Inseguridad de los sistemas

J. de Jesús Vázquez G.
Seguridad Informática
Banco de México

Antes de iniciar

Todas las ideas, ejemplos, propuestas son responsabilidad exclusiva de Jesús Vázquez.

En ningún momento debe considerarse que éstos corresponden a la opinión de sus empleadores, tanto en la Banca como en la Academia mexicana.

Agenda

- Objetivo
- Introducción
- Inseguridades
- Ideas
- Conclusión

Objetivo

- Presentar algunas de las inseguridades comunes de los sistemas en nuestros días, así como algunas de las ideas que se han venido proponiendo para solucionarlas.
- Que el público interesado, quien aún no se ha formado barreras o limitaciones teóricas, sea un posible implementador de las referidas soluciones.

Introducción

Introducción

Los sistemas se han desarrollado durante décadas, en el mejor de los casos la seguridad se ha considerado en las etapas finales del diseño; sin embargo, en la mayoría de estos casos la seguridad es considerada como una corrección a los desarrollos en operación, después de detectarse algún problema.

Introducción

- Algunos de estos problemas los hemos venido acarreando desde los inicios de la informática y no hemos encontrado aún una solución adecuada. Entre estos problemas figuran desde el viejo “stack overflow”, el manejo de información confidencial en sistemas no confidenciales, uso de protocolos de comunicación inseguros, hasta la ausencia total de consideraciones de seguridad en el ciclo de vida de desarrollo de software.

Introducción

- Hemos llegado a acostumbrarnos a la inseguridad considerándola algo normal. Escuchamos expresiones en los pasillos de las áreas de sistemas como las que siguen:
  - "será mejor esperar al service pack 1 antes de iniciar la migración de sistemas…"
  - "¿cuántos clientes usan sus nuevos sistemas de protección?"
  - "con estos cambios tecnológicos tan frecuentes, nunca se logra concluir la curva de aprendizaje…"

Introducción

- Por otra parte
  - Siempre ha existido la dualidad entre la seguridad y la inseguridad.
  - Es más fácil atacar que proteger:
    - El atacante sabe cuándo, cómo y dónde.
    - El defensor hace suposiciones de los 3 parámetros anteriores.
  - Un análisis de riesgos permite al defensor tratar de definir los parámetros del ataque; sin embargo, todo aquello que escape a su análisis cae en el ámbito de la inseguridad.

Introducción

- La inseguridad se ha vuelto parte de la práctica diaria, se ha institucionalizado en algunas áreas, se promueve en algunos estándares bajo nombres como:
  - Planes de contingencia
  - Recuperación de desastres
  - Continuidad operativa o de negocio
  - Respuesta a incidentes
  - Especialidades en forensia informática

