



Jornada de desarrollo seguro

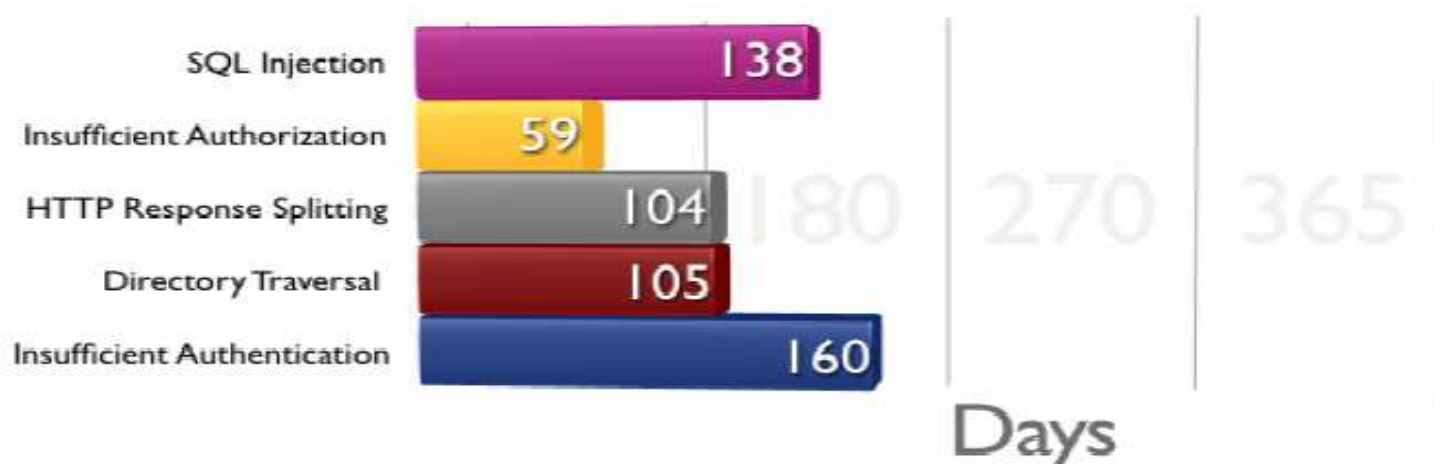
Zamudio, 15-IV-2010

- ❖ **Introducción**
- ❖ **Prevención de vulnerabilidades**
 - ✓ **Modelado de Amenazas**
 - ✓ **Modelado de Vulnerabilidades**
 - ✓ **Modelado de Actividades de Seguridad**

- ❖ **Errores durante la fase de codificación**
 - ✓ **Analizadores dinámicos/estáticos**
 - ✓ **Herramientas de test y de cobertura**
 - ✓ **Herramientas de Profiling**



