Primeras Jornadas de Seguridad Web
OWASP DAY ARGENTINA 2010

"La seguridad como ventaja competitiva"

Web Application: Security Tips

Hernán M. Racciatti
hracciatti@siclabs.com
SICLABS

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

OWASP ARGENTINA

Acerca del Autor

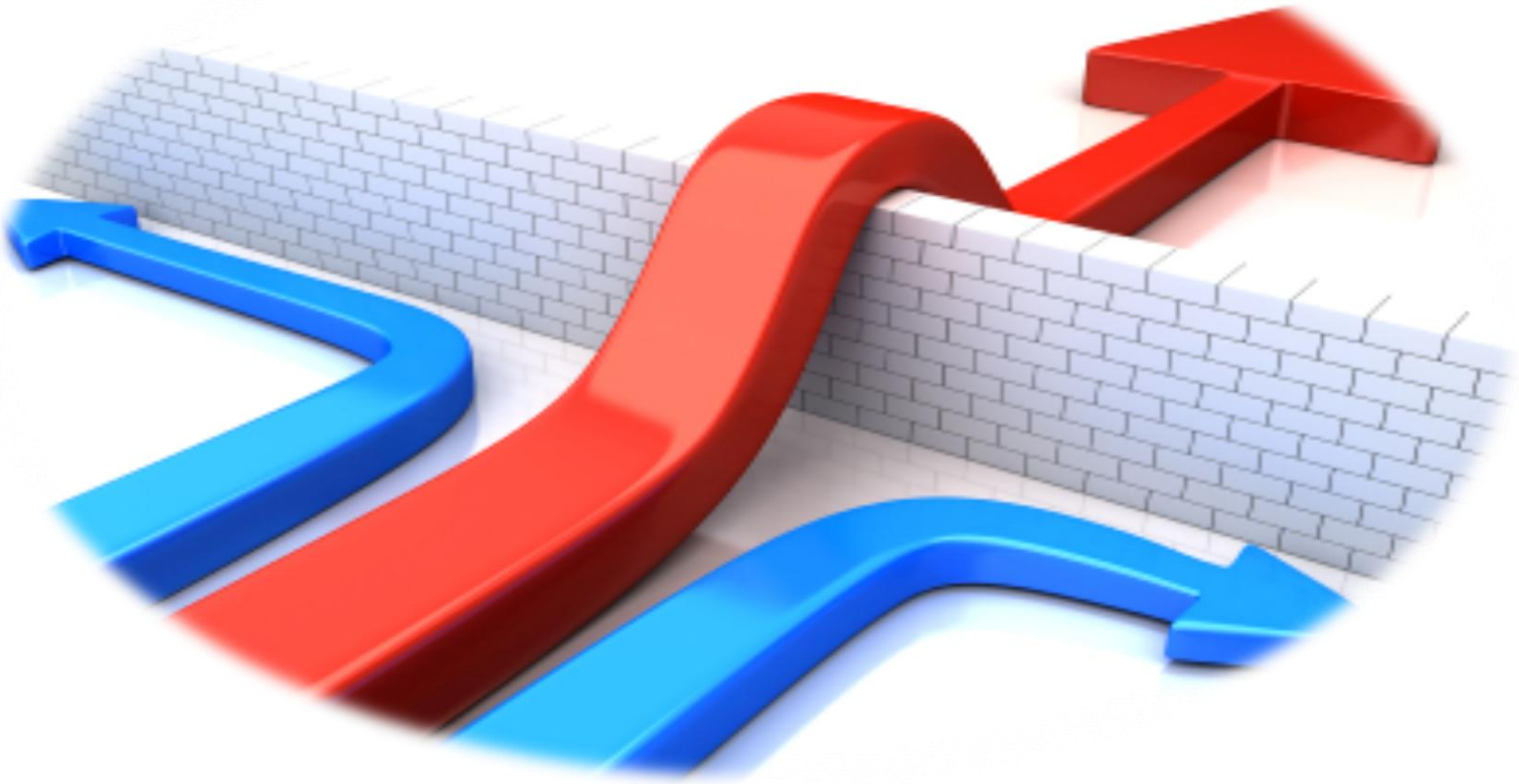
- Analista Programador, CISSP, CSSLP, CEH, MCP, QCS
- Miembro del Core Team de ISECOM
- ISSAF Key Contributor at OISSG
- Colaborador de los proyectos ISSAF, HHS y OSSTMM, entre otros.
- Director de Comunicaciones de ISSA Argentina
- Security Director at SIClabs

** ISECOM (Institute for Security and Open Methodologies)*
** OISSG (Open Information System Security Group)*
** ISSA (Internet Systems Security Association)*
** ISSAF (Information Systems Security Assessment Framework)*
** OSSTMM (Open Source Security Testing Methodology Manual)*
** HHS (Hackers Highschool, Security Awareness for Teens)*

Agenda

- Introducción a la Problemática
- Causas del Software Inseguro
- En la Practica
- Nuevos Motivos
- Security Tips
- Comenzando Ahora!
- Recomendaciones
- Referencias y Lectura Complementaria
- Preguntas

Introducción a la Problemática



Introducción a la Problemática (Cont.)

- El equipo de desarrollo y los responsables de proyectos creen que la seguridad no aporta ningún valor.
- Resolver vulnerabilidades antes del lanzamiento de un producto suele ser visto como un proceso muy costoso.
- La seguridad de la información, no suele ser una competencia requerida a la hora de realizar contrataciones.
- Las organizaciones suelen premiar tiempos de entrega, usabilidad y eventualmente performance… NO seguridad.
- La seguridad no es concebida como un proceso
- Los usuarios finales, en general no suelen demandar software seguro.

Introducción a la Problemática (Cont.)

Desarrollador

"No hace falta agregar estos chequeos aquí, de todos modos se ha configurado seguridad a nivel de base de datos"

Introducción a la Problemática (Cont.)

DBA

"Menos mal que el desarrollador pensó en la seguridad, de no ser así… tendría que haber lidiado con estos extraños permisos a nivel filas y columnas, además… de todos modos el server esta seguro pues el administrador lo ha asegurado"

Introducción a la Problemática (Cont.)

