Curso: (62612) Diseño de aplicaciones seguras

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 XIII: Auditoría de programas

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)

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

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

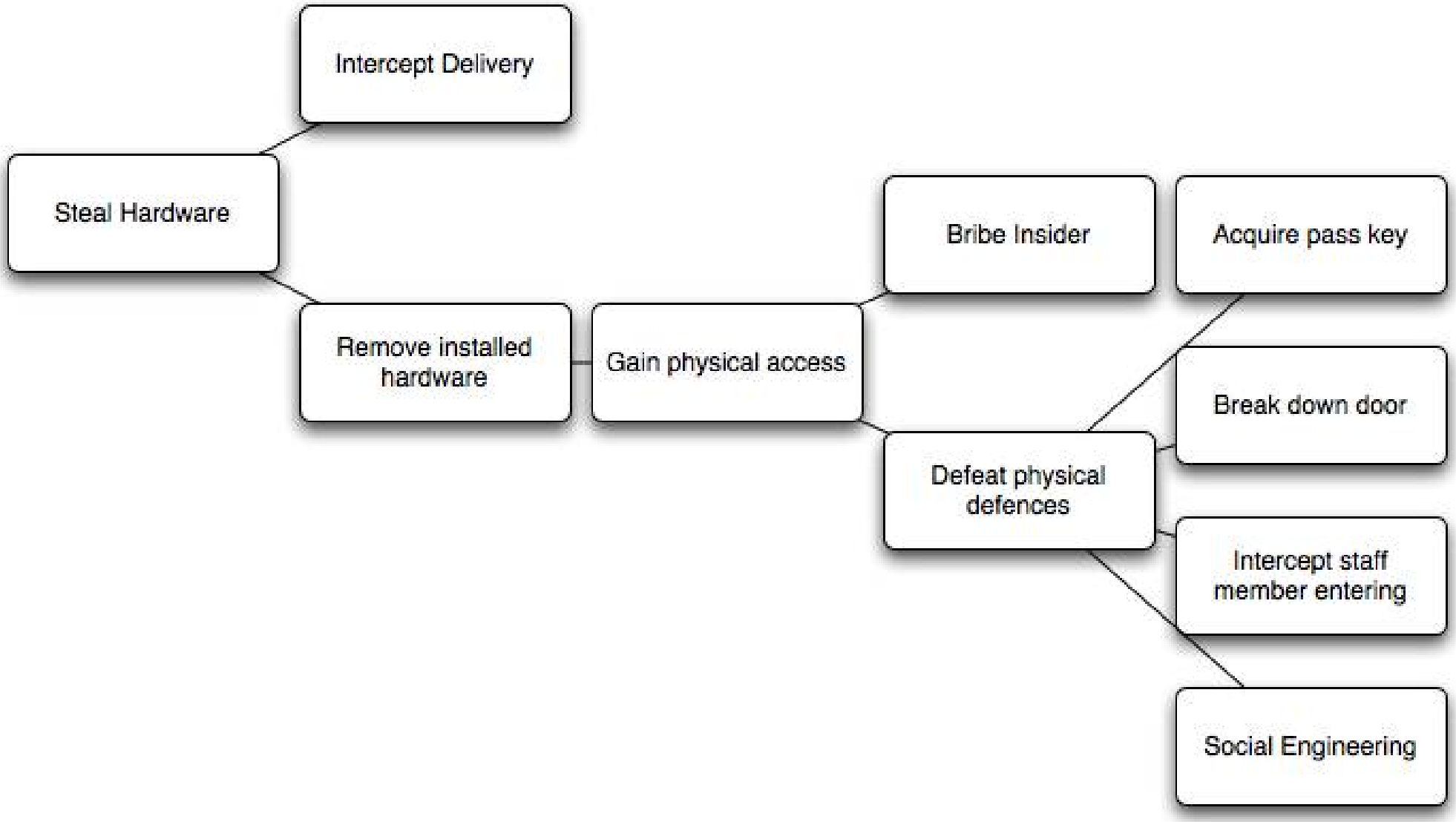
Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

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

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

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`

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?

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://www.hpenterprisesecurity.com/products/hp-fortify-software-security-center/hp-fortify-static-code-analyzer`)
- IBM Security AppScan Source (Antes Ounce Labs).

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