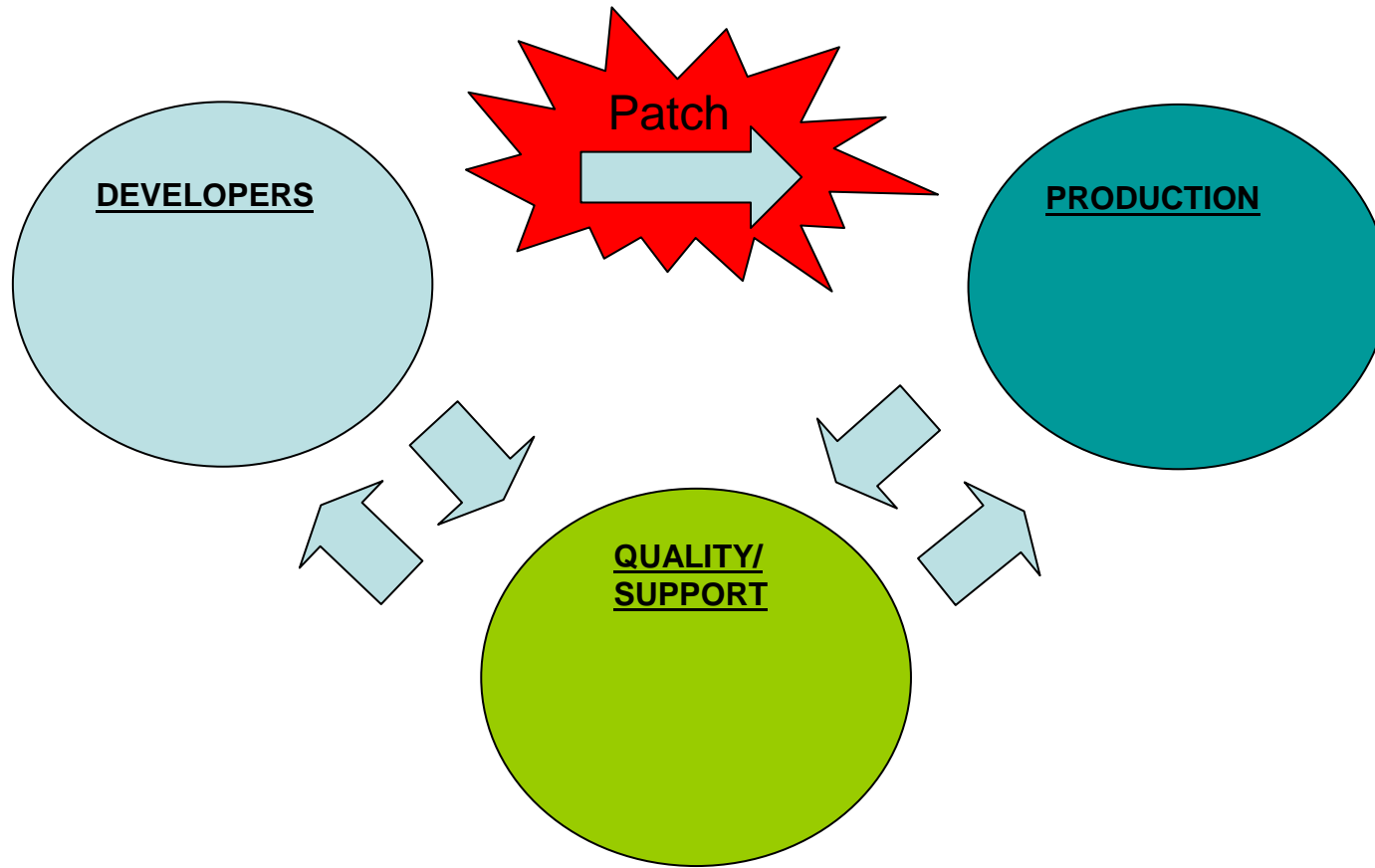
- ❖ Media # de días para la corrección de las 5 vulnerabilidades mas severas y URGENTES



- ❖ Identificación de las vulnerabilidades no es el problema
- ❖ Exploit esta disponible con una media de 6 días²

1 – Whitehat Website Security Statistics Report, March 2008

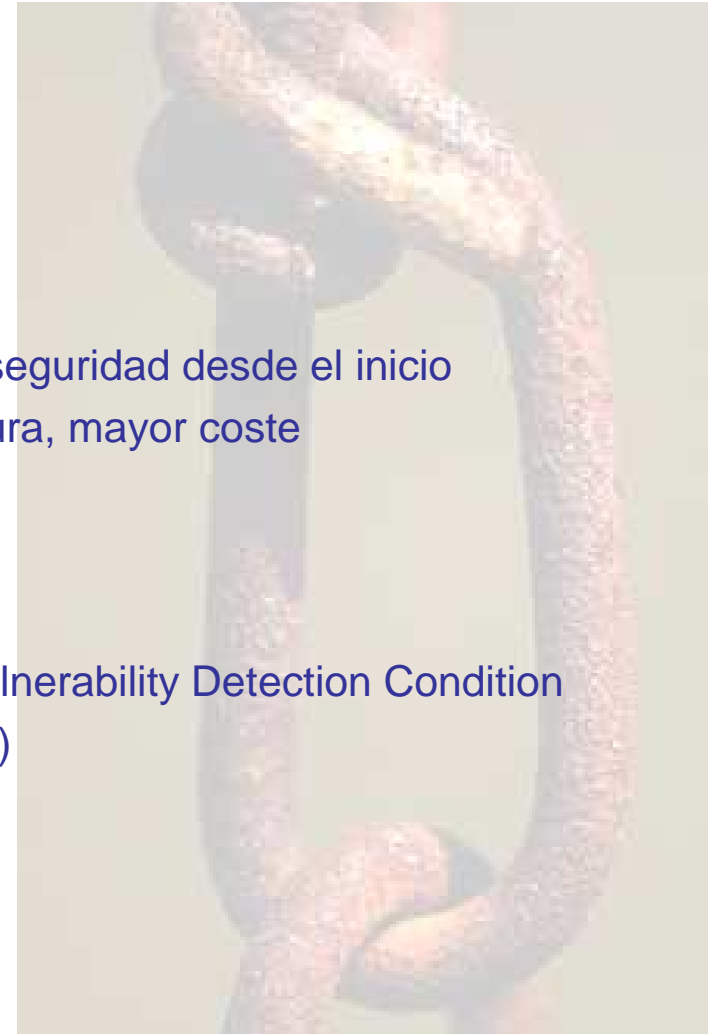
2 – Symantec Internet Security Threat Report, H3, 2007