Inseguridades

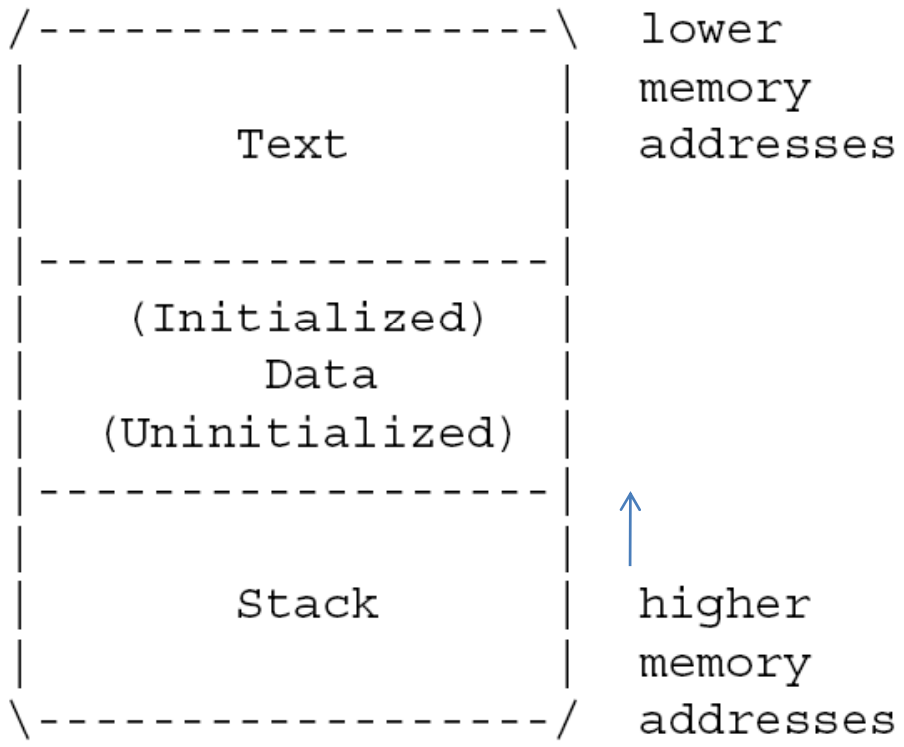
Algunos ejemplos de inseguridad

- Desbordamiento de Stack
- DoS
- Sistemas operativos discrecionarios
- Desarrollo de políticas
- Hash de código fuente
- SDLC inseguro

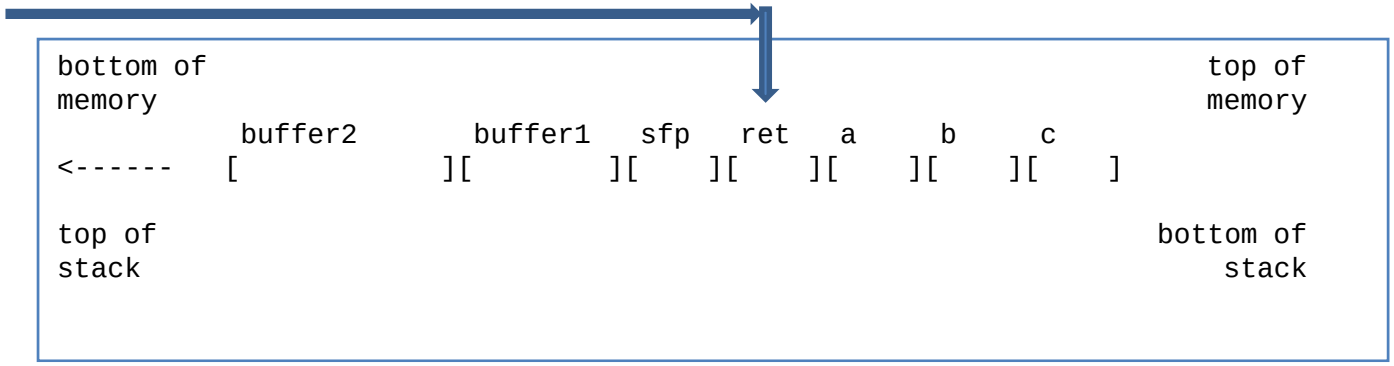
Desbordamiento de Stack

- Una de las técnicas más comunes empleadas para obtener privilegios de forma ilegal es desbordando el stack.
- Ventaja del atacante, cada versión de un sistema operativo utiliza direcciones fijas para ciertas aplicaciones, por lo que basta conocer dónde alterar el stack modificando la “dirección de regreso”, empleando esas aplicaciones, lográndose así alterar la ejecución normal.