Administrador

"Mmm… aquí no incluiré ACLs de todos modos, la aplicación y la base de datos seguramente están configurados para evitar este tipo de accesos"

Introducción a la Problemática (Cont.)

- **Seguridad Convencional**
  - Las aplicaciones son liberadas al circuito comercial, a fin de evitar que el competidor lo haga primero.
  - Hackers/Researchers encuentran vulnerabilidades o debilidades en el software.
  - Aparece un website con el detalle de la vulnerabilidad y como explotarla.
  - Se libera el fix, Parche, SP o se publica el workaround.
  - Este parche pasa a engrosar la pila de parches que el administrador debe testear e instalar!!
  - Y en su empresa? Los desarrollos internos?

Introducción a la Problemática (Cont.)

- **Seguridad Convencional**
  - Cuando alguien detecta una vulnerabilidad en alguna plataforma (Windows / Unix / Linux / Oracle / SQL Server), generalmente ésta tarde o temprano es publicada.
  - Del mismo modo cuando la vulnerabilidad es reparada, generalmente los parches son puestos a disposición de todos los usuarios.

- **Seguridad en Aplicaciones de Desarrollo Propio**
  - Nunca existirá un parche general que resuelva los problemas en NUESTRA aplicación web.
  - Nadie va a descubrir vulnerabilidades en NUESTRA aplicación…

Causas del Software Inseguro

- Requerimientos de seguridad pobres o nulos
- Ausencia de proceso de SDLC
- Diseño orientado a funcionalidad
  - Diseño Funcional != Diseño de Seguridad
- Desarrolladores sin entrenamiento en seguridad
  - Pobre conocimiento de las amenazas
- Escasez de especialistas
- Ausencia de políticas de codificación segura
- Pobres practicas de QA & Security Testing
- Tiempo y presupuesto acotados

En la Practica

SOCIAL NETWORKING

Persistent XSS on Twitter.com

BY PREFECT · JUNE 24, 2010 · PRINT THIS POST · POST A COMMENT

FILED UNDER TWITTER, XSS

Twitter user 0wn3d_5ys has demons
found on June 21st using his own Tw

Home » news

SQL Injection hits again; 168,000 personal records exposed

Submitted by Robert Abela on May 18, 2010 – 9:27 pm

No Comment

A hacker, who calls himself "ins3cted", has demonstrated to Webwereld via video how by exploiting a simple SQL injection, he can retrieve 168,000 personal records from a Dut

Experience the OV

and Flevoland are
gn that promotes
d" also explained
as long as the
ne it's sensitive
you want falling

Analysis of web application vulnerabilities detected in 2009 showed that **almost half the reviewed systems contained errors**. The statistics is based on the data about 5560 web applications gathered in the course of 6239 automatic scans and detailed analysis of 77 web applications. 13434 errors of various risk levels were detected in all reviewed applications and 1412 examples of malicious code were found on the pages of vulnerable systems. **1.7% of compromised sites were spreading malicious software**; each of these sites contained vulnerabilities that allow attackers to execute arbitrary commands directly on server, which proves that such vulnerabilities can be exploited to compromise the system.

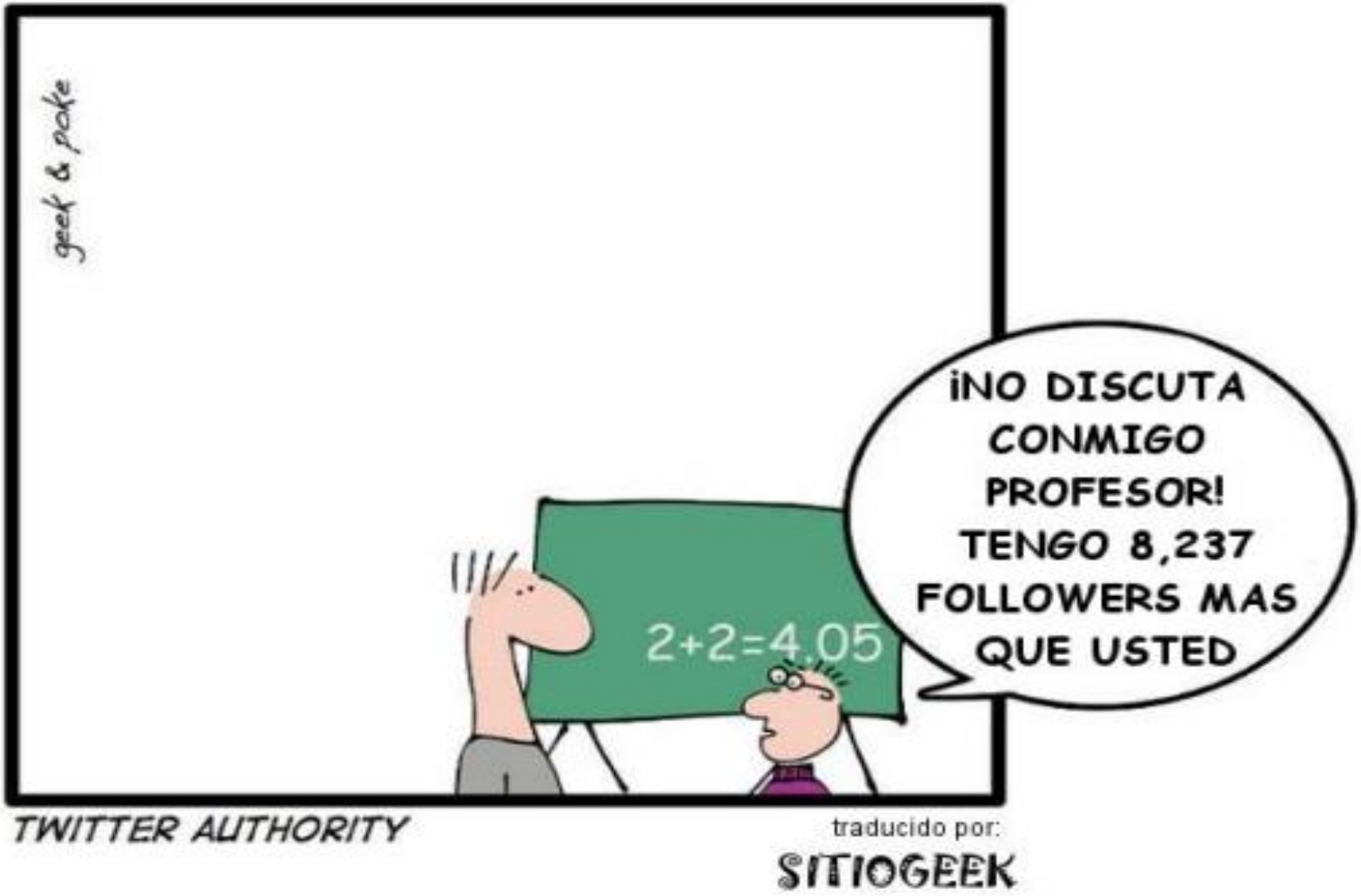
* http://praetorianprefect.com/archives/2010/06/persistent-xss-on-twitter-com/
* http://www.acunetix.com/blog/news/sql-injection-records-exposed/
* http://ptresearch.blogspot.com/2010/06/web-application-vulnerability.html#more