- ❖ La corrección de bugs al modo tradicional lleva demasiado tiempo...

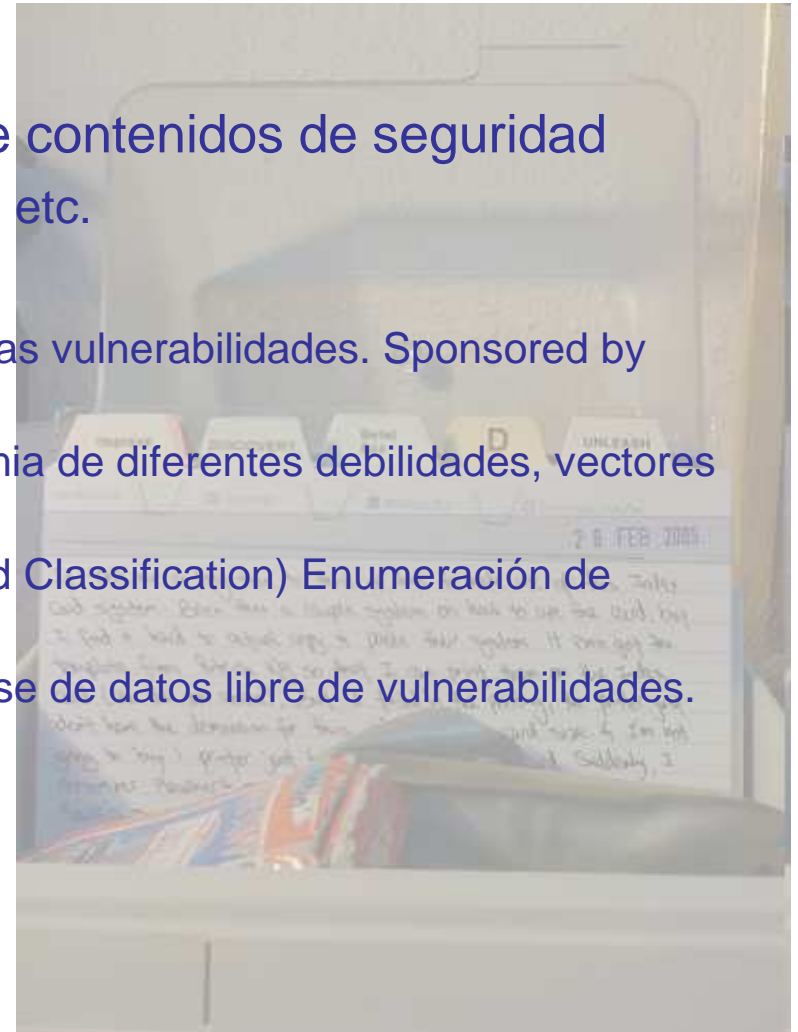
- ❖ Que es el desarrollo seguro
 - ✓ No existe el software 100% seguro
 - ✓ Pero tenemos que tener en cuenta restricciones de seguridad desde el inicio
 - ✓ A mayor tardanza al insertar un cambio de arquitectura, mayor coste

- ❖ Necesitamos nuevos conceptos:
 - ✓ Indices de vulnerabilidades
 - ✓ Misuse case, Attack tree ,Security Activity Graph, Vulnerability Detection Condition
 - ✓ Seguramente no necesitemos todos, pero ayudan ;-)

