Curso: (30227) Seguridad Informática

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Tema Auditoría y pistas sobre algunos lenguajes

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Auditoría de programas

- Hacer revisiones de código es productivo casi siempre, pero no siempre vale la pena:
  - si hay problemas de este tipo, probablemente también los habrá mas graves.
- Cuanto antes pensemos en estas cosas, mejor. Por supuesto, no dejar de pensar en ellas.
- Primera auditoría: cuando se ha terminado el diseño del sistema, realizado por expertos

Análisis de la seguridad: arquitectura

- Adquisición de información (aprender sobre el sistema)
  - Leer y comprender las especificaciones, los documentos y otros materiales de diseño
  - Discutir en grupo
  - Determinar el contorno y cómo son los datos de críticos
  - 'Jugar' con el programa
  - Estudiar el código
  - Identificar las amenazas y acordar las fuentes relevantes de ataques

Análisis de la seguridad: arquitectura

- Discutir cuestiones de seguridad del producto
  - Discutir el funcionamiento del producto (y detectar desacuerdos o ambigüedades).
  - Identificar vulnerabilidades posibles
  - Hacer mapas de ataques y empezar a discutir posibles remedios
  - Entender los sistemas de seguridad, tanto planeados como instalados (pueden introducir sus propios problemas)

Análisis de la seguridad: arquitectura

- Calcular la probabilidad del compromiso
  - Preparar escenarios para fallos y ataques
  - Evaluar los controles contra esos ataques, para evaluar la probabilidad
- Hacer análisis de impacto
  - Determinar el impacto en los bienes y las metas del negocio
  - Considerar los impactos en la política de seguridad

Arquitectura: ¿qué mirar?

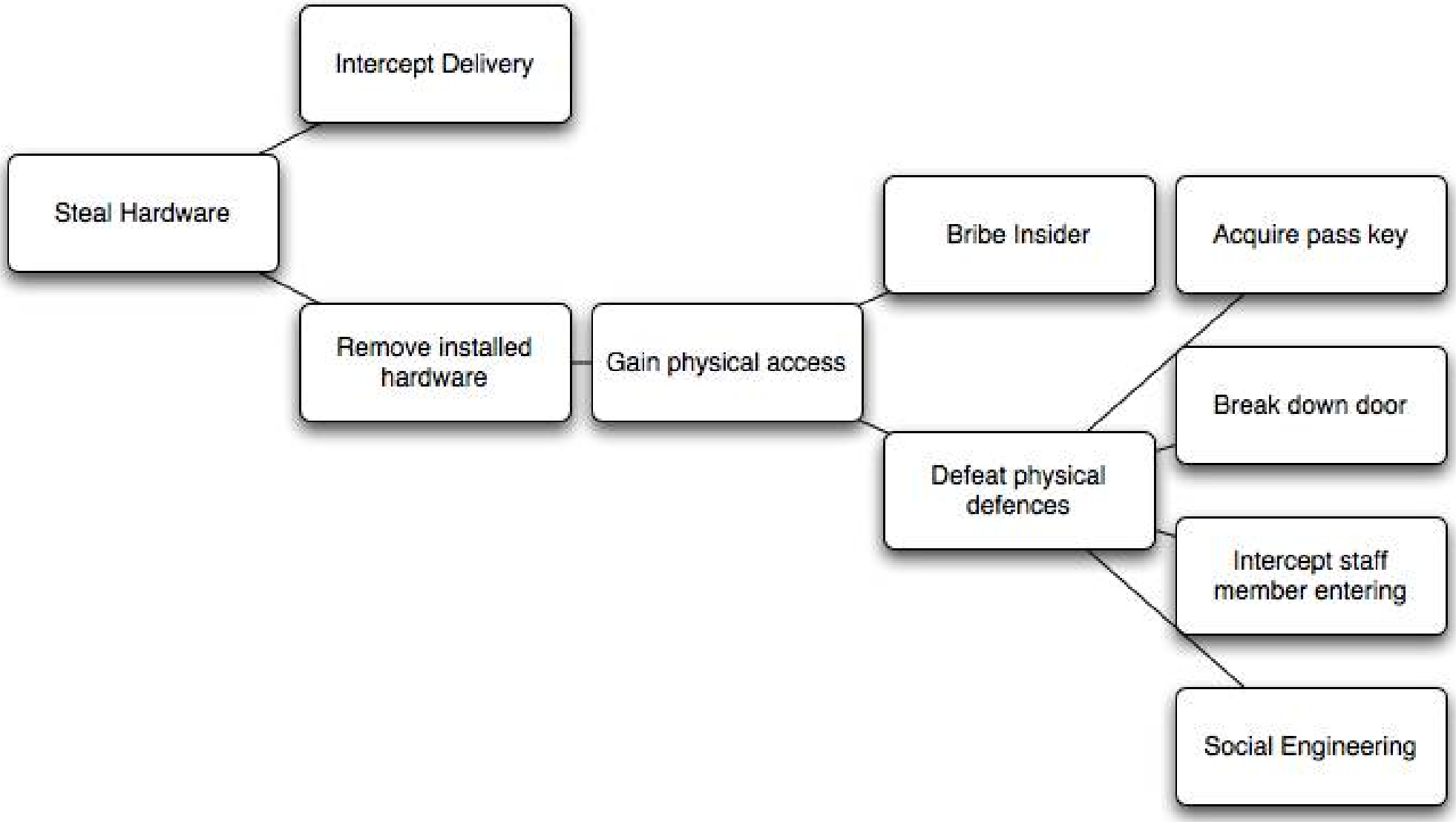
- Examinar los requisitos (debería haberlos, ¿no?)
- No tiene sentido buscar problemas que no son vistos como tales
- Si no los hay, es mejor empezar por construirlos

