y centrarse en un pequeño trozo de código vulnerable.

3.5. Estudio de medidas correctivas

Finalizado el análisis de la vulnerabilidad (sabemos qué y por qué falla), es necesario crear el parche que arregle el problema. Este paso se puede hacer de forma supervisada, mediante el análisis del parche que los responsables de la aplicación publicaron y reescribiendo el código para hacerlo más legible (por ejemplo mediante el añadido de comentarios). Otra forma, cuando el alumnado adquiere destreza o bien cuando el parche es sencillo, consiste en desarrollarlo desde cero y comprobar su correctitud mediante su contraste con el parche original open source.

3.6. Análisis de medidas preventivas

No basta con saber encontrar la vulnerabilidad y corregirla. Es necesario aprender a evitarla de raíz. Para ello, el alumno planteará posibles soluciones preventivas (evitar determinadas construcciones ó funciones de código, filtrar siempre los caracteres que llegan desde el usuario, usar frameworks de seguridad que filtren automáticamente cualquier intento de ataque, etc.) Si hay tiempo, se planteará el análisis de las nuevas versiones de las aplicaciones vulnerables utilizadas para buscar pruebas de que dichas medidas

preventivas han sido efectivamente usadas en las nuevas versiones.

3.7. Generalización y planteamiento de teoría

Tras tratar los puntos anteriores, estaremos en condiciones de intentar generalizar las lecciones aprendidas y proponer una definición teórica del problema analizado. Por ejemplo, si se ha estudiado un ataque tipo SQL Injection en concreto (contra una aplicación y una base de datos en concreto), se generalizará para definir en qué consiste este tipo de ataque y qué pautas se ha de seguir para su corrección (independientemente de la aplicación y BBDD utilizada)

3.8. Documentación del proceso

Siguiendo la filosofía del software libre, se documentará el proceso en un repositorio de documentación compartida, de tal forma que los alumnos de los cursos venideros puedan construír sobre ella, sin partir de cero. Esto permitirá innovar cada año "subiéndose a hombros de gigante".

3.9. Exposición de resultados

Finalmente, y a modo de complemento que permita realizar una mejor valoración del trabajo realizado por cada alumno, se planteará la realización de exposiciones y presentaciones públicas, bien de los temas tratados en clase o bien de nuevos temas, siguiendo el procedimiento explicado en este documento.

4. Dos ejemplos prácticos

A modo de ejemplo de lo hasta ahora expuesto, proponemos dos casos prácticos que se han llevado a cabo durante los cursos 06/07 y 07/08 siguiendo los puntos explicados anteriormente en la asignatura de Seguridad Informática. Expondremos también la forma de configurar el entorno o laboratorio de anti-seguridad en el que trabajaron los alumnos.

4.1. Ataque SQL Injection contra Wordpress

El proceso comienza instalando en una máquina de pruebas una antigua versión del sistema de blogs Wordpress 1.5.1.1 (la aplicación PHP para crear blogs más usada en la actualidad). Esta versión contenía una vulnerabilidad que permitía al atacante inyectar código SQL a discreción, debido a un parseo incorrecto de parámetros [1].

En concreto, el problema consiste en que el parámetro *cat* que llega por la URL se usa en el código sin realizar ninguna comprobación previa de seguridad ni filtro. Por tanto, cualquier usuario malicioso podría inyectar código SQL en ese parámetro desde la URL y obtener el nombre de usuario y el password de administración (realmente obtendrá el nombre del usuario administrador y un resumen criptográfico MD5 del password).

Existen varios exploits en las webs de black-hat aprovechando esta vulnerabilidad. Incluso en las bases de datos públicas de vulnerabilidades se pueden encontrar pruebas de concepto. Usando el exploit de OSVD [5], podemos ver que ejecutando la siguiente URL contra el servidor de prueba

```
http://localhost/wordpress/index.php?cat=999%20UNION%20SELECT%20null,CONCAT(CHAR(58),user_pass,CHAR(58),user_login,CHAR(58)),null,null,null%20FROM%20wp_users/*
```

se conseguiría visualizar en pantalla el resumen MD5 de la contraseña del administrador.