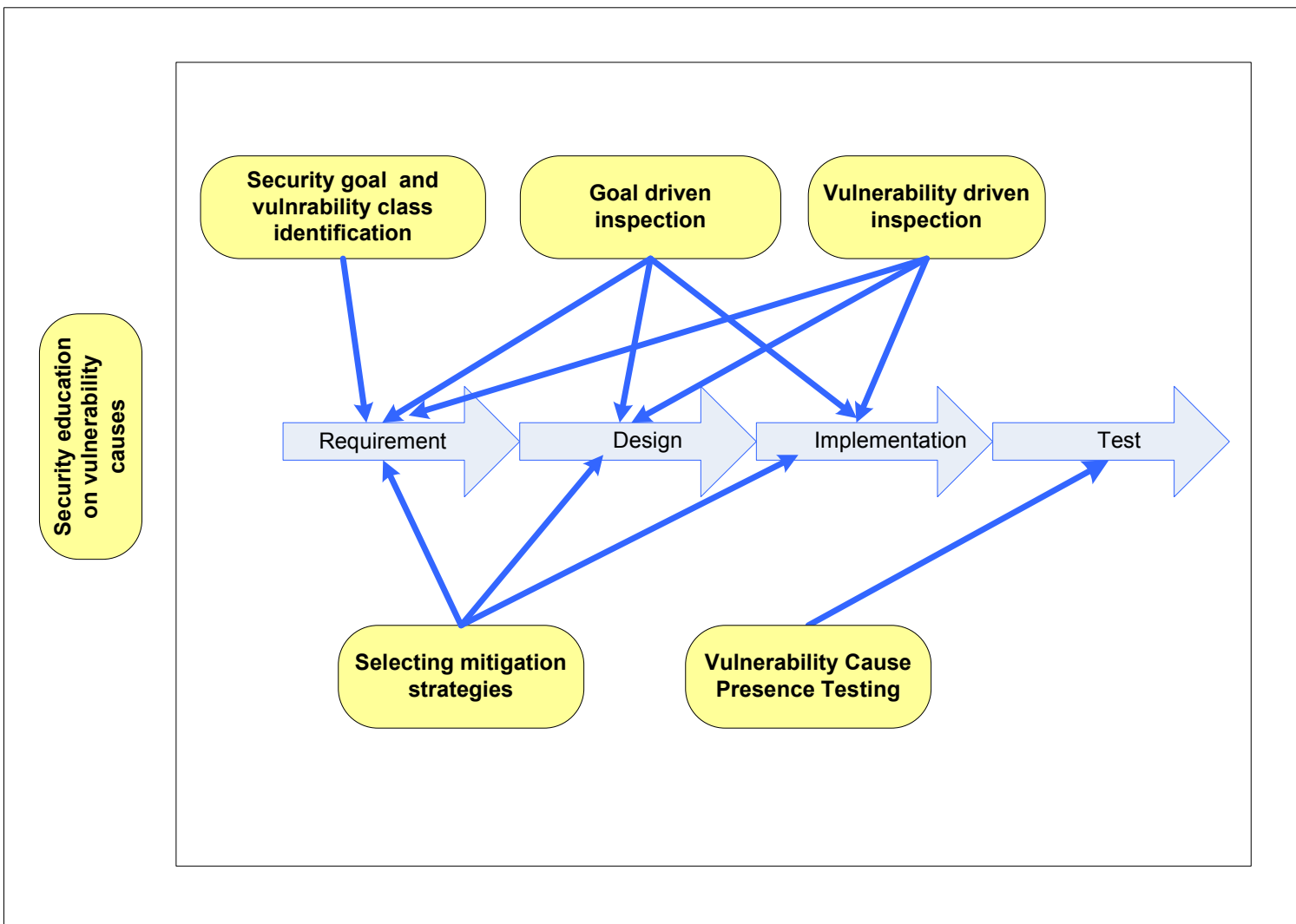


- ❖ El desarrollo introduce *per se* amenazas y vulnerabilidades
- ❖ Objetivo:
 - ❖ Detectar de forma sistemática errores de diseño
 - ❖ Insertar la seguridad a lo largo del ciclo de desarrollo

- ❖ Surgen como respuesta a la segregación de contenidos de seguridad
 - ✓ Mucha información en foros, sitios web, IRC, etc.
- ❖ Principales puntos de referencia:
 - ✓ CVE (Common Vulnerability Database) Enumera las vulnerabilidades. Sponsored by DoD
 - ✓ CWE (Common Weakness enumeration) Taxonomía de diferentes debilidades, vectores de ataque, etc. Sponsored by DoD
 - ✓ CAPEC (Common Attack Pattern Enumeration and Classification) Enumeración de patrones de ataque. Sponsored by DoD
 - ✓ OSVDB (Open Source Vulnerability Database) Base de datos libre de vulnerabilidades.



