

Análisis de vulnerabilidades en aplicaciones web basado en flujo de información

Alumno: Gabriel Videla

Director: Dr. Pablo E. Martínez López

Dr. Eduardo A. Bonelli

Introducción

Análisis de seguridad

Prototipo

Trabajos futuros

Conclusiones

Introducción

Vulnerabilidades en aplicaciones web

- ▶ Una **vulnerabilidad** es un error de seguridad en una aplicación web.
- ▶ Para que un atacante pueda aprovechar una vulnerabilidad debe insertar un dato no confiable y manipular la aplicación usando ese dato.

Introducción

Vulnerabilidades en aplicaciones web

```
ejecutarConsulta(  
"UPDATE usuarios  
  SET clave="" + clave + ""  
  WHERE nombre="" + nombre + ""  
);
```

Introducción

Vulnerabilidades en aplicaciones web

```
ejecutarConsulta(  
"UPDATE usuarios  
  SET clave="" + clave + ""  
  WHERE nombre="" + nombre + ""  
);
```

Si el valor de la variable nombre es:

```
' OR '0'='0
```

Introducción

Vulnerabilidades en aplicaciones web

```
ejecutarConsulta(  
"UPDATE usuarios  
  SET clave="" + clave + ""  
  WHERE nombre="" + nombre + ""  
);
```

Si el valor de la variable nombre es:

```
' OR '0'='0'
```

nos queda la condición siempre verdadera:

```
nombre='' OR '0'='0'
```

Introducción

Flujo de información

- ▶ La técnica de análisis **flujo de información** consiste en ver cómo fluye la información en un programa para garantizar una propiedad.
- ▶ En nuestro análisis consideramos tanto los flujos explícitos como los implícitos.
 - ▶ Flujo explícito: $x = y$
 - ▶ Flujo implícito:

```
if (y > 0)
  x = 1
else
  x = 2
```

Introducción

Detección de vulnerabilidades

Detectar cuando una entrada de usuario es manejada de forma insegura.

1. Indicamos métodos **fuelle** para entradas de usuario.
2. Indicamos métodos **destino** donde no puede influir una entrada de usuario.
3. Tenemos una vulnerabilidad cuando la entrada de un usuario fluye desde un fuelle a un destino.

Introducción

Análisis de seguridad

Prototipo

Trabajos futuros

Conclusiones

Análisis de seguridad

Ejecución simbólica

- ▶ El análisis de seguridad diseñado se basa en **ejecución simbólica**, un paradigma en el cual se simula la ejecución del programa usando valores simbólicos en las variables.
- ▶ El efecto de analizar un comando del lenguaje se modela a través de cambios en una representación simbólica del estado de ejecución.

Análisis de seguridad

Valores de seguridad

A cada variable se le asigna un valor σ que está formado por:

- ▶ **a** Una referencia a la información de seguridad de los campos de un objeto

```
x = new B();
```

```
y = x;
```

```
y.f = valor;
```

- ▶ **C** La clase del objeto
- ▶ **I** El valor U (confiable) o I (no confiable)

Análisis de seguridad

Elementos del análisis

- ▶ Estado de memoria \mathcal{M} formado por:
 - ▶ El ambiente de variables $\Gamma: x \mapsto \sigma$
 - ▶ La heap $H: a \mapsto [f_1 : \sigma_1, \dots, f_n : \sigma_n]$
- ▶ El nivel del contexto l

```
if (y > 0)
  x = 1
else
  x = 2
```

Análisis de seguridad

Análisis basado en supremos

```
if (e)
  x = valor1
else
  x = valor2
```

Análisis de seguridad

Reglas de análisis

- ▶ Se diseñó por cada construcción del lenguaje Java una **regla de análisis**.
- ▶ Ejemplo de regla de análisis:

$$\frac{\begin{array}{l} \mathcal{M}_1, l \vdash \quad e \quad \Rightarrow_e \quad \mathcal{M}_2, \sigma \\ \mathcal{M}_2, l \sqcup \sigma \vdash \quad c_1 \quad \Rightarrow_c \quad \mathcal{M}_3 \\ \mathcal{M}_2, l \sqcup \sigma \vdash \quad c_2 \quad \Rightarrow_c \quad \mathcal{M}_4 \end{array}}{\mathcal{M}_1, l \vdash \quad \text{if } (e) \ c_1 \ \text{else } c_2 \quad \Rightarrow_c \quad \mathcal{M}_3 \sqcup \mathcal{M}_4}$$

Introducción

Análisis de seguridad

Prototipo

Trabajos futuros

Conclusiones

Introducción

Análisis de seguridad

Prototipo

Trabajos futuros

Conclusiones

Trabajos futuros

- ▶ Mejoras al prototipo
 - ▶ Declasificación
 - ▶ Excepciones
 - ▶ Métodos recursivos
- ▶ Reflection
- ▶ Concurrencia

Introducción

Análisis de seguridad

Prototipo

Trabajos futuros

Conclusiones

Conclusiones

- ▶ El resultado de este trabajo es la aplicación de un análisis de flujo de información a Java con la finalidad de detectar vulnerabilidades en aplicaciones web.
- ▶ Se buscó un enfoque práctico para analizar código legacy Java sin la necesidad de anotaciones por parte de los usuarios.
- ▶ Se obtuvo además un prototipo que implementa el análisis.

Conclusiones

- ▶ El presente trabajo de grado se desarrolló en el marco del proyecto *Plataforma híbrida de seguridad para protección de aplicaciones web* con código NA 080/04 para la Agencia Nacional de Promoción Científica y Tecnológica, entre la empresa Core SA y el laboratorio de investigación LIFIA.
- ▶ En Core se están haciendo pruebas del prototipo para considerar su grado de escalabilidad.