Prevención de ataques ROP en ejecutables mediante instrumentación dinámica

Miguel Martín Pérez

Director: Ricardo J. Rodríguez
Codirector: Víctor Viñals Yúfera

Diciembre de 2015
Curso 15/16

Proyecto Fin de Carrera – Ingeniería en Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Contenidos

Universidad Zaragoza
1542

Contenidos

Universidad Zaragoza
1542

¿Qué es un ataque ROP?

Función vulnerable

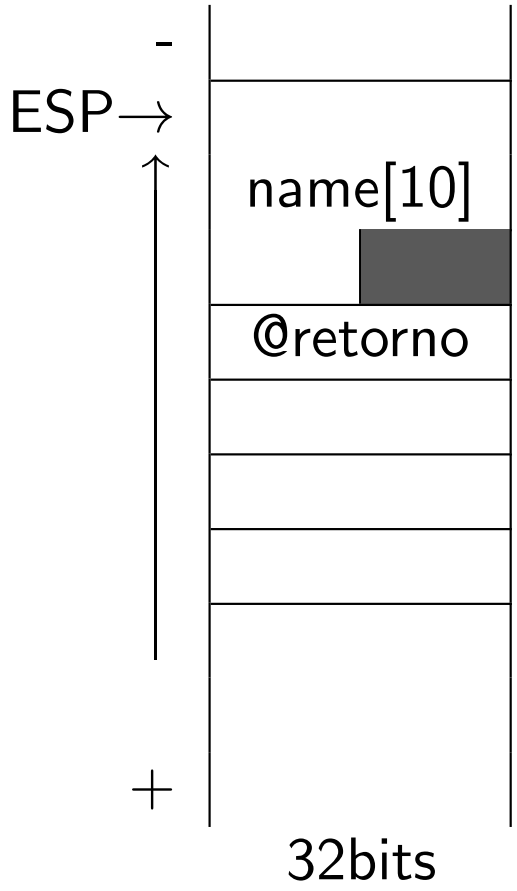
```
int hello ()
{
   char name[10];
   printf ("Enter your name:");
   scanf("\%s", name);
   printf("Hello \%s\n", name);
}
```

Universidad Zaragoza
1542

¿Qué es un ataque ROP?

Función vulnerable

```
int hello ()
{
   char name[10];
   printf ("Enter your name:");
   scanf("\%s", name);
   printf("Hello \%s\n", name);
}
```



Universidad Zaragoza
1542

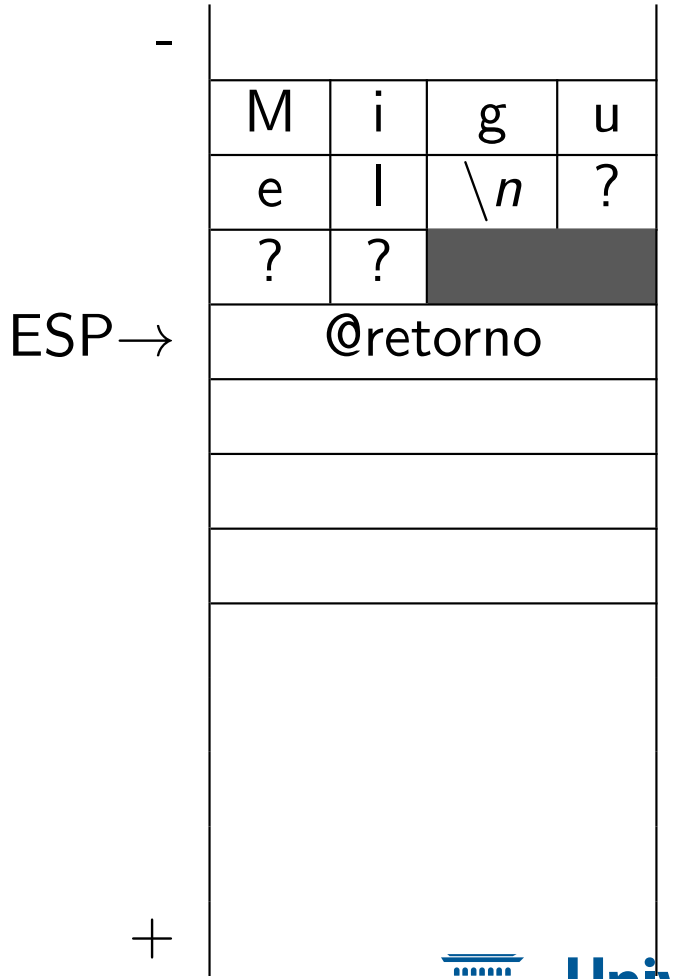
¿Qué es un ataque ROP?