Development phases



- ❖ Que es el modelado de vulnerabilidades
 - ✓ Crear de una manera formal las **CAUSAS** tque hacen que una vulnerabilidad aparezca

- ❖ Quiero mantener el conocimiento en 127.0.0.1
 - ✓ Vamos a ver cómo...

❖ Misuse case:

- ✓ Nuevos conectores: *mitigates, threatens, detect, prevent*
- ✓ Nuevos casos de uso: *Threat, Vulnerability*
- ✓ Nuevos actores: *Crook, Insider*

❖ Seamonster:

- ✓ Desarrollado SINTEF → Centro de i+D Noruego
- ✓ Basado en eclipse
- ✓ Facilidad de uso
- ✓ LGPL v3

<http://seamonster.wiki.sourceforge.net>

❖ Attack Tree:

- ✓ Foco en los atacantes, sus objetivos y como conseguirlos
- ✓ Primero, identifico mis activos porque van a ser su objetivo
- ✓ Relacionado con CAPEC y CWE

- ❖ Demo de SeaMonster



❖ Diferentes maneras de encarar esto;

- ✓ Security Goal Inspector Tree: Foco en la inspección de la búsqueda de objetivos de seguridad
- ✓ Security Activity Graph: Foco en las salvaguardas para prevenir determinada vulnerabilidad
- ✓ Vulnerability Detection Condition: Foco en técnicas de detección

- ❖ Herramienta para inspecciones manuales de software
 - ✓ No necesariamente código, sino cualquier tipo de documento

- ❖ El Objetivo de seguridad está asociado a indicadores que proveen evidencias de la correcta realización de determinado tipo de documento

- ❖ Cada Objetivo se expresa de una forma positiva
 - ✓ Si el objetivo es complejo, lo explotamos

- ❖ La idea subyacente es mitigar los problemas de seguridad
 - ✓ Prevenir vulnerabilidades de software
- ❖ Describe actividades de seguridad para prevenir vulnerabilidades de software
 - ✓ Como se deben realizar las actividades
 - ✓ Como comprobar la ausencia de vulnerabilidades

- ❖ Foco en las vulnerabilidades

- ✓ Cuales son las causas que me derivan en determinada vulnerabilidad

- ❖ Lo vemos mejor con un ejemplo

❖ Herramienta GOAT

- ❖ Desarrollada por la universidad de Linköpings (Suecia)
- ❖ Diferentes tipos de formalismos
- ❖ Conexión con repositorios externos
- ❖ Release GPLv3 en Junio 2010

Mas información:
<http://shields-project.eu>

- ❖ Vulnerabilidades
- ❖ Análisis de código fuente
 - ✓ Estático
 - ✓ Dinámico
 - ✓ Profiling
- ❖ Análisis de código fuente
 - ✓ Test Framework
 - ✓ Cobertura

- ❖ Porque no solo poner seguridad perimetral o de servidor?
 - ✓ SAAS como paradigma actual de desarrollo
 - ✓ Es complementario

- ❖ El eslabón mas débil de una cadena determina la fortaleza de la cadena completa
 - ✓ *“Two pieces of code put together, one with a limited spec for strong data typing, and the other with weak handling of output, result in a new set of behaviors that fail to meet specification, though each unit of code individually meets it's own specification”.*
– Arian Evans

- ❖ Las debilidades se manifiestan mas contundentemente en sistemas no controlados
 - ✓ Cloud computing
 - ✓ System of System

MEMORY ACCESS

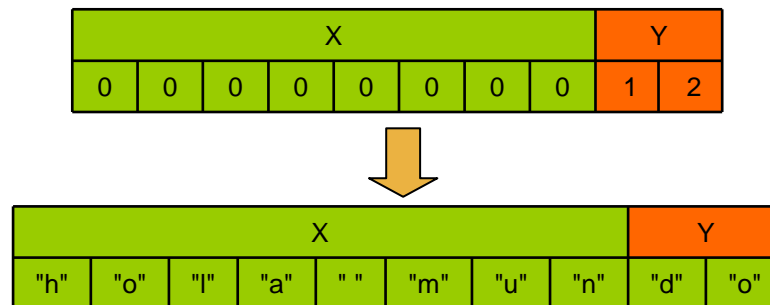
Buffer overflows

❖ Que?