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Árboles de ataque

- Utilizar una visión de alto nivel del sistema
- Revisar la documentación relevante
- Conversar con los diseñadores
- Documentar contradicciones y discrepancias
- Establecer un plan de ataque
  - Buscamos no sólo riesgos reales, sino también potenciales
  - Al informar: describir los posibles ataques, análisis de las consecuencias, técnicas de mitigación

Ejemplo: Robar hw



Microsoft STRIDE Threat Model

- **S**poofing (Suplantar)
- **T**ampering (Modificar)
- **R**epudiation (Renegar)
- **I**nformation disclosure (Divulgar información)
- **D**enial of service (Denegación de servicio)
- **E**levation of privilege (Elevación de privilegios)

`http://msdn.microsoft.com/en-us/library/ee823878(v=cs.20).aspx`

Y, como un juego: ‘ Elevation of Privilege (EoP) Card Game’

`http://www.microsoft.com/security/sdl/adopt/eop.aspx`

Libro: Adam Shostack. ‘Threat Modeling: Designing For Security’. Wiley. 2014.

Auditoría de la implementación

Objetivos

- Validar que la programación realmente cumple la especificación
  - A veces, irrealizable
  - Puede ser mejor hacer las preguntas adecuadas a los desarrolladores
- Fallos propios de esta fase
  - desbordamiento de buffer, condiciones de carrera, ...
- Complicado!

Auditoría de código

Una auditoría completa es demasiado trabajo, normalmente basta con una auditoría 'suficientemente' completa.

- Identificar puntos de entrada al programa
  - Entradas de los usuarios
  - Entradas de otros programas
    Lecturas de la red, de un fichero, de las interfaces...
- ¿Cómo es el API de entrada?

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Auditoría de código

Buscar síntomas de problemas

- Experiencia
- Examen cuidadoso
- No siempre lo que parece peligroso lo es
- Tan profunda como sea posible

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Herramientas

Primera etapa: análisis léxico (búsqueda de patrones)

- ITS4 (Cigital, abandonada)
- RATS (Fortify, abandonada)
- Flawfinder (`http://www.dwheeler.com/flawfinder/`)

Son poco mas que buscadores de cadenas o de patrones.