Función vulnerable

```
int hello ()
{
   char name[10];
   printf ("Enter your name:");
   scanf("\%s", name);
   printf("Hello \%s\n", name);
}
```

Ejecución

```
Enter your name: Miguel
Hello Miguel
```



Universidad Zaragoza
1542

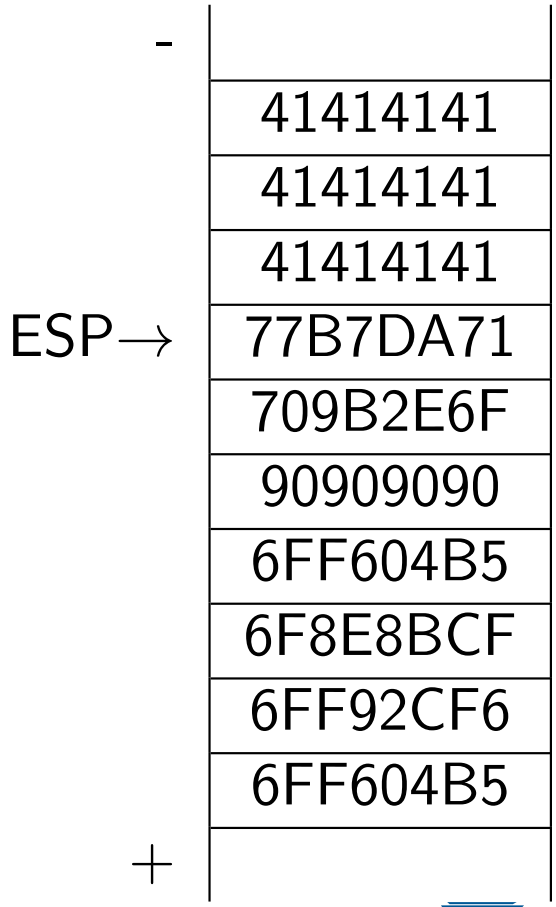
¿Qué es un ataque ROP?

Función vulnerable

```
int hello ()
{
   char name[10];
   printf ("Enter your name:");
   scanf("\%s", name);
   printf("Hello \%s\n", name);
}
```

Ejecución

Enter your name: AAAAAAAAAAAA
qÚ·wo.¿pöoÏ¡Žoö,ùoöo
Hello AAAAAAAAAAAAqÚ·wo.¿pöoÏ
¡Žoö,ùoöo



Universidad Zaragoza
1542

¿Qué es un ataque ROP?

-		
	41414141	
	41414141	
	41414141	
ESP→	77B7DA71	→ PUSH ESP; POP EBP; RETN 4; (gdi32.dll)
	709B2E6F	→ XCHG EAX, EBP; RETN; (msctf.dll)
	90909090	Compensa "RETN 4"
	6FF604B5	→ ADD EAX, C; RETN; (msvcrt.dll)
	6F8E8BCF	→ XCHG EAX, ECX; RETN; (usp10.dll)
	6FF92CF6	→MOV EAX, ECX; RETN; (msvcrt.dll)
	6FF604B5	→ ADD EAX, C; RETN; (msvcrt.dll)
+		

Universidad Zaragoza

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario

Universidad Zaragoza
1542

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario
- Arquitecturas de tamaño variable de instrucción
  - Instrucciones no alineadas
  - Permiten la ejecución de instrucciones no intencionadas
  - Facilitan la obtención de *gadgets*

Universidad Zaragoza
1542

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario
- Arquitecturas de tamaño variable de instrucción
  - Instrucciones no alineadas
  - Permiten la ejecución de instrucciones no intencionadas
  - Facilitan la obtención de *gadgets*

```
F7 C7 07 00 00 00        TEST EDI,0x7
0F 95 45 C3              SETNE BYTE PTR SS:[EBP−0x3D]
```