Nuevos Motivos

- Habeas Data
- BCRA 4609
- ISO27000
- DSS-PCI
- HIPAA
- SOX

Se puede estar en "Compliance" y aun ser inseguro!!

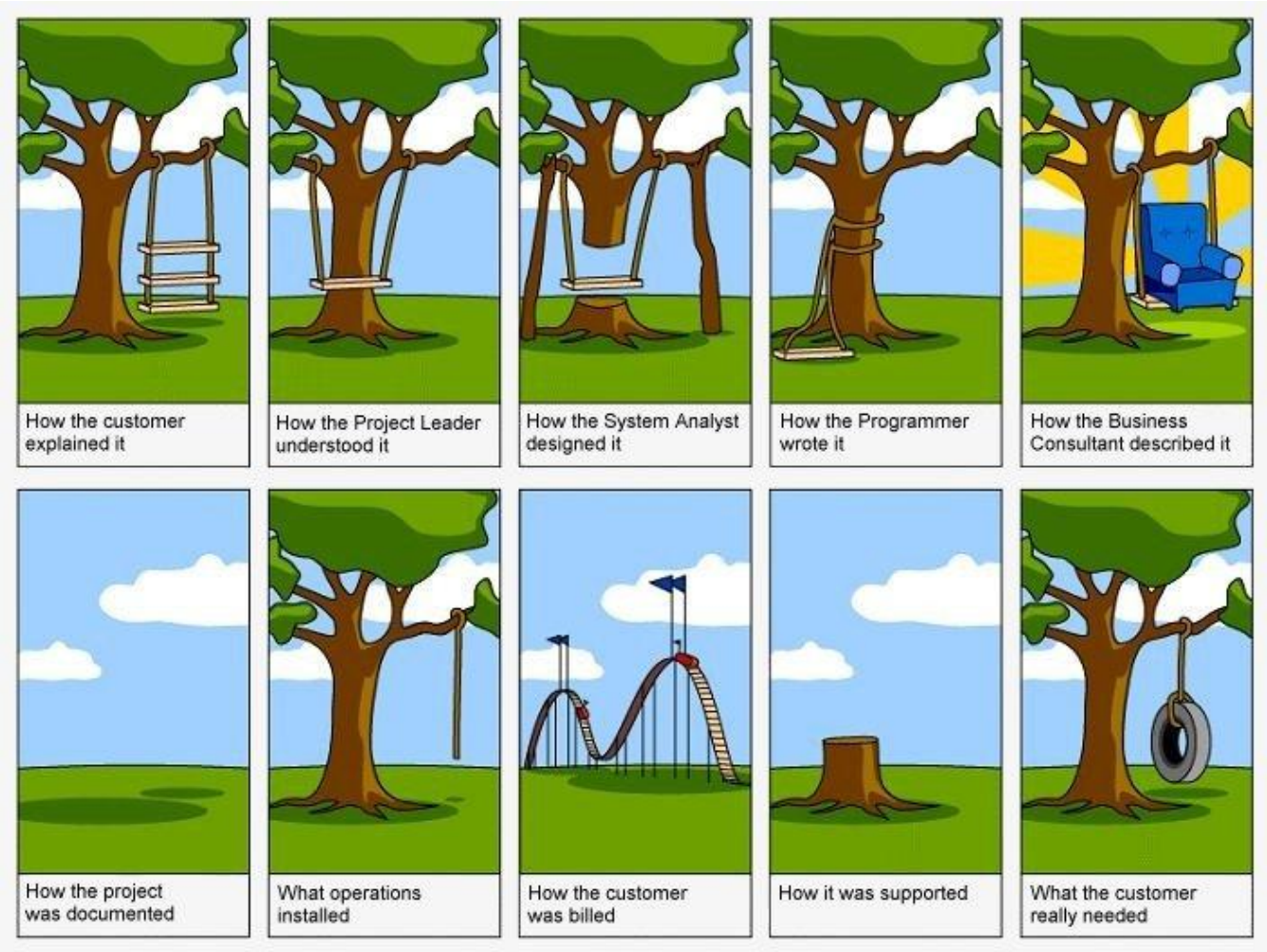
Tip #1: Entrenamiento



Tip #1: Entrenamiento (Cont.)