- ✓ Ocorre cuando una aplicación escribe mas información en el buffer que el espacio que tenia alojado en memoria para este buffer.
- ✓ no de los mas famosos, en parte por las consecuencias tan desastrosas que ha tenido a lo largo del tiempo

❖ Como?

- ✓ Cuando el buffer es desbordado la memoria adyacente es sobrescrita



MEMORY ACCESS

Buffer overflows

❖ Ej:

```
#include <stdio.h>
void function (char *param) {
    char c[12];
    memcpy(c, param, strlen(param)); // no bounds checking...
}
int main (int argc, char **argv) {
    function(argv[1]);
}
```

- ❖ Lenguajes de alto nivel ya poseen sus propias técnicas para impedir esta vulnerabilidad
- ❖ Otros lenguajes (C, C++) poseen tecnología en tiempo de compilación que proveen protección de esta vulnerabilidad
 - ✓ Stack-Smashing Protector (ProPolice)
 - ✓ StackGuard

INPUT Format String

❖ ¿Que?

- ✓ La ejecución de una entrada a nuestro programa genera:
 - Comportamientos inesperados
 - Compromete la seguridad y la estabilidad de todo el sistema

❖ ¿Como?

- ✓ La aplicación no valida correctamente las entradas
 1. El parámetro Format String (%x) es insertado.
 2. La cadena es pasada a la función que la parsea y ejecuta las conversiones
 3. Puede incluso no recibir ningún parámetro como entrada, y si esos argumentos no son proporcionados la función podría incluso leer o escribir en la pila

INPUT

Format String

❖ Ej:

```
char buf [100]
int x = 1
snprintf ( buf, sizeof buf, argv [1] ) ;
buf [ sizeof buf - 1 ] = 0
printf ( "Buffer size is: (%d) \nData input: %s \n" , strlen (buf) , buf ) ;
return 0 ;
```

❖ Se inserta código como entrada a una función

```
./formattest "Bob %x %x"
```

❖ El código resultante es el siguiente

```
printf ( "Buffer size is: (%d) \n Data input: Bob %x %x \n" , strlen (buf) , buf ) ;
```

❖ Cuando la aplicación imprime los resultados e interpreta la cadena imprime el contenido de una zona de memoria

Concurrency

Race conditions

❖ Que?

- ✓ “Una condición de carrera ocurre cuando un programa no funciona como se “supone” debido a una ordenación de eventos no prevista y que produce un desacuerdo de acceso sobre un recurso compartido”-- David Wheeler --
- ✓ Las condiciones de carrera son unas de las vulnerabilidades mas comunes relativas a la programación concurrente y quizás una de las mas difíciles de detectar y corregir.

❖ Como?

- ✓ Esto ocurre principalmente debido al concepto de paralelismo
 - las operaciones no son atómicas
 - tanto los procesos como los hilos pueden ejecutarse de manera simultanea
- ✓ El flujo de una aplicación suele depender de eventos de tiempo, un programador difícilmente puede tener en mente todos los caminos y suele asumir un flujo de eventos que suele ser el incorrecto (por ejemplo que SIEMPRE un evento seguirá a otro, cuando no es realmente lo que ocurre)
- ✓ No siempre aparece , depende la probabilidad que esta aumentando:
 - Aumento de la potencia hardware (sobre todo CPU)
 - Tendencia a programación multi-hilo y multi-proceso

Source code analysis Static

❖ Def:

- ✓ El proceso de ejecutar una herramienta de escaneo sobre código fuente (C#, Java, VB.NET, etc.) para revelar problemas de seguridad y de calidad de código
- ✓ El proceso de mirar el código fuente desde una perspectiva estática.