Regiones de memoria de un proceso

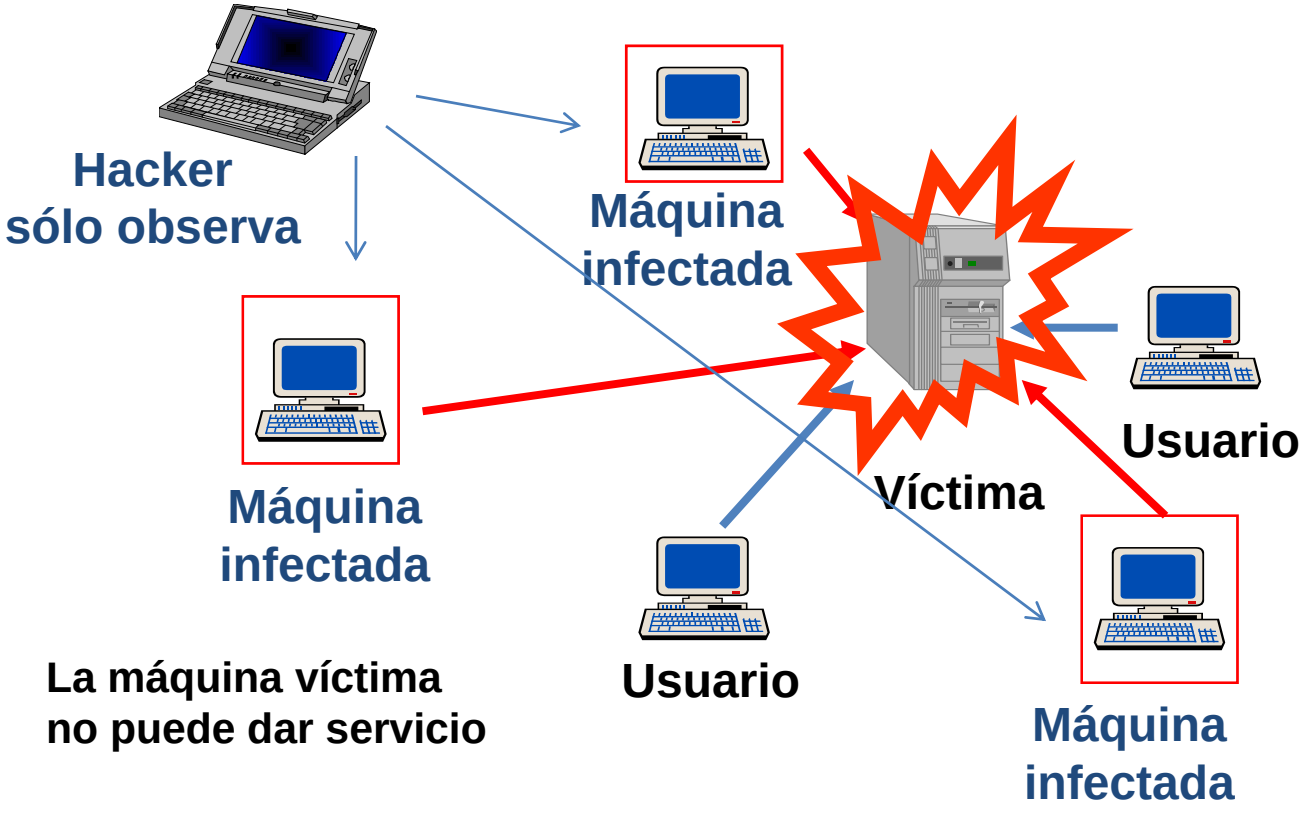


```
void function(int a, int b, int c)
{
   char buffer1[5];
   char buffer2[10];
}
void main() {
  function(1,2,3);
}
```



Ref. Aleph One

Denial of Service



Sistemas operativos discrecionarios

- Información confidencial y estratégica almacenada en sistemas discrecionarios.
- El usuario o el administrador pueden dar privilegios sobre su información, a su discreción, para que otros usuarios tengan acceso a ella.
- Es demasiado simple perder la confidencialidad, pues esos otros usuarios pueden a su vez difundir la información.

Desarrollo de políticas

- Es frecuente que las políticas de seguridad definidas en las diferentes áreas sean inconsistentes entre sí, lo que provoca ambigüedades en su aplicación en la organización.

Consistencia

- *Consistency*
  - the degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component [IEEE 90].

Necesidad de consistencia

- Los diferentes tipos de políticas de la organización pueden ser contradictorias entre sí.
- Una misma política de seguridad puede ser interpretada de manera diferente por dos implementadores, conduciendo a situaciones en conflicto.
- Las diferentes políticas que pudiesen surgir en las diferentes áreas de una organización pueden ser inconsistentes entre sí.

Necesidad de consistencia

action	ourhost	port	theirhost	port
block	*	*	Atacante	*
allow	GW	25	*	*
allow	*	*	*	25
block	*	80	*	*

Hash de código fuente

- En marzo de 2005, Xiaoyun Wang and Hongbo Yu de la Shandong University, China, publicaron la descripción del algoritmo que permite encontrar secuencias diferentes de bytes que generan el mismo hash con MD5.
- Aunado a esto, empieza a haber pruebas de posible mal uso de esta inseguridad en el referido algoritmo.