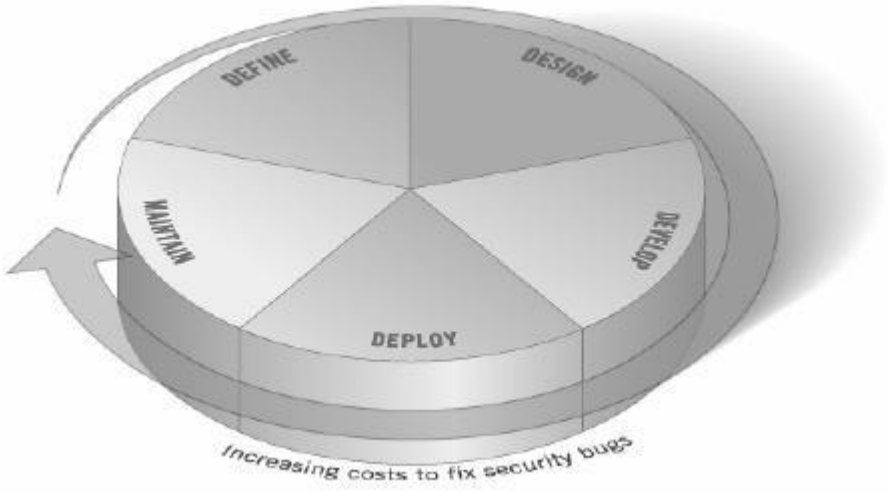
Canonicalization
Information Disclosure
LDAP Injection
Spoofing
Heap Overflows
Format Strings
Weak Permissions
Authentication Bypass
Shared Sections
ActiveX Repropoursing
Stack Overflows
Weak Crypthography
Path Traversal
XSS
Logic Errors
XML Injection
Integer Overflows
Design Flaws
Commands Injection

Tip #2: SDLC

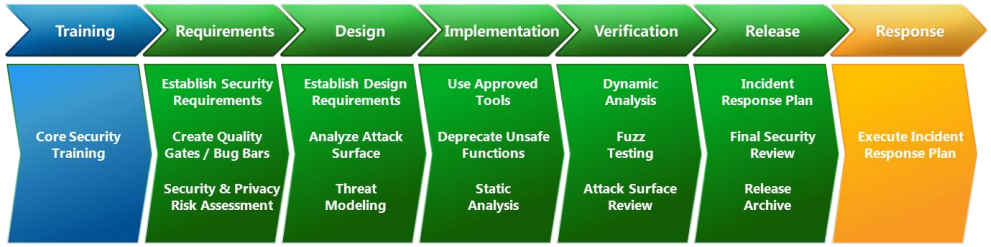

How the customer explained it
How the Project Leader understood it
How the System Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Tip #2: SDLC (Cont.)

- SDL implica instaurar procesos de desarrollo de software, en los cuales la seguridad de la información, sea tenida en cuenta en cada una de las etapas del su ciclo de vida.

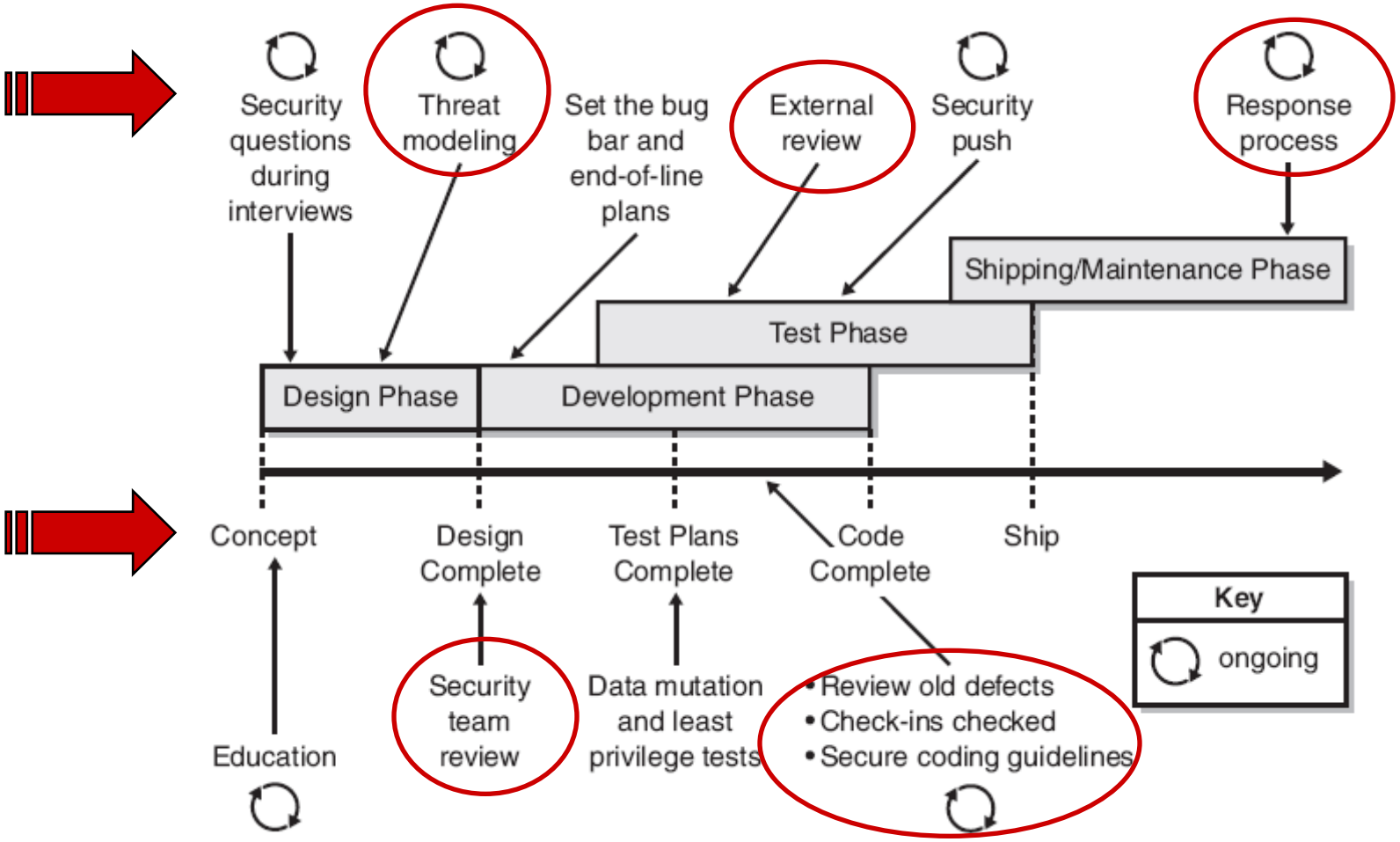


- Iniciación del Proyecto y Planificación
- Definición de los Requerimientos Funcionales
- Especificaciones de Diseño del Sistema
- Desarrollo del Software y Documentación
- Aceptación
- Puesta en Producción (Implementación)
- Mantenimiento Operativo (Post-Instalación)
- Revisión y Reemplazo de Sistemas



- Análisis y Definición
- Diseño y Desarrollo
- Implantación y Explotación

Tip #2: SDLC (Cont.)