Universidad Zaragoza
1542

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario
- Arquitecturas de tamaño variable de instrucción
  - Instrucciones no alineadas
  - Permiten la ejecución de instrucciones no intencionadas
  - Facilitan la obtención de *gadgets*

```
C7 07 00 00 00 0F      MOV DWORD PTR DS:[EDI],0xF000000
95                     XCHG EAX,EBP
45                     INC EBP
C3                     RET
```

Universidad Zaragoza
1542

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario
- Arquitecturas de tamaño variable de instrucción
  - Instrucciones no alineadas
  - Permiten la ejecución de instrucciones no intencionadas
  - Facilitan la obtención de *gadgets*

```
C7 07 00 00 00 0F      MOV DWORD PTR DS:[EDI],0xF000000
95                     XCHG EAX,EBP
45                     INC EBP
C3                     RET
```

- Conjunto completo de Turing: potencia computacional equivalente a la máquina universal de Turing

Universidad Zaragoza 1542

Características de los ataques ROP

- Encadenamiento de secuencias de instrucciones existentes en la memoria del proceso, *gadgets*, evadiendo **W⊕X**
- Ejecución de comportamiento arbitrario
- Arquitecturas de tamaño variable de instrucción
  - Instrucciones no alineadas
  - Permiten la ejecución de instrucciones no intencionadas
  - Facilitan la obtención de *gadgets*

```
C7 07 00 00 00 0F      MOV DWORD PTR DS:[EDI],0xF000000
95                     XCHG EAX,EBP
45                     INC EBP
C3                     RET
```

- Conjunto completo de Turing: potencia computacional equivalente a la máquina universal de Turing
- Ataques JOP
  - Evolución ataques ROP: monitorización de **RETs**
  - Sustituir **RET** por "pop x; jmp x;"

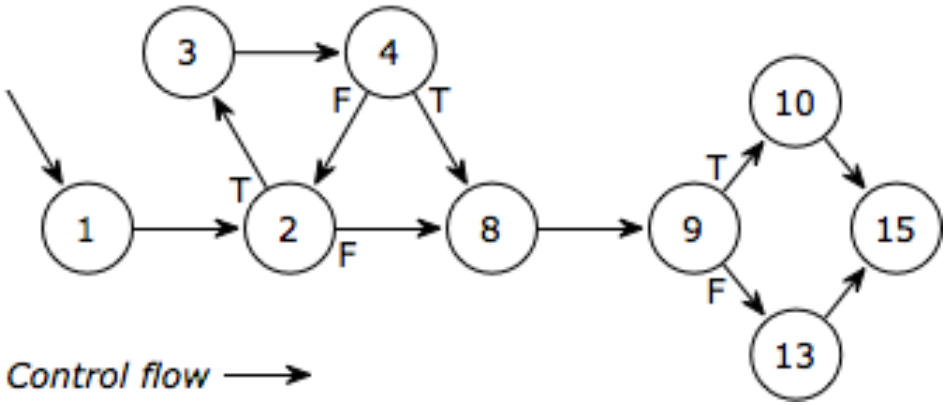
Universidad Zaragoza

Contenidos

Universidad Zaragoza
1542

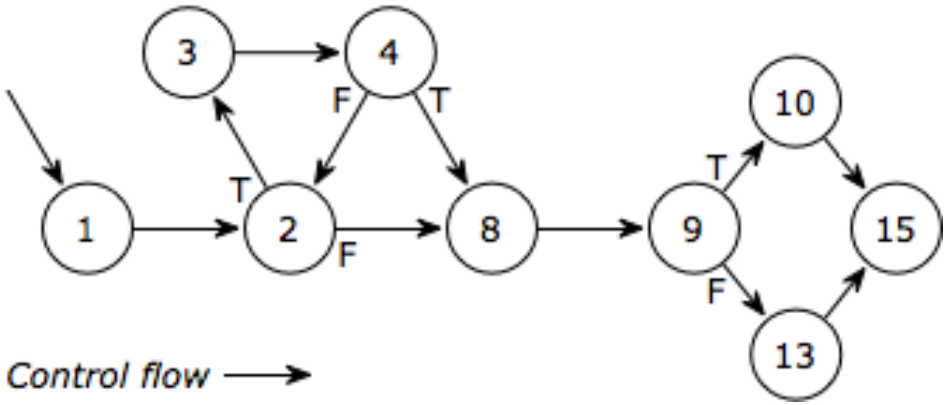
Modelo matemático

- *Control Flow Graph* (CFG): Modelo del comportamiento de un programa representado por todos los caminos posibles
  - Nodos representan Bloques Básicos (B): Secuencia de instrucciones con un único cambio de flujo en la última instrucción



