Seguridad en un juez de maratones de programación

Jhon Jimenez
Manuel Pineda

UNIVERSIDAD TECNOLÓGICA DE PEREIRA

30 de noviembre de 2013

Contenido I

Contenido II

Maratones

Maratones de Programación

- Descripción
- Proceso de juzgamiento
- Iniciativa
- Problema

Descripción

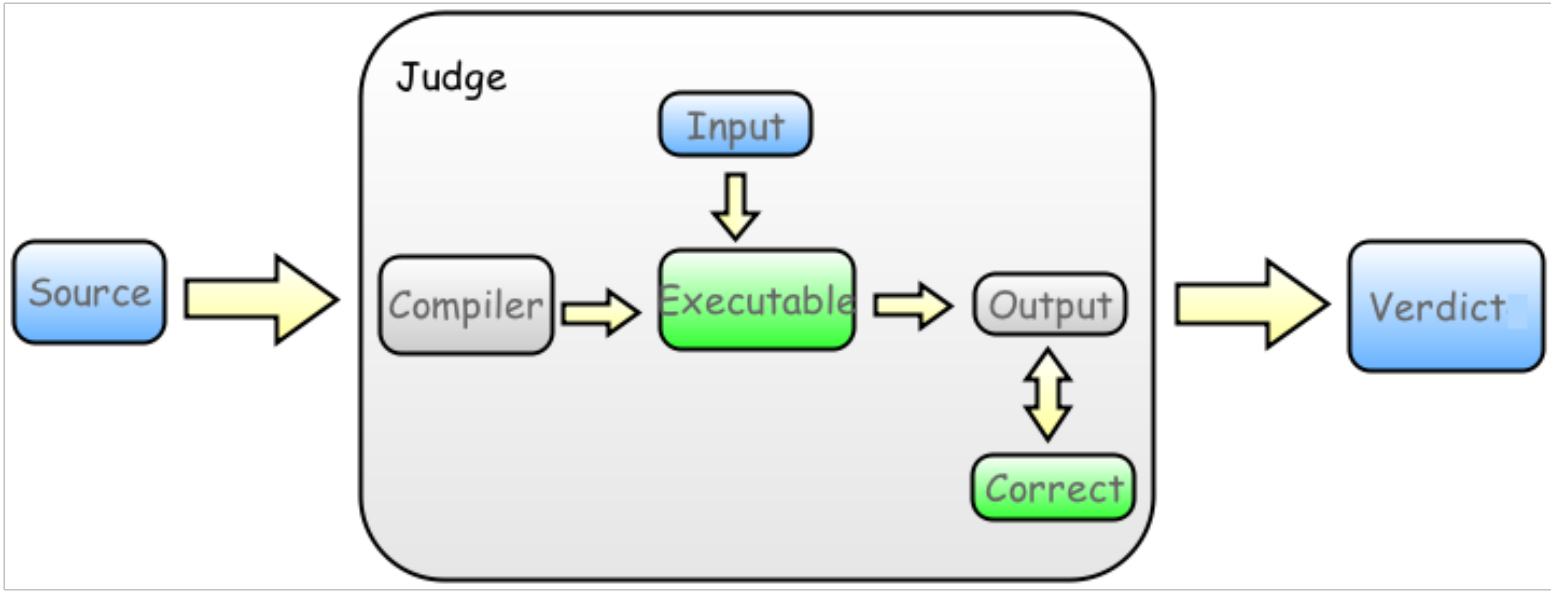
Maratones de Programación

Las maratones de programación son competencias en las cuales se intentan resolver varios problemas relacionados con algoritmia, matemáticas, geometría, entre otros.

Juez de Maratones

Es un programa encargado de calificar las soluciones enviadas por los participantes respondiendo a el tiempo de ejecución, la memoria consumida, errores en tiempo de ejecución, ...

Proceso de juzgamiento



Iniciativa

La idea nace en el semillero in-silico de la UTP con el fin de ayudar en el proceso de las clases relacionadas con programación.

También se buscaba fortalecer los entrenamientos del semillero de programación dando la posibilidad de crear maratones propias.

Problema

- El proceso de juzgamiento trae consigo un problema de seguridad bastante importante, pues no se tiene control sobre lo que se puede escribir en un código fuente escrito por un tercero.
- La disponibilidad del juez durante una competencia es crucial.
- Se deben limitar los recursos usados por el competidor para garantizar las restricciones del problema. Las más comunes son:
  - Tiempo de ejecución.
  - Memoria utilizada.
  - Tamaño de archivos creados

Clasificación de ataques

- En tiempo de compilación
- En tiempo de ejecución
- En tiempo de competencia

Clasificación de Ataques

Forzar un alto consumo de tiempo durante la compilación

Este tipo de ataque se da cuando un programa usa un tiempo indeterminado durante la compilación.