Herramientas . . .

Segunda generación: análisis sintáctico y (cada vez) más

- Coverity (`http://www.coverity.com/`)
- HP Fortify Static Code Analyzer (Antes Fortify)

  (`http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/`)
- IBM Security AppScan Source (Antes Ounce Labs).

  (`http://www-01.ibm.com/software/rational/products/appscan/source/`)

Las plataformas y sus consecuencias

Aviso: elegir una tecnología supone elegir también sus problemas asociados

CGI

Common Gateway Interface (CGI)

- Cualquier ejecutable puede invocarse desde el servidor web
- La entrada al programa es mediante:
  - variables de entorno
  - entrada estándar
  - línea de instrucciones
- La salida deben ser instrucciones HTTP por la salida estándar
- Prácticamente ya no se usa, pero las plataformas actuales heredan algunas características

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

CGI. Variables de entorno

Estáticas

- GATEWAY_INTERFACE (versión)
- SERVER_SOFTWARE (programa y versión)

Directas:

- REMOTE_ADDR
- REMOTE_HOST
- REMOTE_IDENT
- ...

Se traducen a variables de entorno y se transmiten a la aplicación

- Las variables a mayúsculas
- Se añade HTTP_ delante

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

CGI. Variables

Replicadas

- Cada línea HTTP que ve el servidor web puede generar una
- Algunas puede que no

Sintetizadas

- Variables relativas al nombre de los objetos referenciados (que, a pesar de que originalmente tenían que ver con objetos del sistema, actualmente en muchos casos no tiene por qué ser así)

Perl

- Empezó usándose para programitas CGI por su flexibilidad y desarrollo rápido de aplicaciones de procesamiento de texto

Problemas de inyección de SQL:

- Prevenciones habituales

Acceso a ficheros

- open()

Invocación de un intérprete de instrucciones (shell)

- open(), system(), exec()
- Comillas invertidas: $fileinfo = ‘ ls -l $filename ‘

Inclusión de ficheros:

- require(). También use(), do()

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Perl

Evaluación en línea

- eval()

Cross–Site Scripting

- HTML::Entities::encode()
- URI::Escape::uri_encode()
- Y algunas mas ...

Taint Mode

- Vigila variables que contienen datos provenientes del exterior
- Si el programa va a hacer algo peligroso, falla

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

PHP

Inyección SQL

- Atención!

Acceso a ficheros

- fopen(), readfile(), dir(), unlink(), file(), mkdir(), symlink(), get_file_contents()
  $myfile = "/usr/local/myapp/var/:".$_GET['filename'];
  $fp = fopen($myfile, "r")
- También conexiones (urls: http, ftp, ...)
  Aunque se puede desactivar con allow_url_fopen en el php.ini

Invocación intérprete de instrucciones

- exec(), shell_exec(), system(), popen(), proc_open(), passthru()

PHP

Inclusión de ficheros

- require, include
- mejor require_once() e include_once()

Evaluación en línea

- eval()
- preg_replace() (con /e) ejecuta un código determinado por cada texto que satisface la expresión

Cross–site scripting

- htmlspecialchars(), htmlentities(), urlencode()

PHP

Configuración

- register_globals: cualquier variable enviada por los usuarios se pone a disposición del programa (cada vez se usa menos). Principalmente es peligroso con variables del programa que no se han inicializado
- magic_quotes:
  - Opción de configuración magic_quotes_gpc (get, post, cookies)
  - magic_quotes_runtime (también para otros valores generados en tiempo de ejecución)
- .inc opción. Se trata de poner determinados ficheros de configuración con esa extensión. Si el servidor no está bien configurado, puede ocurrir que la configuración se muestre.

Java

Inyección SQL

- Ojo!!