Modelo matemático

- *Control Flow Graph* (CFG): Modelo del comportamiento de un programa representado por todos los caminos posibles
  - Nodos representan Bloques Básicos (B): Secuencia de instrucciones con un único cambio de flujo en la última instrucción

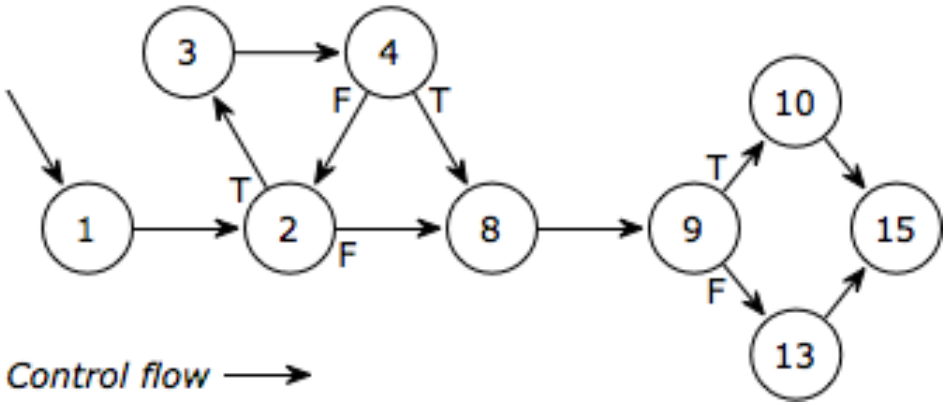


- *Control Flow Integrity* (CFI): Método que comprueba que el camino seguido por un proceso pertenece al CFG del programa

Universidad Zaragoza 1542

Modelo matemático

- *Control Flow Graph* (CFG): Modelo del comportamiento de un programa representado por todos los caminos posibles
  - Nodos representan Bloques Básicos (B): Secuencia de instrucciones con un único cambio de flujo en la última instrucción



- *Control Flow Integrity* (CFI): Método que comprueba que el camino seguido por un proceso pertenece al CFG del programa
- Autómatas de pila deterministas (M): Modelo matemático que reconoce si una palabra (proceso) pertenece a un lenguaje (CFG)

Universidad Zaragoza 1542

Modelo matemático

$M = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z, F \rangle$

- $Q = \langle Correcto, Fallo \rangle$
- $\Sigma \subseteq B \times B$
- $\Gamma = B \bigcup \{\text{Vacía}\}$

Modelo matemático

$M = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z, F \rangle$

- $Q = \langle Correcto, Fallo \rangle$
- $\Sigma \subseteq B \times B$
- $\Gamma = B \bigcup \{\text{Vacía}\}$
- $Z = \text{Vacía}$
- $q_0, F = \{Correcto\}, F \subseteq Q$
- $E \subseteq (B \times B) \in CFG$

Universidad Zaragoza
1542

Modelo matemático

$M = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z, F \rangle$

- $Q = \langle Correcto, Fallo \rangle$
- $\Sigma \subseteq B \times B$
- $\Gamma = B \bigcup \{\text{Vacía}\}$
- $Z = \text{Vacía}$
- $q_0, F = \{Correcto\}, F \subseteq Q$
- $E \subseteq (B \times B) \in CFG$

$$\Delta(q,(s,d),p) \begin{cases} (Correcto, p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Branch \\ (Correcto, SigB(s)p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Call \\ (Correcto, \varepsilon) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Ret; d = p \\ (Fallo, \varepsilon) & \text{sino} \end{cases}$$