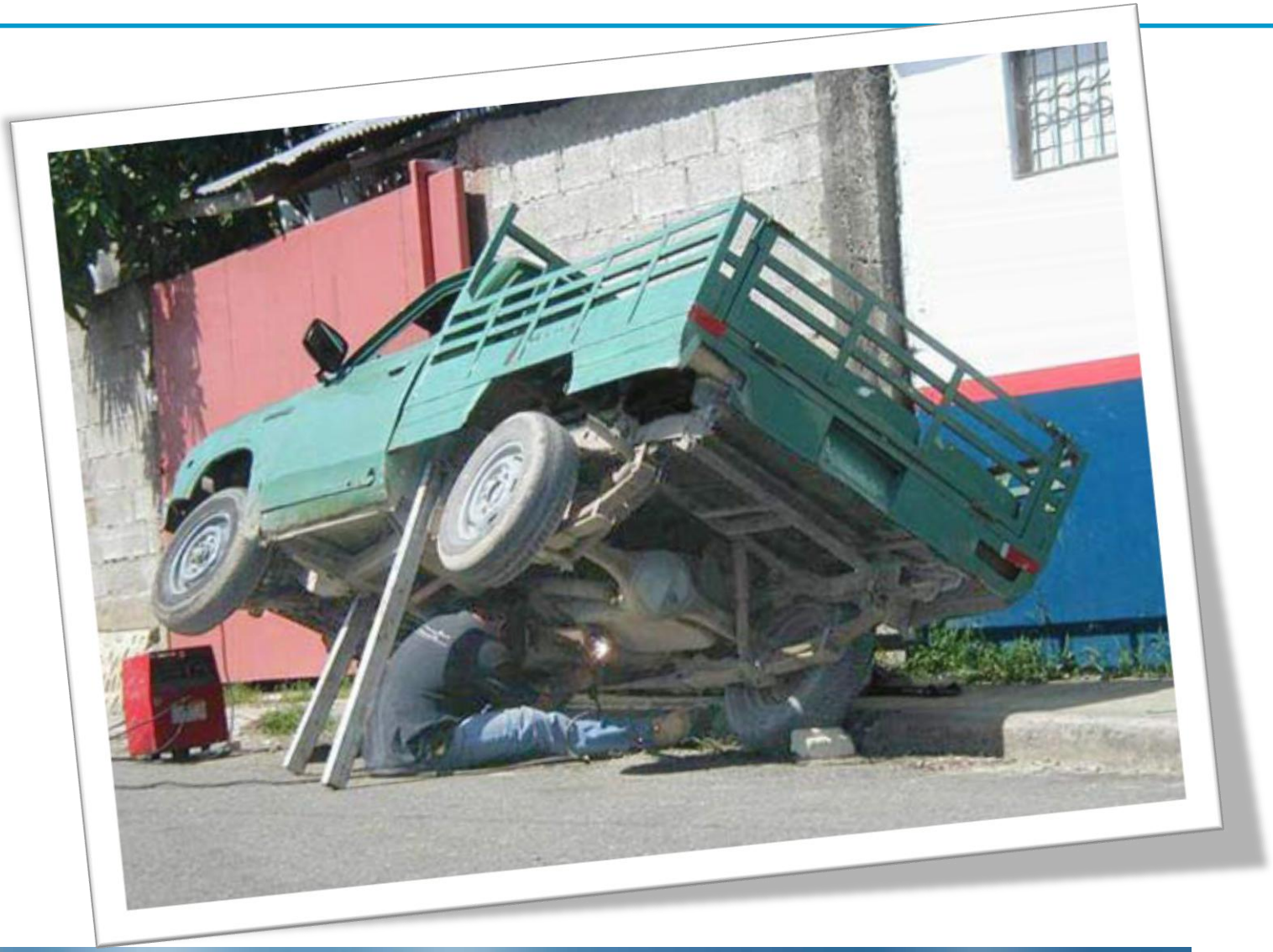
* From the Book: Writing Secure Code, Second Edition - Microsoft Press (Michael Howard / David LeBlanc)

Tip #2: SDLC (Cont.)

- Seguridad a lo largo del proceso
- Análisis de Riesgo desde el Inicio
- Especificación de Requisitos de Seguridad
- Definición y Ejecución del Plan de Pruebas
- Separación de Ambientes
- Métodos Puesta en Producción
- Mantenimiento / Control de Cambios / Operaciones

Es imposible construir aplicaciones seguras, si no se entienden las amenazas

Tip #3: Gestión de Riesgos



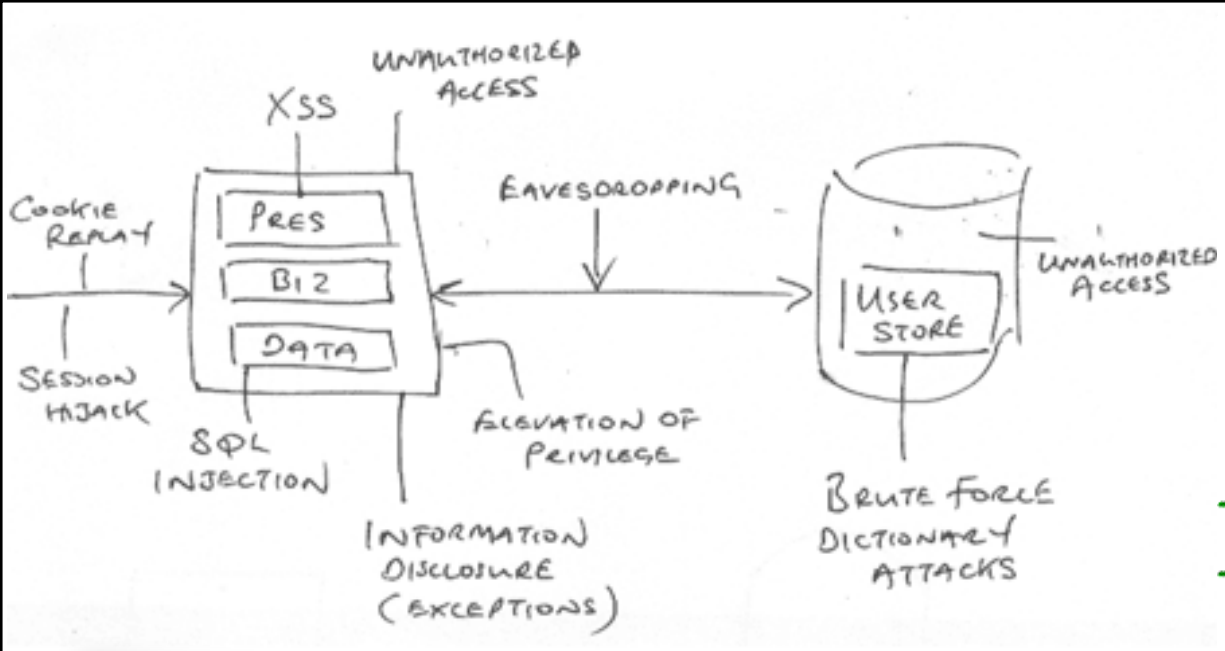
Tip #3: Gestión de Riesgos (Cont.)