❖ Tools:

✓ Open source

- Splint -- <http://splint.org/>
- Uno -- <http://spinroot.com/uno/>

✓ Commercial

- Prevent - Coverity -- <http://www.coverity.com/products/prevent.html>
- Insight - KlocWork -- <http://www.klocwork.com/products/insight.asp>
- PC Lint & Flexelint - Gimpel -- <http://www.gimpel.com/html/products.htm>
- Prevent - Coverity -- <http://www.coverity.com/products/coverity-prevent.html>

Source code analysis Dynamic

❖ Def:

- ✓ “Es el análisis de software que es realizado ejecutando programas creados de esos sistemas en procesadores tanto reales como virtuales. Para que un análisis dinámico sea efectivo, el programa debe ser alimentado con suficientes datos de entrada como para que el comportamiento del programa sea interesante” – Wikipedia –

- ✓ Open source
 - Valgrind – <http://valgrind.org>
 - Electric Fence -- <http://perens.com/FreeSoftware/ElectricFence>

- ✓ Commercial
 - Purify IBM <http://www.ibm.com/software/awdtools/purify>
 - Insure++ PARASOFT <http://www.parasoft.com/Insure>

Source code analysis Profiling

❖ Def:

- ✓ Mide el comportamiento de un programa al ser ejecutado
- ✓ *“Medición es un componente crucial de la mejora de rendimiento, sin embargo la intuición es una buena guía que debe ser complementada con herramientas”*
- Kernighan and Pike –
- ✓ *“Optimización prematura es el camino del fracaso”*
- ✓ -- Knuth's --

❖ Tools:

- ✓ Open source
 - Oprofile -- <http://oprofile.sourceforge.net/>
- ✓ Commercial
 - Purify Plus – IBM -- <http://www.ibm.com/software/awdtools/purifyplus/>
 - VTune – INTEL -- <http://software.intel.com/en-us/intel-vtune/>

Post Development analysis Test Framework

❖ Def:

- ✓ “Testing es un `proceso sin fin`, así que uno debe establecer el criterio para parar antes de comenzar”
-- Mohammad M. Basha --

❖ Types:

- ✓ *Black box*: trata el software sin ningún conocimiento interno de implementación
- ✓ *White box*: se dispone de conocimiento de las estructuras internas y algoritmos e incluso a código fuente

❖ Frameworks:

- ✓ Open source
 - Check -- <http://check.sourceforge.net/>
 - JUnit -- <http://www.junit.org/>
- ✓ Commercial
 - TestUff - <http://www.testuff.com/>
 - Cantana – IPL -- <http://www.ipl.com/products/tools/pt400.uk.php>

Post Development analysis Coverage

❖ Def

- ✓ Evalúa el código cubierto por los casos de usos
- ✓ Cuantifica cuanto código es cubierto por los tests => calidad de tests
- ✓ Elimina agujeros en los test
- ✓ Estaba aquí desde 1963 => casi la primera técnica de test sistemático de software