“Integridad” del código.

```
C:\TEMP> md5sum hello.exe
cdc47d670159eef60916ca03a9d4a007
C:\TEMP> .\hello.exe
Hello, world!

(press enter to quit)
C:\TEMP>
```

```
C:\TEMP> md5sum erase.exe
cdc47d670159eef60916ca03a9d4a007
C:\TEMP> .\erase.exe
This program is evil!!!
Erasing hard drive...1Gb...2Gb... just kidding!
Nothing was erased.

(press enter to quit)
C:\TEMP>
```

```
C:\OpenSSL\bin>openssl md5 hello.exe
MD5(hello.exe)= cdc47d670159eef60916ca03a9d4a007

C:\OpenSSL\bin>openssl md5 erase.exe
MD5(erase.exe)= cdc47d670159eef60916ca03a9d4a007
```

Cálculo de hash

Artículo de Peter Selinger en
http://www.mathstat.dal.ca/~selinger/md5collision/

SDLC inseguro

- Aunque empieza a haber una conciencia en el desarrollo de software seguro, la generalidad de los desarrollos aún no incluyen principios de seguridad.
- Existe la falsa creencia de que los Firewalls, IDS y VPNs protegen las aplicaciones.
- Los fabricantes de software están más preocupados por liberar nuevos sistemas que por garantizar la seguridad de los mismos.

- En el código fuente:

```
if (ioctl (rec, SIOCGIFHWADDR, &if_data) < 0) {
    perr ("can't get HW addres of my interface!\n");
    exit(1);
  }
  memcpy (myMAC, if_data.ifr_hwaddr.sa_data, 6);
  printf ("> My HW Addr: %s\n", hwaddr (myMAC));

  if (ioctl (rec, SIOCGIFADDR, &if_data) < 0) {
    perr ("can't get IP addres of my interface!\n");
    exit(1);
  }
  memcpy ((void *) &ip, (void *) &if_data.ifr_addr.sa_data + 2, 4);
  myIP = ntohl (ip);
  printf ("> My IP Addr: %s\n", inetaddr(ip));
```

- “The Department of Defense once created its own software, but today only the most highly classified code is written in-house, at places such as the secretive National Security Agency. But a good deal of code for some of the military's most sophisticated weapons--fighter aircraft and missile defense systems, for example--is written in other countries.”

- http://www.sciam.com/article.cfm?id=software-insecurity

Ideas

Algunas ideas

- Desbordamiento de Stack
- DoS
- Sistemas operativos discrecionarios
- Desarrollo de políticas
- Hash de código fuente
- SDLC inseguro
- Stack dinámico
- Anti-DOS
- Sistemas multinivel (SE Linux)
- Consistencia de Políticas
- Hash de código en ejecución
- SDLC seguro NIST SP 800-64

Desbordamiento de Stack

- Como existe el manejador de memoria, por qué no contar con un manejador de stack … “dinámico”. Por ejemplo que las direcciones fijas no sean únicas para una versión particular de un sistema operativo, sino a cada instalación de esa versión del sistema operativo.
- Desde luego que una mejor validación de las entradas a sistemas operativos y aplicaciones puede reducir considerablemente este problema.