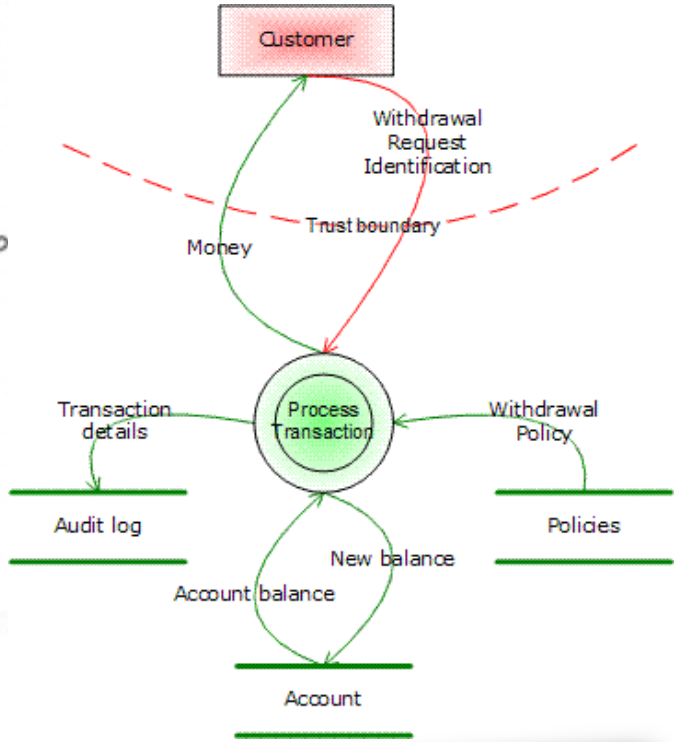
Threat Modeling

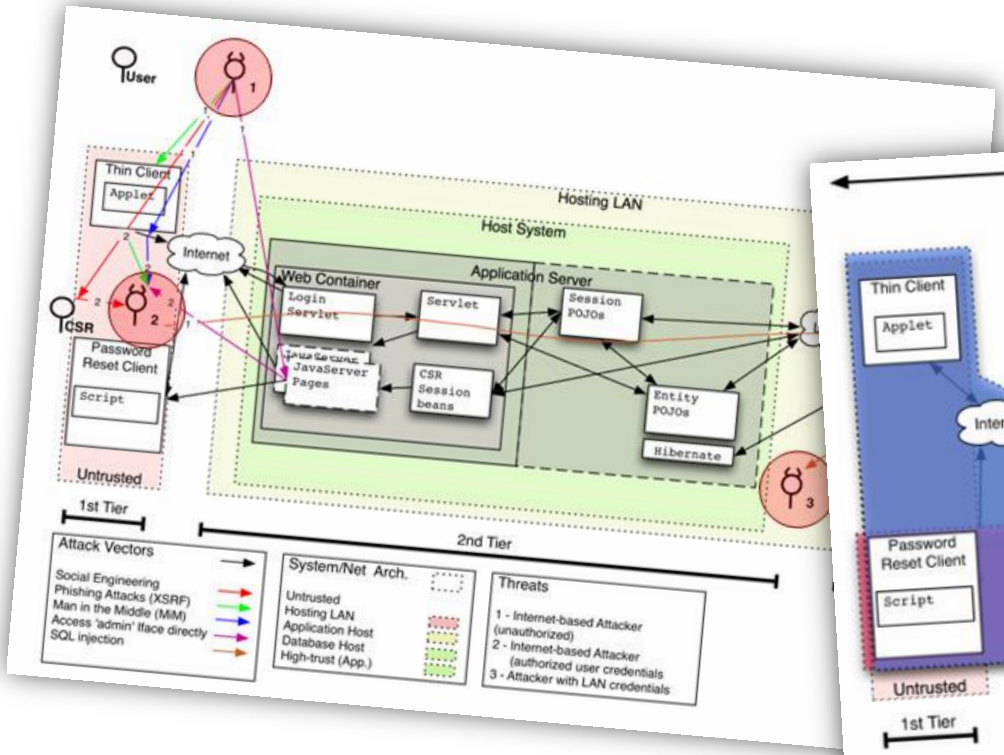
- Método de análisis basado en la seguridad.
- Parte crucial del proceso de diseño.
- Proceso para la evaluación y documentación de riesgos de seguridad (++).
- Potente instrumento de comunicación!