Universidad Zaragoza 1542

Modelo matemático

$M = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z, F \rangle$

- $Q = \langle Correcto, Fallo \rangle$
- $\Sigma \subseteq B \times B$
- $\Gamma = B \bigcup \{\text{Vacía}\}$
- $Z = \text{Vacía}$
- $q_0, F = \{Correcto\}, F \subseteq Q$
- $E \subseteq (B \times B) \in CFG$

$$\Delta(q,(s,d),p) \begin{cases} (Correcto, p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Branch \\ (Correcto, SigB(s)p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Call \\ (Correcto, \varepsilon) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Ret; d = p \\ (Fallo, \varepsilon) & \text{sino} \end{cases}$$

Protección de pila: Firma de pila

La pila del autómata (Z) almacena las direcciones de retorno; almacenando su posición en pila se consigue una firma de pila con la que protegerla

Universidad Zaragoza 1542

Modelo matemático

$M = \langle Q, \Sigma, \Gamma, \Delta, q_0, Z, F \rangle$

- $Q = \langle Correcto, Fallo \rangle$
- $\Sigma \subseteq B \times B$
- $\Gamma = B \bigcup \{\text{Vacía}\}$
- $Z = \text{Vacía}$
- $q_0, F = \{Correcto\}, F \subseteq Q$
- $E \subseteq (B \times B) \in CFG$

$$\Delta(q,(s,d),p) \begin{cases} (Correcto, p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Branch \\ (Correcto, SigB(s)p) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Call \\ (Correcto, \varepsilon) & \text{si } q = Correcto; (s,d) \in E; InstFlujo(s) = Ret; d = p \\ (Fallo, \varepsilon) & \text{sino} \end{cases}$$

Protección de pila: Firma de pila

La pila del autómata (Z) almacena las direcciones de retorno; almacenando su posición en pila se consigue una firma de pila con la que protegerla

Obtención del CFG de un programa es un proceso complejo

- Falta de etiquetas y símbolos en los ejecutables
- Cambios de flujo indirectos

Simplificación del modelo y prueba de concepto

Redefinición de función de transición, $\Delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

$$\Delta(q,(s,d),p) \begin{cases} (Correcto, p) & \text{si } q = Correcto; InstFlujo(s) = Branch \\ (Correcto, SigB(s)p) & \text{si } q = Correcto; InstFlujo(s) = Call \\ (Correcto, \varepsilon) & \text{si } q = Correcto; InstFlujo(s) = Ret; d = p \\ (Fallo, \varepsilon) & \text{sino} \end{cases}$$

Universidad Zaragoza
1542

Simplificación del modelo y prueba de concepto

Redefinición de función de transición, $\Delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$