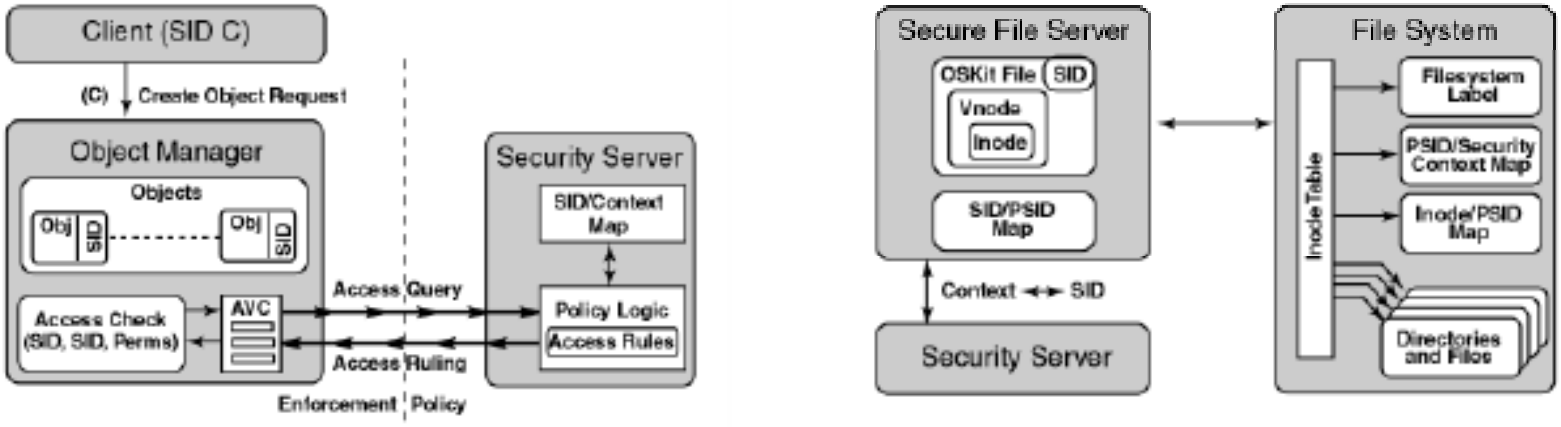
Denial of Service

- ***El principio 3) de*** *las "Guidelines for the Security of Information Systems and Networks"* de la OCDE, denominado "RESPONSE", indica que:

- ***"Participants should act in a timely and co-operative manner to prevent, detect and respond to security incidents."***
  - "Recognising the interconnectivity of information systems and networks and the potential for rapid and widespread damage, participants should act in a timely and co-operative manner to address security incidents. They should share information about threats and vulnerabilities, as appropriate, and implement procedures for rapid and effective co-operation to prevent, detect and respond to security incidents. Where permissible, this may involve cross-border information sharing and co-operation."