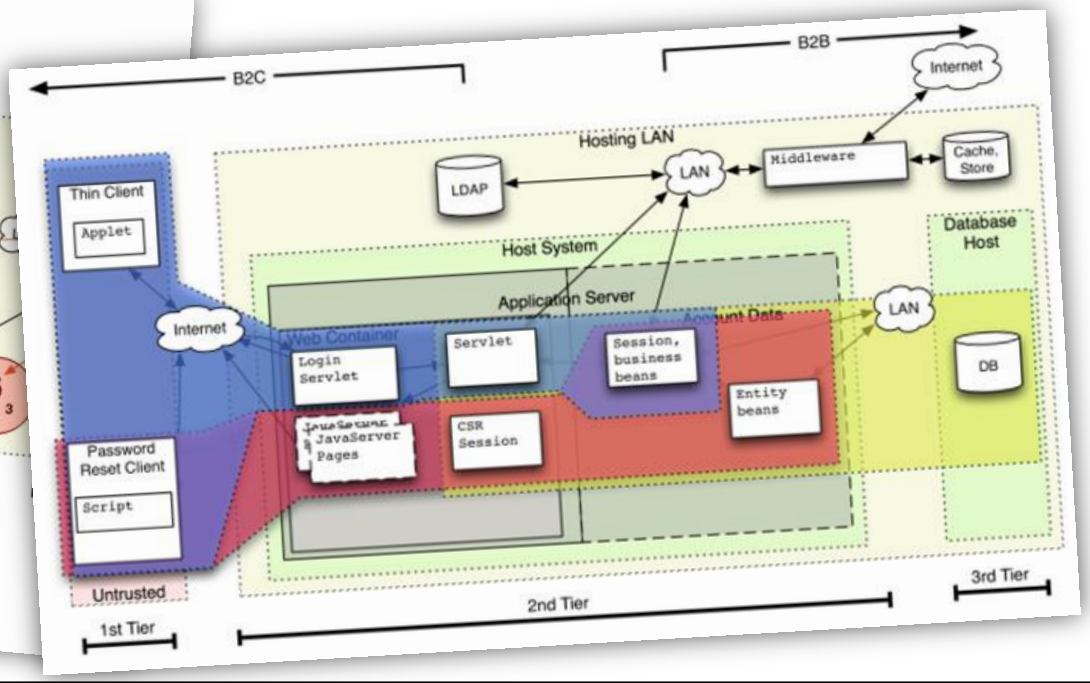
Permite:

- Identificar, evaluar y comprender amenazas de seguridad.
- Colaborar en la identificación de estrategias de mitigación.
- Colaborar en el proceso general de reducción de riesgos. (↓Costo de Asegurar)
- Justificar la implementación de controles.
- Mejorar la integración de nuevos miembros al *"Dev Team".*
- Conocer *donde* el producto es mas vulnerable, *cuales* amenazas requieren ser mitigadas y *como* direccionar las mismas… *proveer un proceso!!*


UNAUTHORIZED ACCESS
XSS
COOKIE REPLAY
PRES
BIZ
DATA
EAVESDROPPING
USER STORE
UNAUTHORIZED ACCESS
SESSION HIJACK
SQL INJECTION
ELEVATION OF PRIVILEGE
INFORMATION DISCLOSURE (EXCEPTIONS)
BRUTE FORCE DICTIONARY ATTACKS

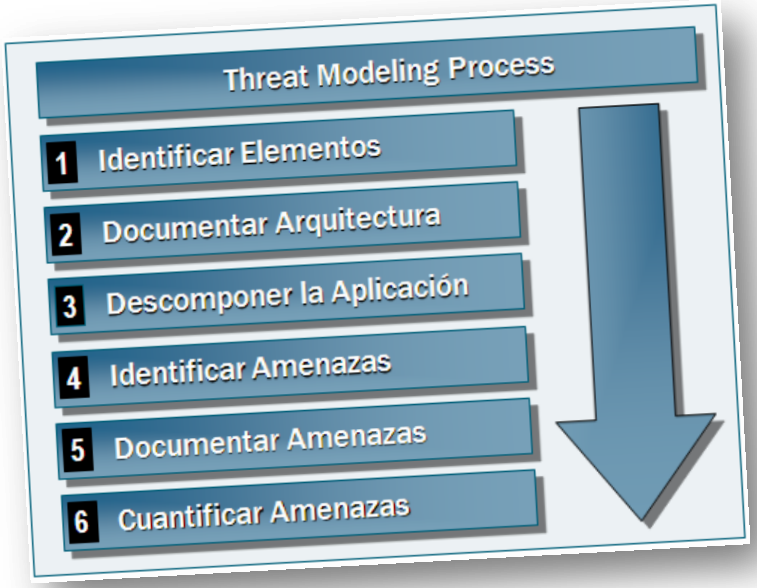

Customer
Withdrawal Request Identification
Trust boundary
Money
Transaction details
Process Transaction
Withdrawal Policy
Audit log
Policies
New balance
Account balance
Account

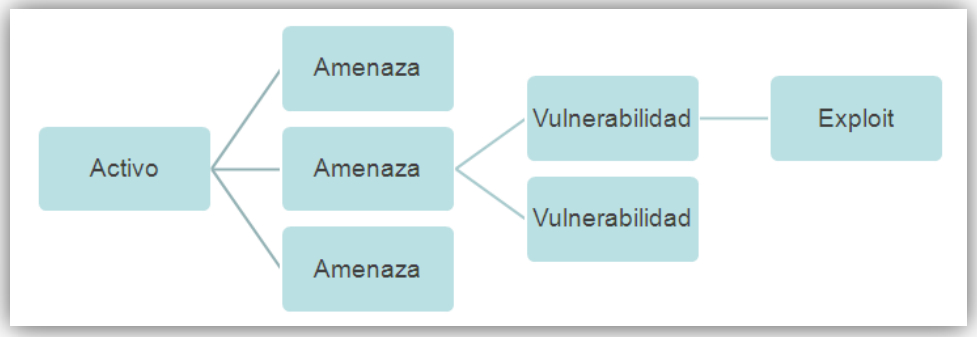

User
Thin Client
Applet
Internet
Hosting LAN
Host System
Web Container
Application Server
Login Servlet
Servlet
Session POJOs
JavaServer Pages
CSR Session beans
Entity POJOs
Hibernate
CSR
Password Reset Client
Script
Untrusted
1st Tier
2nd Tier
Attack Vectors
Social Engineering
Phishing Attacks (XSRF)
Man in the Middle (MiM)
Access 'admin' Iface directly
SQL injection
System/Net Arch.
Untrusted
Hosting LAN
Application Host
Database Host
High-trust (App.)
Threats
1 - Internet-based Attacker (unauthorized)
2 - Internet-based Attacker (authorized user credentials
3 - Attacker with LAN credentials


B2C
B2B
Internet
Hosting LAN
LDAP
LAN
Middleware
Cache, Store
Thin Client
Applet
Host System
Database Host
Application Server
Account Data
Internet
Web Container
Login Servlet
Servlet
Session, business beans
Entity beans
CSR Session
JavaServer Pages
LAN
DB
Password Reset Client
Script
Untrusted
1st Tier
2nd Tier
3rd Tier

Tip #3: Gestión de Riesgos (Cont.)