Acceso a ficheros

- java.io
- getRealPath(), getPathTranslated()

Acceso al interprete de instrucciones

- No se usa mucho
- getRuntime() en java.lang
- Luego exec()

Inclusión de ficheros

- RequestDispacher (para transferir el control de flujo)
- Cuidado con include(), forward()

Java

Inclusión de ficheros JSP

- <%@ include file="include.jsp" %>
- <jsp:include page="include.jsp" />
- <jsp:include page='<%"browserActions/" + request.getParameter("_actionPage") + ".jsp" %>

Evaluación 'inline'

- Java no ejecuta el código sobre la marcha (hay compilación)
- Pero hay algunos sistemas para permitir ese tipo de ejecución (BeanShell, Jython y, claro, ejecución de ficheros JSP).

Cross Site Scripting

- java.net.UrlEncoder.encode()
- Método response.encodeURL() codifica datos para salida
- Cuidado con las etiquetas <c> (sólo sirven las <c:out>

Java

Problemas con la concurrencia

- Hay que tener cuidado de que los servlets sean seguros frente a la concurrencia
- Se puede hacer que sean secuenciales (SingleThereadModel)
- Cuidado con las variables globales, . . . , (o similares)

Configuración

- Los servlets se pueden ver como un árbol web virtual en web.xml en el directorio WEB-INF
- Todos los servlets que se ofrezcan 'al exterior' son potencialmente peligrosos: los menos posibles, y en las condiciones mas exigentes posibles

ASP. Active Server Pages

Inyección de SQL

- Típicamente se usan lo ActiveX Data Objects (ADO)
  - Connection (cuidado con Connection.Execute()
  - Command (Command.Execute())
  - RecordSet (el método Open)

Acceso a ficheros

- Objeto Scripting.FileSystemObject
- Y los relacionados

Acceso al intérprete de instrucciones

- Habitualmente se hace a través del objeto Windows Scripting Host (WshShell) aunque no es tan habitual como en Unix.
  - Métodos Exec(), Run()

Inclusión de ficheros

- Habitualmente se hace mediante SSI (Server Side Includes) lo que lo inmuniza frente a algunos problemas (se procesa antes de mostrarse).
- De todas formas, cuidado con Server.Execute(), Server.Transfer()

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

ASP

Evaluación en línea

- VBScript es el lenguaje utilizado habitualmente
- Execute(), Eval(), ExecuteGlobal()

Cros Site Scripting

- Server.HTMLEncode()
- Server.URLEncode()

Configuración

- Ficheros .inc (cuidado con que el servidor los maneje correctamente, igual que en PHP)

ASP.Net

Inyección de SQL

- System.Data para interactuar con las fuentes de datos
- Cuidado con fijar los tipos de datos

Acceso a ficheros

- La entrada salida se gestionan mediante System.IO
- Pero hay mas posibilidades

Invocación del intérprete de instrucciones

- Process.Start()

Inclusión de ficheros

- Como Java, no es fácil incluir programas dinámicamente
- <!--#include file="inc_footer.aspx"-->

Evaluación en línea

- No está permitida la ejecución directa
- System.CodeDom.Compiler permite ejecutar para diversos lenguajes

ASP.Net

Cross Site Scripting

- Server.HTMLEncode(), Server.URLEncode()
- Deniega explícitamente peticiones que contienen < y >

Configuración

- web.config
- Puede ser también machine.config

ViewState

- Se almacena en el cliente, mediante una cookie, con información sobre contenido de formularios, y otras informaciones
- Protegido, pero no cifrado
- Se puede cifrar (pero hay que activarlo)

ASP.Net

Control de acceso

- ASP.Net permite establecer control de acceso para todo el sitio.

Autorización

- web.config permite configurar restricciones sobre métodos, usuarios, grupos, papeles,...

AppSettings

- En web.config para proporcionar parámetros específicos para una aplicación