Acceso a material restringido durante la ejecución

Busca como objetivo acceder a recursos no autorizados. Por ejemplo durante una competencia el atacante podria buscar el acceso a los casos de prueba.

Clasificación de Ataques

Saltarse la medición del tiempo (*Time Limit*)

Con este tipo de ataque se consigue vulnerar el tiempo que tiene de ejecución un determinado problema.

Modificando o dañando el entorno de pruebas

Busca como propósito vulnerar el entorno de pruebas, encargado de comparar las soluciones de los algoritmos enviados por los participantes

Clasificación de Ataques

Uso indebido de la red

Si alguien puede monitorear y almacenar el tráfico de la red del sistema de juzgamiento se puede presentar la oportunidad de la intercepción de envíos de otros participantes para posteriormente enviarlos como propios.

Uso indebido de recursos adicionales

Es un ataque que se presenta principalmente en competencia organizadas en un mismo sitio. Un ejemplo es el uso indebido de la impresión.

En tiempo de Compilación

Ejemplo de ataque usando directivas de preprocesamiento.

Código Ataque (Tiempo de compilación)

```
#include __FILE__

int main(){
    return 0;
}
```

En tiempo de Compilación

Ejemplo de ataque usando directivas de preprocesamiento.

Código Ataque (Tiempo de compilación)

```
#include "/dev/stdin"

int main(){
    return 0;
}
```

En tiempo de Ejecución

Ejemplo de ataque haciendo un fork bomb.

Código Ataque (Tiempo de ejecución)

```
#include <unistd.h>

int main(){
    while (1) fork();
    return 0;
}
```

En tiempo de Competencia

- Monitorear y almacenar el tráfico de red.
- Comunicación a través de la red entre los participantes.
- Mal uso de contraseñas.
- Intentar atacar a la máquina que realiza los juzgamientos.
- Intentar atacar la aplicación web (DoS).

¿Sandbox?

Cuando en el campo de la seguridad informática se habla de sandbox, se está refiriendo a un mecanismo de seguridad para la ejecución de programas de una manera segura y aislada de otros procesos.

Enfoques

- Chroot
- Virtualización
- Ambiente controlado

Chroot

Es una operación que ejecuta un proceso, pero cambia para este y todos sus hijos el directorio raíz del sistema. Comúnmente cuando se utiliza este enfoque de sandboxing se le conoce como “jaula chroot”.

- Se deben permitir las operaciones entre librerías.
  - Debootstrap.
- Cambio de contexto y tiempo de respuesta.
- No es suficiente

Virtualización

En este enfoque se permite que un sistema operativo llamado cliente se ejecuta dentro de otro llamado sistema anfitrión.

- Vuelve lento el sistema de juzgamiento.
- ¿Qué pasa si la máquina virtual ya no está disponible?.
- No es suficiente.

Ambiente controlado

Un ambiente controlado es básicamente ejecutar un programa bajo "la supervisión"de otro programa.

RLIMIT

```
int getrlimit(int resource,struct rlimit *rlim);
int setrlimit(int resource,const struct rlimit *rlim);

struct rlimit {
    rlim_t rlim_cur;
    rlim_t rlim_max;
};
```

RLIMIT: Es una directiva del sistema que permite controlar procesos a nivel del kernel.

- RLIMIT_CPU : Controla el tiempo de ejecución del programa.
- RLIMIT_DATA : Tamaño máximo del segmento de datos en el programa (datos inicializados, datos no inicializados y heap).
- RLIMIT_FSIZE : Tamaño máximo de archivos que el programa puede crear.
- RLIMIT_NPROC : Máximo número de procesos hijos.
- RLIMIT_STACK : Máximo tamaño del stack.

Construyendo un mini-safeexec

- Códigos de pruebas, tiempo, memoria, forkbomb.
- Limitando recursos con rlimit.

Permisos

Una configuración incorrecta de los permisos puede comprometer seriamente el funcionamiento del servidor. Algunas preguntas a tener en cuenta son:

- ¿Qué usuario compila y ejecuta el código?
- ¿Qué archivos y carpetas son importantes?
- ¿Quién puede leer los archivos de configuración?

¿Accepted?

Código Ataque (Tiempo de ejecución)

```
#include<iostream>
#include<stdlib.h>
using namespace std;

int main(){
    string line;
    while(cin >> line);
    system("cp correct.OUT Main.OUT");
    return 0;
}
```

Ejemplos

- Capturando contraseñas.
- Cambiando veredictos.

Repositorios

- https://github.com/in-silico/utpjudge
- https://github.com/jhonber/Judgebot

Gracias.