- Que activos de información se deben asegurar?
- Como puede infringirse el daño?
- Que debilidad o falla permite que la amenaza se concrete?
- Que acción puede un atacante llevar adelante para explotar una vulnerabilidad?





✓ *Atacante gana acceso privilegiado a base de datos con información sensible. (Amenaza)*
✓ *Cuenta de administrador "admin" con password "admin" (Vulnerabilidad)*

Tip #3: Gestión de Riesgos (Cont.)

- **D** Damage (Cuanto daño puede causar?)
- **R** Reproducibility (Cuan dificultosa es de reproducir?
- **E** Exploitability (Qué tan fácil es de explotar?)
- **A** Affected Users (Qué cantidad de usuarios pueden verse afectados?)
- **D** Discoverability (Que tan fácil es su descubrimiento?)

- **S** Spoofing (Se puede acceder con una identidad falsa?)
- **T** Tampering (Se pueden modificar datos de modo no autorizado?)
- **R** Repudiation (Puede un atacante repudiar sus acciones?)
- **I** Information Disclosure (Se puede acceder a información reservada?)
- **D** Denial of Service (Es posible disminuir la dispo. de la aplicación?)
- **E** Elevation of Privilege (Puede un atacante elevar sus privilegios?)

	Alto (3)	Medio (2)	Bajo (1)
Damage	El agresor puede obtener datos muy sensibles, dañar servidores etc.	Puede obtener datos sensibles	Puede acceder a datos poco importantes
Reproducibility	Siempre es posible	Sucede si se realiza en un corto tiempo	Raramente se puede concretar
Exploitability	puede hacerlo	Se deben tener ciertos conocimientos	Tal vez alguno
Affected Users	La mayoría	Algunos	Pocos, si es que hay alguno
Discoverability	Fácil de ver	Más costoso de ver	Muy difícil de encontrar

Combinando STRIDE con DREAD

- Calculo del Nivel de Riesgo de una Amenaza
- Alto/Medio/Bajo – 1 a 10
- Ejemplo:
  - Amenaza #1: Usuario malicioso visualiza información confidencial
    - STRIDE: Information Disclosure
  - Damage: 8
  - Reproducibility: 10
  - Exploitability: 7
  - Affected Users: 10
  - Discoveranility: 10
  - Valor de Riesgo DREAD: (8+10+7+10+10/5)=9
- Establecimiento de prioridades
- Respuesta a las amenazas

Comenzando Ahora!

1. Asuma que todo INPUT es malicioso
2. Valide todo INPUT (Todo es TODO…)
3. Establezca Validaciones Efectivas (Tipo, Largo, Formato y Rango)
4. Codifique (*Encode*) el OUTPUT

Recomendaciones

"La Aplicación debe ser segura"
NO es un Requerimiento!

Los requerimientos de seguridad deben ser:
Claros, Comprensibles y Sin Ambigüedades!

Su código, es el nuevo perímetro de seguridad!!

Recomendaciones (Cont.)

- **Entrenamiento**
  - Lea el documento Top 10 de OWASP.
  - Obtenga entrenamiento especifico respecto de seguridad en aplicaciones web.
  - Aprenda como funcionan las vulnerabilidades en la vida real.
- **Política**
  - Elabore y documente los requerimientos de seguridad que deben ser cubiertos por su aplicación.
- **Diseño y Revisión**
  - Diseñe con la seguridad en mente (Gestión de Riesgos/TM)
  - Someta sus aplicaciones a procesos de revisión de código y test de intrusión en forma periódica y asegúrese que los mismos son llevados a cabo por expertos.

Recomendaciones (Cont.)

- **Cliente**
  - Demande aplicaciones web que sencillamente no incluyan ninguno de los problemas mencionados en el Top 10 de OWASP
- **Desarrollador**
  - Tome la responsabilidad de asegurar su código
- **Empresa de Desarrollo de Software**
  - Garantice que sus aplicaciones web NO poseen ninguno de los problemas mencionados en el Top 10 de OWASP
- **Profesor**
  - Deje de enseñar código inseguro
- **Gerente de Proyectos**
  - Considere la seguridad en cada parte del proyecto. Haga de las revisiones de seguridad un proceso continuo.

Referencias y Lecturas Complementarias

- **OWASP "Guide to Building Secure Webs Apps"**
  http://www.owasp.org/documentation
- **OWASP (OWASP Top Ten Project**)
  www.owasp.org/documentation/topten.html
- **Web Application Security Consortium: Clasificación de Amenazas**
  http://www.webappsec.org/projects/threat/v1/WASC_TC-1.0.spa.pdf
- **Improving Web Application Security: Threats and Countermeasures**
  http://msdn.microsoft.com/library/default.asp?url=/library/en-s/dnnetsec/html/ThreatCounter.asp
- **OWASP's Ten Most Critical Web App. Security Vulnerabilities**
  http://www.owasp.org/index.php/Image:OWASP_Top_Ten.ppt **(Jef Williams Presentation)**
- **Hernan M Racciatti – Papers & Presentations**
  http://www.hernanracciatti.com.ar
- **Writing Secure Code, Second Edition - Microsoft Press**
  (MichaelHoward / David LeBlanc) ISBN 0-7356-1722-8

Preguntas

Gracias!!

hracciatti@siclabs.com
www.siclabs.com

www.hernanracciatti.com.ar
@my4ng3l

SIC Labs
INFORMATION TECHNOLOGY
& SECURITY SOLUTION