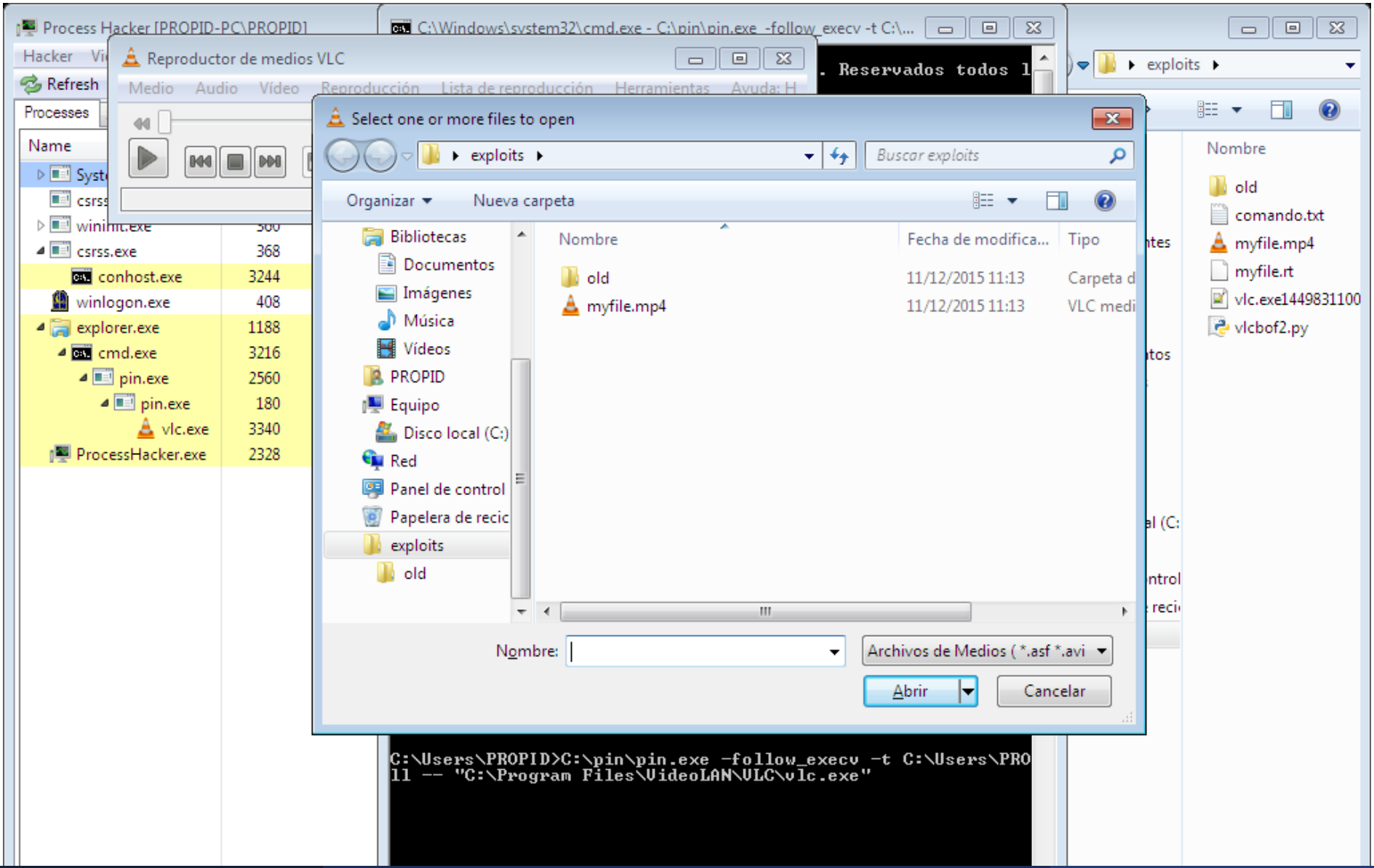
$$\Delta(q,(s,d),p) \begin{cases} (Correcto, p) & \text{si } q = Correcto; InstFlujo(s) = Branch \\ (Correcto, SigB(s)p) & \text{si } q = Correcto; InstFlujo(s) = Call \\ (Correcto, \varepsilon) & \text{si } q = Correcto; InstFlujo(s) = Ret; d = p \\ (Fallo, \varepsilon) & \text{sino} \end{cases}$$

PROPID: Prevención de ataques ROP mediante Instrumentación Dinámica

Mediante Pin, *framework* de instrumentación dinámica, se implementa:

- Para cada hilo se crea un autómata: sólo pila, *ShadowStack*
- Cada **CALL** apila dirección de retorno y posición de pila en *ShadowStack*
- Cada **RET** desapila direcciones de retorno y posicion de *ShadowStack* y comprueba que corresponden con la ejecución
  - Si no coincide se genera un informe y se cierra el programa