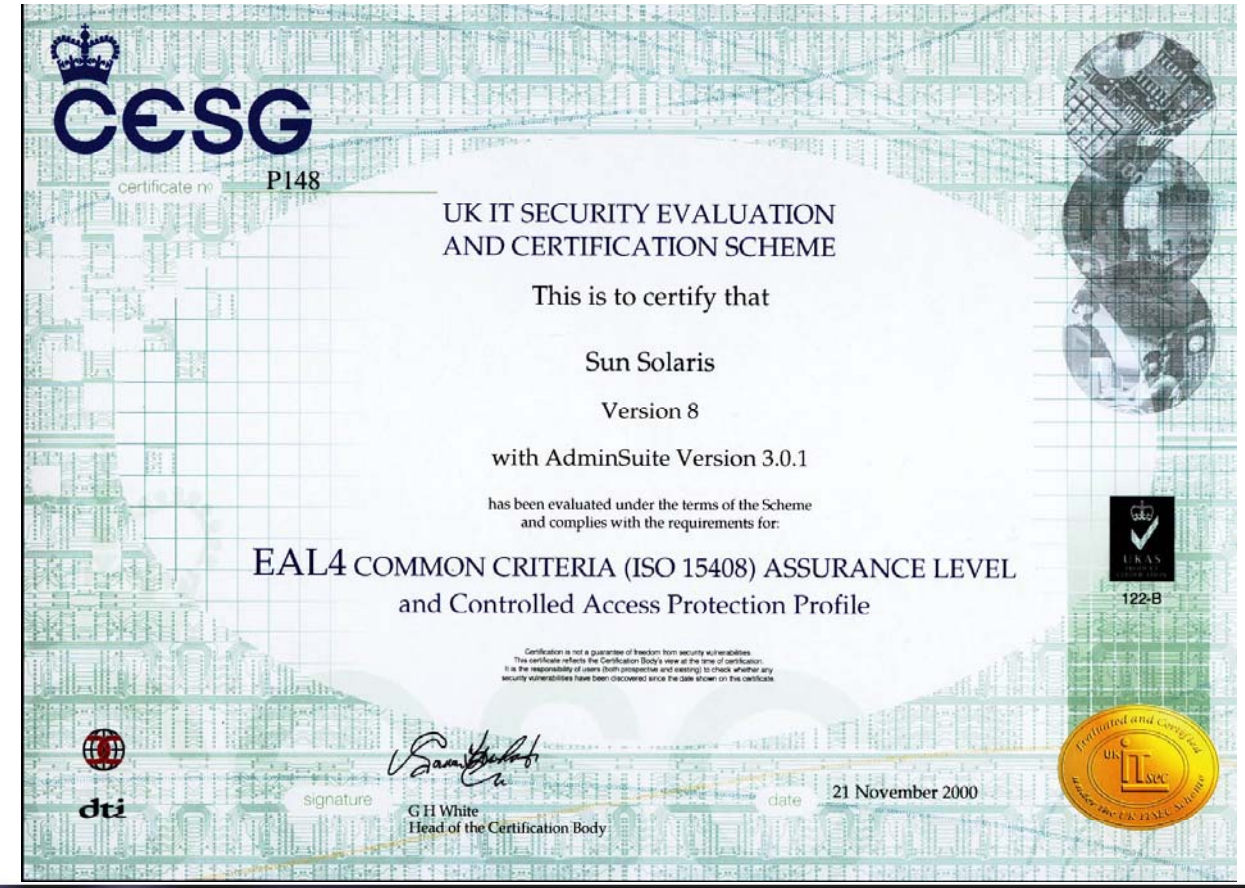
Sistemas operativos discrecionarios

- Emplear sistemas mandatorios.



Source: ***The Flask Security Architecture: System Support for Diverse Security Policies by Ray Spencer (Secure*** *Computing Corporation), Stephen Smalley, Peter Loscocco (National Security Agency), Mike Hibler, David Andersen, and Jay Lepreau (University of Utah).*

Plataformas y aplicaciones evaluadas conforme al CC


CESG

certificate no P148

UK IT SECURITY EVALUATION
AND CERTIFICATION SCHEME

This is to certify that

Sun Solaris

Version 8

with AdminSuite Version 3.0.1

has been evaluated under the terms of the Scheme
and complies with the requirements for:

EAL4 COMMON CRITERIA (ISO 15408) ASSURANCE LEVEL
and Controlled Access Protection Profile

Certification is not a guarantee of freedom from security vulnerabilities.
This certificate reflects the Certification Body's view at the time of certification.
It is the responsibility of users (both prospective and existing) to check whether any
security vulnerabilities have been discovered since the date shown on this certificate.

UKAS
122-B

dti

signature

G H White
Head of the Certification Body

date 21 November 2000

Desarrollo de políticas

- Una solución posible a la inconsistencia al definir políticas, es expresar los objetivos de seguridad de una organización en términos de especificación formal, que permita el análisis automatizado de contradicciones o incoherencias.

Ejemplo

```
% Lista de Axiomas.
%
formula_list(usable).
%  Propiedad A %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "Todo empleado puede accesar la informacion de la empresa"
%
  (all a ( Empleado(a) ->
             (all b (Informacion(b) & Accesa(a,b) ) ) ) ).
```

```
% Propiedad B %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "Todo acceso a la informacion de la empresa, por parte
% de un empleado, requiere un password"
%
  (all a all b ((Empleado(a) & Informacion(b) & Accesa(a,b)) ->
                  (exists c (Password(c) & Posee(a,c) )))).
```

```
% Propiedad C %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "Todo Barrendero es un empleado de la empresa"
%
  (all a ( Barrendero(a) -> Empleado(a))).
```

```
% Propiedad D %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% "Todos los barrenderos no poseen passwords"
%
  (all a ( Barrendero(a) ->
                -(exists b (Password(b) & Posee(a,b))))).


end_of_list.
```

```
% Lista de Soporte
%
formula_list(sos).
 (exists a Informacion(a)).
 (exists a Empleado(a)).
 (exists a Barrendero(a)).
 (exists a Password(a)).
end_of_list.
```

```
---------------- PROOF ----------------

1 [] -Empleado(x1)|Informacion(x2).
2 [] -Empleado(x1)|Accesa(x1,x2).
3 [] -Empleado(x3)| -Informacion(x4)|
          -Accesa(x3,x4)|Password($f1(x3,x4)).
4 [] -Empleado(x3)| -Informacion(x4)|
          -Accesa(x3,x4)|Posee(x3,$f1(x3,x4)).
5 [] -Barrendero(x5)|Empleado(x5).
6 [] -Barrendero(x6)| -Password(x7)| -Posee(x6,x7).
8 [] Empleado($c2).
9 [] Barrendero($c3).
12 [ur,8,1] Informacion(x).
13 [ur,9,5] Empleado($c3).
15 [ur,13,2] Accesa($c3,x).
18 [ur,15,4,13,12] Posee($c3,$f1($c3,x)).
19 [ur,15,3,13,12] Password($f1($c3,x)).
24 [ur,18,6,19] -Barrendero($c3).
25 [binary,24.1,9.1] $F.

------------ end of proof -------------
```