Ése es uno de los objetivos del alumno, encontrar el exploit, probarlo y deducir a qué se debe el problema. Es necesario para ello descargar el código fuente de la versión afectada de Wordpress y analizarlo, con ayuda del exploit.

Descifrando la URL, vemos que se intenta ejecutar lo siguiente:

```
index.php?cat=999 UNION SELECT
null,CONCAT(':',user_pass,':',user_login,':'),null,null,null
FROM wp_users
```

lo que nos da a entender que el error debe situarse en aquella parte del código de Wordpress que parsea la variable *cat* , lo que nos lleva, tras análisis detallado en clase, y ayudándonos del histórico de cambios que ofrece la propia web del producto Wordpress [8], a la línea 103 del fichero *wp-includes/template-functions-category.php* . Vemos que el parámetro $cat_ID se usa inocentemente sin ningún tipo de filtro. Basta con forzar una conversión a tipo de datos entero (int) de ese parámetro para solucionar el problema.

```
$cat_ID = (int) $cat_ID;
```

Esa es precisamente otra de las tareas del alumno, averiguar la forma de resolver el problema para, a continuación, auditar el resto del código en busca de patrones inseguros similares.

Este ejemplo además, nos sirve para introducir el tema sobre ataques contra el algoritmo de resumen criptográfico MD5, dado que a pesar de que alguien consiguiera explotar la vulnerabilidad anterior, lo que obtendría como recompensa por pantalla es 'simplemente' el resumen MD5 del password de administración. El siguiente reto consistiría por tanto en encontrar el password original a partir del resumen MD5, lo cual nos permite enlazar con la teoría de tablas Rainbow [4] y ataques por fuerza bruta.

4.2. Ataque de inclusión de Ficheros ejecutables contra TikiWiki

Tiki CMS/Groupware o TikiWiki es un sistema opensource de gestión de contenidos de índole colaborativa (CMS/Groupware) vía web, fácil de configurar y personalizar.

Se presenta a los alumnos un servidor TikiWiki afectado de la vulnerabilidad CVE-2006-4602, que compromete la versión 1.9.4 de TikiWiki permitiendo a un atacante remoto subir archivos PHP arbitrarios al servidor y ejecutarlos (con los permisos del proceso del servidor web), debido a un problema de seguridad en el fichero jhot.php.

En primer lugar, el alumno debe de buscar un exploit capaz de aprovecharse de la vulnerabilidad anterior, analizarlo y probarlo. En concreto, la web MilW0rm [10], recurso muy usado dentro del mundo de la anti-seguridad, ofrece una enorme base de datos de exploits, clasificados por tipo, aplicación y fecha. Entre ellos, puede encontrarse un exploit contra la vulnerabilidad citada de TikiWiki (el exploit 2288 en MilW0rm) . Este exploit sube una shell (un intérprete de comandos) al servidor afectado y ejecuta a través de él los comandos que se le indiquen como parámetro.

El objetivo principal de esta prueba consiste en modificar la página inicial del servidor web, aprovechándose del exploit anterior. Una vez realizada la tarea anterior, el alumno debe de indicar la forma de solucionar este agujero de seguridad. Para ello, tal y como se ha explicado anteriormente, el alumno podrá valerse del código fuente de la versión siguiente a la fecha de

publicación de la vulnerabilidad, en nuestro caso, la versión 1.9.5, y comparar los cambios sufridos por el fichero jhot.php entre una versión y otra:

Fichero jhot.php (1.9.4) Vulnerable	***Fichero jhot.php (1.9.5) Parche***
if (strstr($name, '.gif')) {	if (substr($name,-4,4) == '.gif') {
$hash = $absolute_name . md5(uniqid('.')). '.gif'; }	$hash = $absolute_name . md5(uniqid('.')). '.gif';
...	...
chmod("img/wiki/ $name", 0755);	}

Tabla 1. Ejermplo de código vulnerable (izq.) y parche correctivo (dcha.)

Como puede apreciarse, la vulnerabilidad se debe a un mal filtrado del tipo de archivos admitidos. En la versión vulnerable se filtran todos aquellos nombres de archivo que no contengan la cadena .gif; de forma inocente se quiere permitir sólo la subida de archivos gráficos .gif, pero no es un filtro adecuado el que se aplica, dado que un script PHP malicioso podría subirse al servidor con el nombre maligno.gif.php . Además, dichos archivos se tornan ejecutables mediante el comando chmod($fichero, 0755) lo que hace más devastador el error de seguridad.