Demostración



Contenidos

Universidad Zaragoza 1542

Pruebas de fiabilidad y rendimiento

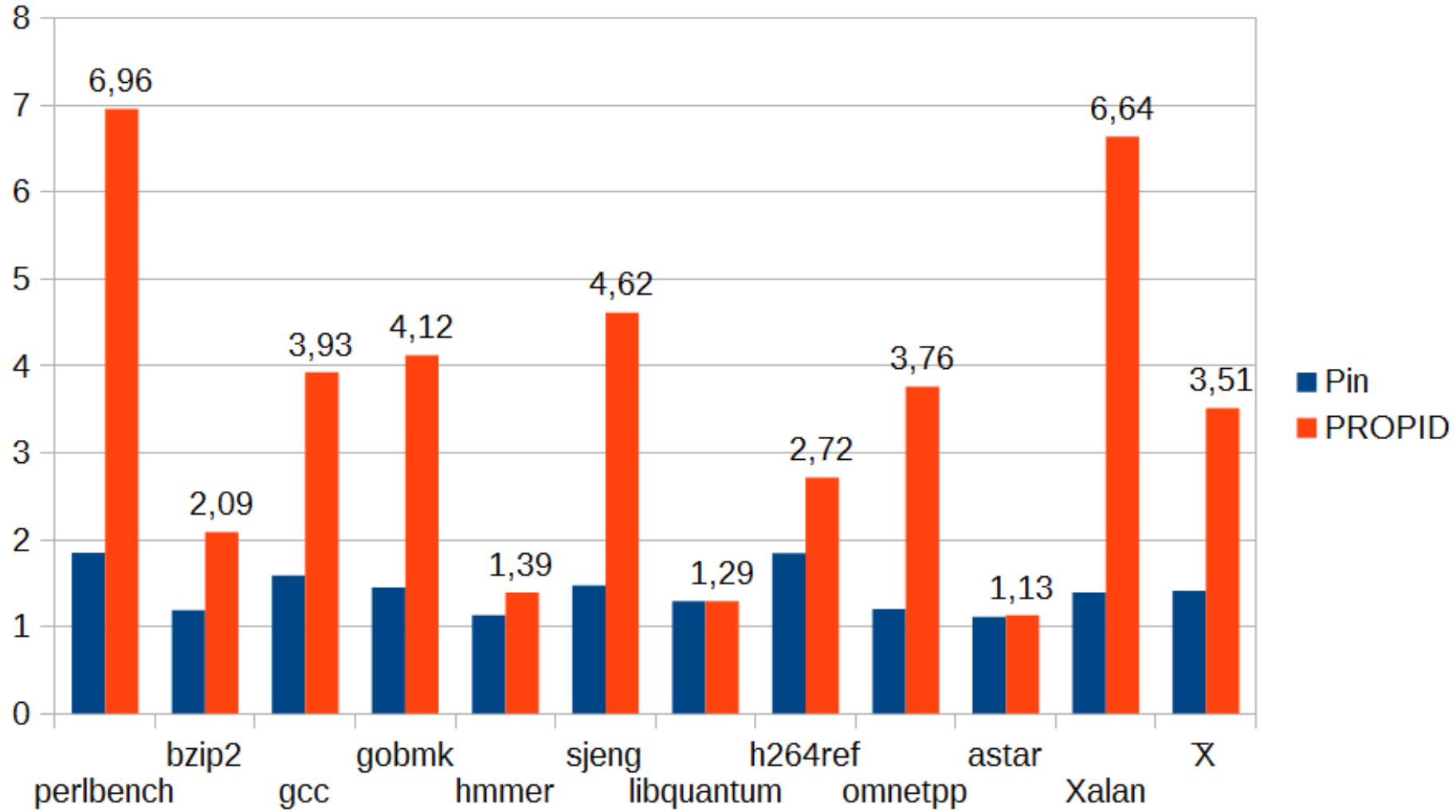
Detección de ataques

- Programa propio con vulnerabilidad intencionada
- Vulnerabilidad conocida en el reproductor VLC
- Detección del 100% ataque antes de ejecutarse el 1º *gadget*

Falsos positivos y rendimiento

- SPECint2006, AMD Athlon II X2 270, 3.4GHz 2GiB RAM, Win7 SP1
- 0% falsos positivos
- 60 ejecuciones: base, Pin, PROPID
- Sobrecarga media 3.5 veces la ejecución base
  - Confianza del 95% según distribución normal, error/media $< 1\%$

Resultados del benchmark SPEC CPUint06 comparando PROPID y Pin respecto a la ejecución base



Contenidos

Universidad Zaragoza
1542

Trabajo relacionado

Estático

- [ABEL05] etiquetan las funciones y sus retornos con identificadores únicas y añaden código que verifica en ejecución
- [EAV+06] comprueba al compilar que el código es correcto y reordena las variables para reducir la vulnerabilidad del proceso, si no se consigue en compilación se añade código que lo verifica en ejecución
  - Medidas estáticas: si se evade el CFI nada bloquea el ataque

Dinámico

- [PPK13] mediante hardware especifico de Intel registran todos los cambios de flujos ejecutados por el proceso, cuando hay una llamada al sistema se verifica que la secuencia de saltos corresponde con la secuencia teórica
  - Sólo aplicabe en arquitecturas Intel con harware *Last Branch Record*