Hash de código fuente

- El ejemplo presentado por Peter Selinger, sumado al problema de la inseguridad en los desarrollos, el hecho de que un desarrollador puede incrustar código malicioso ¿cómo reducir la inseguridad de los desarrollos de software?
- Verificación de código en ejecución...
- Revisiones de “caja blanca” sobre el código

SDLC inseguro

- La seguridad debería ser parte integral del Ciclo de Vida de Desarrollo de Software.
- Deberían existir responsabilidades aplicables legalmente a los desarrolladores de software, de modo que se obliguen a cumplir con ciertas evaluaciones o certificaciones de seguridad antes de liberar nuevos productos.

Security Checks Along SDLC

Vulnerability Checks	SDLC Phases	Maturity of Tools, Practices	Injected Vulnerabilities (Not Necessarily Security)
Vulnerabilities in requirements, business processes flow, algorithms	Analysis	Embryonic	15%
Vulnerabilities caused by interrelations of modules and (Web) services, logic and data flow	Design	Embryonic	40%
Vulnerabilities in language instructions, implementation of logic and data flow	Construct	Low	35%
Vulnerabilities in executables, UI. Assembly of secure services could be insecure	Testing	Low	10%
Missing patches, administrative errors, misconfiguration. If vulnerability found — back to analysis	Operations	Low-Medium	

Gartner

NIST SP800-64

	Initiation	Acquisition / Development	Implementation	Operations / Maintenance	Disposition
SDLC	- Needs Determination: • Perception of a Need • Linkage of Need to Mission and Performance Objectives • Assessment of Alternatives to Capital Assets • Preparing for investment review and budgeting	- Functional Statement of Need - Market Research - Feasibility Study - Requirements Analysis - Alternatives Analysis - Cost-Benefit Analysis - Software Conversion Study - Cost Analysis - Risk Management[7] Plan - Acquisition Planning	- Installation - Inspection - Acceptance testing - Initial user training - Documentation	- Performance measurement - Contract modifications - Operations - Maintenance	- Appropriateness of disposal - Exchange and sale - Internal organization screening - Transfer and donation - Contract closeout
SECURITY CONSIDERATIONS	- Security Categorization - Preliminary Risk Assessment	- Risk Assessment - Security Functional Requirements Analysis - Security Assurance Requirements Analysis - Cost Considerations and Reporting - Security Planning - Security Control Development - Developmental Security Test and Evaluation - Other Planning Components	- Inspection and Acceptance - System Integration - Security Certification - Security Accreditation	- Configuration Management and Control - Continuous Monitoring	- Information Preservation - Media Sanitization - Hardware and Software Disposal

Conclusión

Conclusión

- La inseguridad de los sistemas no se puede eliminar del todo; sin embargo, es necesario que la academia y la industria reconsideren sus causas, que se han presupuesto como irremediables en los últimos años, pues algunas de ellas pudieran ser ahora tratables.
- Los responsables de la seguridad informática debemos considerar en la estrategia que proponemos a las organizaciones que apoyamos, la existencia de las inseguridades, así como el análisis de sus causas para complementar y hacer mejor nuestra propuesta.

Referencias

- Smashing The Stack For Fun And Profit By Aleph One
- Building Secure Applications, Joseph feiman, Gartner, 2007
- MD5 Collision Demo, Peter Selinger, 2007, http://www.mathstat.dal.ca/~selinger/md5collision/

Referencias

- Institute of Electrical and Electronics Engineers. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* New York, NY: 1990.
- Security Considerations in the Information System Development Life Cycle, NIST SP 800-64 Rev 1
- OECD Guidelines for the Security of Information Systems and Networks, 2002.

Gracias …

jjesusvg@banxico.org.mx