El alumno finalmente debe de documentar el error y exponer de forma razonada pautas de programación segura que eviten o minimicen la posibilidad de cometer errores de programación como el anterior, así como realizar una pequeña auditoria al código en busca de patrones similares.

5. Entorno de competición

Las pruebas que forman parte de los retos propuestos a los alumnos se instalan en un entorno de competición seguro. Para la instalación de dicho campo de ataque y las aplicaciones web vulnerables, se usó una máquina virtual VMWare donde corría la distribución Damn Vulnerable Linux [2]. Esta distribución Linux está especialmente dotada de multitud de herramientas de seguridad, tests de penetración en sistemas, auditorías, exploits, etc. así como de versiones antiguas y vulnerables de aplicaciones web ya instaladas, configuradas y listas para ser explotadas en laboratorios de seguridad informática como el que hemos propuesto en este artículo.

6. Experiencias docentes

Durante los últimos dos años se ha venido aplicando el proceso descrito de presentación y explicación práctica de algunos temas de seguridad. Como puede verse en la tabla 2, las encuestas disponibles del curso 06/07 indican que los alumnos adoptan con agrado esta práctica. Durante el curso lectivo 06/07 los retos eran totalmente optativos. Durante el curso 07/08 algunos retos eran obligatorios y otros optativos.

Concepto evaluado	**Puntuación (sobre 5)**
El material usado para las prácticas y la teoría es adecuado	4,3
Se fomenta el trabajo en grupo	4,4
Las actividades prácticas propuestas se adecúan a la teoría vista en clase	4,5
El curso aporta capacitación para el trabajo del futuro	4,6

Tabla 2. Puntuación del alumnado al método propuesto (06/07, 30 alumnos)

Casi la totalidad de los alumnos participaron en todos ellos, logrando así aumentar la puntuación en las calificaciones finales e implicándose de forma activa en la asignatura.

7. Conclusiones y Trabajo futuro

Durante años el acceso a los detalles de problemas de seguridad relacionados con aplicaciones informáticas estaba prácticamente vedado, debido a que las licencias de los sistemas afectados impedían el acceso al código fuente. A día de hoy, cuando el software de código abierto se ha hecho ya de uso común, disponemos de enormes cantidades de código fuente disponible para ser analizado y escudriñado, en busca de aquellos errores y malas prácticas de programación que los hicieron vulnerables. Los alumnos pueden aprender de los errores, utilizar el histórico de conocimientos, de grandes y pequeñas aplicaciones, muchas de las cuales se usan actualmente a gran escala.

Se ha esbozado un procedimiento de docencia de los aspectos prácticos de la (anti)seguridad informática mediante el planteamiento de retos a los alumnos, cubriendo distintas áreas (vulnerabilidad, medidas correctivas, medidas preventivas, documentación y establecimiento de teoría). Como posible mejora y un mayor estímulo, se podría analizar la posibilidad de establecer un torneo inter-universitario de pruebas de seguridad informática, al igual que la ACM organiza anualmente sus torneos inter-universitarios de programación.

Referencias

[1] Common Vulnerabilities and Exposures Database. 2005, http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2005-1810

[2] Damn Vulnerable Linux, 2008, http://www.damnvulnerablelinux.org

[3] IBM Internet Security Systems, http://blogs.iss.net/archive/2007XFReport-Day1.html (2007)

[4] Oechslin, Philippe, Making a Faster Cryptanalytical Time-Memory Trade-Off, ,Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 2003, Proceedings. Lecture Notes in Computer Science 2729 Springer 2003, ISBN 3-540-40674-3

[5] Open Source Vulnerability Database, http://www.osvdb.com/show/osvdb/19634, 2005

[6] phpBB, http://www.phpbb.com/

[7] TikiWiki, http://tikiwiki.org

[8] Wordpress, Actualización de seguridad, http://wordpress.org/development/2005/05/security-update/

[9] WordPress, http://wordpress.org/

[10] www.milw0rm.com: web de seguridad informática con pruebas de concepto contra vulnerabilidades (exploits),