1542

Contenidos

Universidad Zaragoza 1542

Conclusiones

Contribución

- Primera solución que unifica, mediante un autómata de pila:
  - Protección del flujo del proceso, CFI
  - Protección de pila, *ShadowStack*
- Prueba de Concepto muestra que:
  - 100% detecciones 0% falsos positivos
  - Eficiencia mejorable, 3.5x de media, picos de 7x

Trabajos futuros

- Estudio de otros DBIs para mejorar el rendimiento
- Implementación del modelo completo
- Firma de pila

Zaragoza 1542

Prevención de ataques ROP en ejecutables mediante instrumentación dinámica

Miguel Martín Pérez

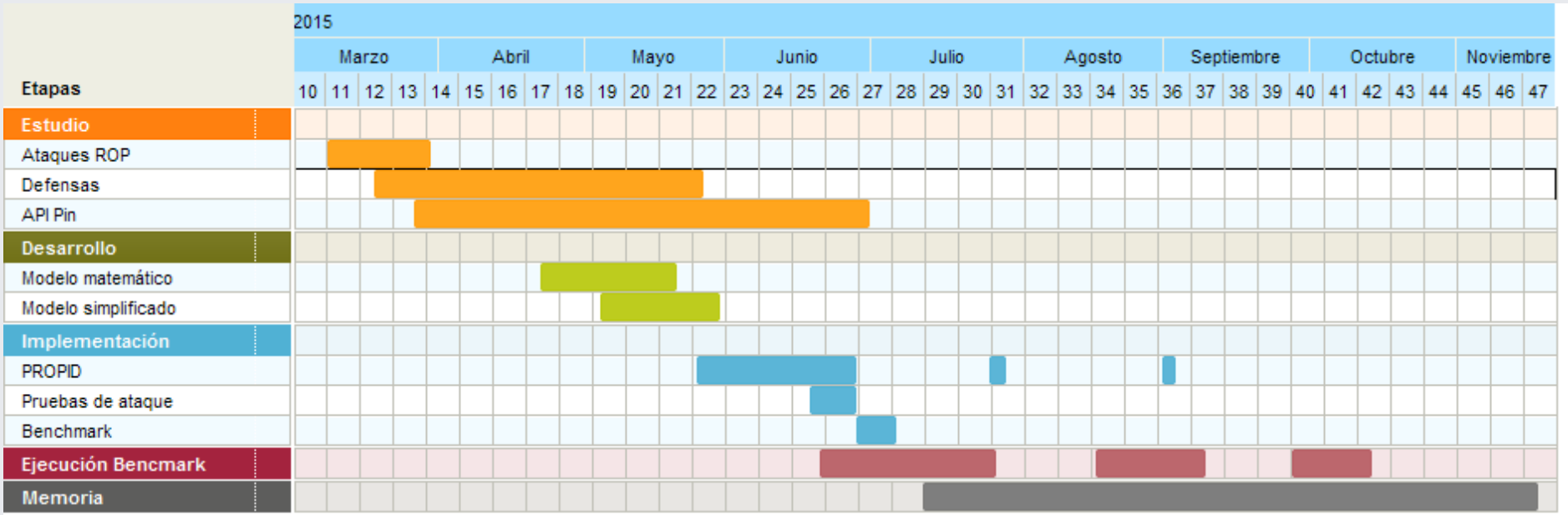
Director: Ricardo J. Rodríguez
Codirector: Víctor Viñals Yúfera

Diciembre de 2015
Curso 15/16

Proyecto Fin de Carrera – Ingeniería en Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Horas dedicadas

Tareas	Horas
Reuniones	20
Estudio Ataques ROP	50
Estudio otras defensas	290
Estudio API Pin	70
Modelo matemático	50
Implementación PROPID	30
Benchmark y pruebas	30
Memoria	180
Total	**720**



Bibliografía

- **[ABEL05]** Abadi, M.; Budiu, M.; Erlingsson, U. Ligatti, J., **Control-flow integrity**, ACM, 2005
- **[EAV+06]**Erlingsson, U.; Abadi, M.; Vrable, M.; Budiu, M. Necula, G. C., **XFI: Software guards for system address spaces**, USENIX Association, 2006
- **[PPK13]** Pappas, V.; Polychronakis, M. Keromytis, A. D., **Transparent ROP Exploit Mitigation Using Indirect Branch Tracing**, USENIX Security, 2013

Universidad Zaragoza 1542