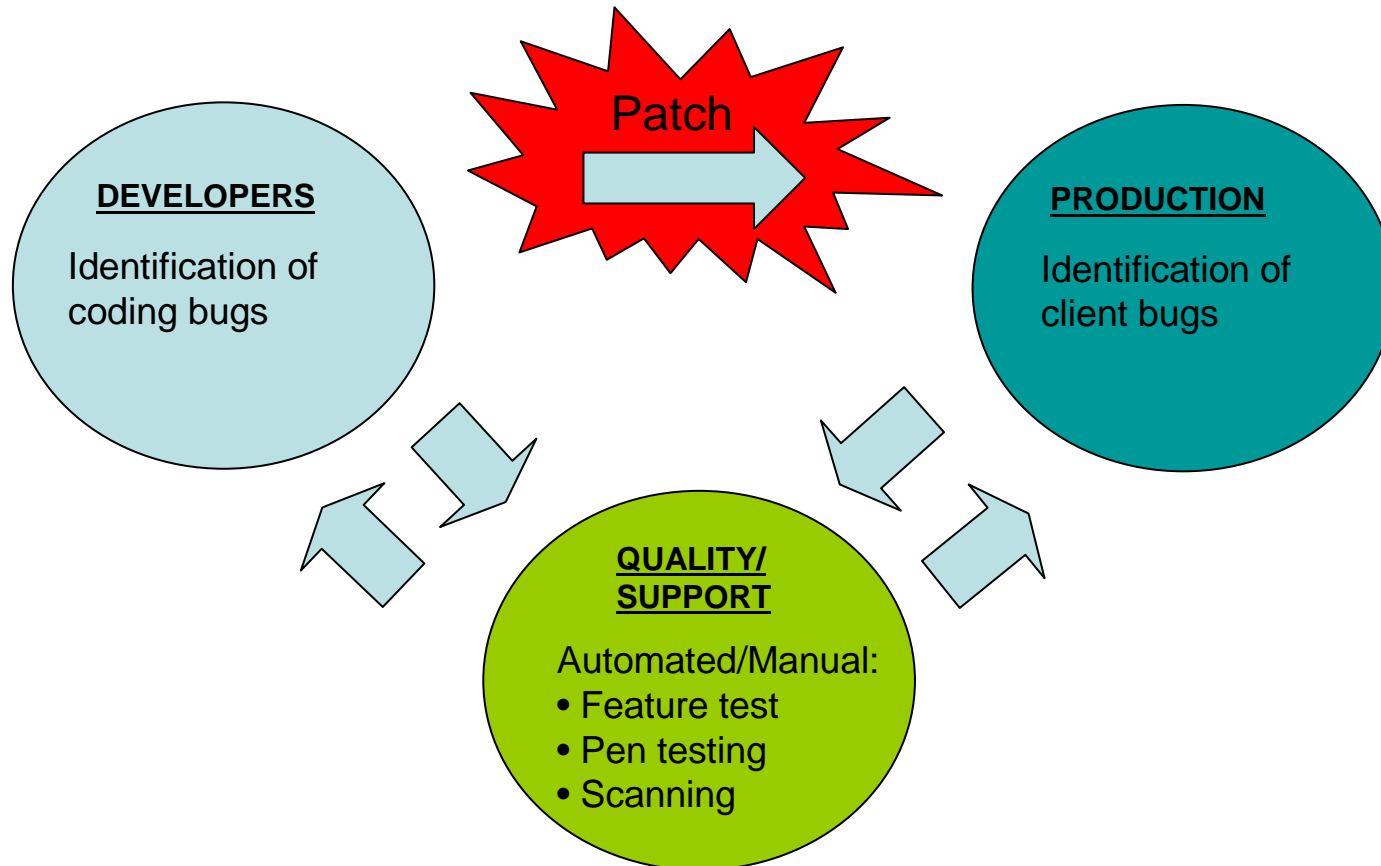
❖ Tools:

- ✓ Open Source
 - Gcov -- <http://gcc.gnu.org/onlinedocs/gcc/Gcov.html>
 - EMMA -- <http://emma.sourceforge.net/>
- ✓ Commercial
 - Insure++ PARASOFT <http://www.parasoft.com/Insure>

- ❖ Def
 - ✓ Permite automatizar lo mas posible el desarrollo de software
 - ✓ Integra herramientas de desarrollo y de calidad
 - ✓ Muy flexibles

- ❖ Tools:
 - ✓ Open Source
 - HUDSON-- <http://wiki.hudson-ci.org>
 - BUILDBOT-- [http:// buildbot.net/](http://buildbot.net/)

Soucion (I)




❖ Traditional code fixes take too long...

- ❖ Traditional code fixes take too long...
- ❖ Código C
 - ❖ Desarrollo continuo BUILDDBOT
 - ❖ Analisis estatico PC-LINT
 - ❖ Analisis dinamico Valgrind
 - ❖ Test C-Test
 - ❖ Coverage – GCOV
 - ❖ Profiling - oprofile
- ❖ Código JAVA
 - ❖ Desarrollo continuo HUDSON
 - ❖ Framework Eclipse TPTP
 - ❖ PMD
 - ❖ Checkstyle
 - ❖ Profiling
 - ❖ Coverage
 - ❖ Junit

- ❖ Diseñadas para detectar agujeros de seguridad en un sistema entero
 - ✓ Frecuentemente sirven como ayudas para una porción del código
 - ✓ Excesivamente complejas en algunas ocasiones

- ❖ Demasiadas herramientas donde escoger
 - ✓ Algunas herramientas están incluyendose dentro de los IDEs
 - ✓ Vulnerabilidades pueden ser detectadas durante la fase de desarrollo

- ❖ Los requisitos de seguridad de la mayoría de los proyectos NUNCA cubren los aspectos de seguridad del código
 - ✓ Difícil medir la calidad
 - ✓ El uso de muchas de éstas técnicas depende del desarrollador



Parque Tecnológico, # 204
E-48170 Zamudio
Bizkaia (Spain)
Tel.: +34 94 420 95 19
Fax: +34 94 420 94 20
www.esi.es

Txus Sanchez
R&D Projects
jesus.sanchez@esi.es
Borja Urkizu
R&D Projects
Borja.Urkizu@esi.es

