Búsqueda de conjuntos ROP Turing-completos en sistemas Windows

Daniel Uroz Hinarejos

Director: Ricardo J. Rodríguez

Septiembre de 2016
Curso 15/16

Trabajo Fin de Grado – Ingeniería en Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Contenidos

Universidad Zaragoza
1542

Contenidos

Universidad Zaragoza
1542

Buffer overflow

Función vulnerable

```
void website() {
    char buf[10];

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```

Buffer overflow

Función vulnerable

```
void website() {
    char buf[10];

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```

- En arquitecturas de 32-bits, la pila almacena:
  - Parámetros de funciones
  - Variables locales
  - Dirección de retorno

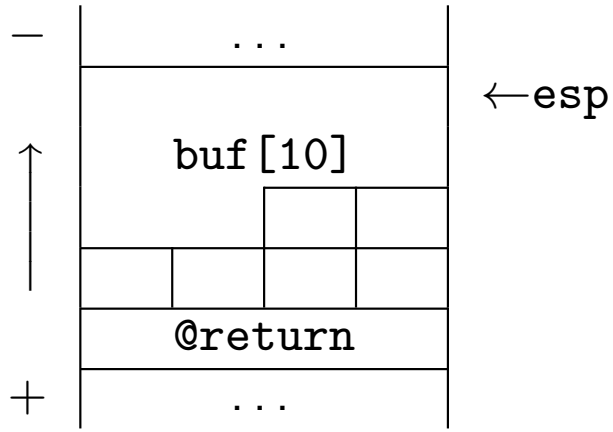
Universidad Zaragoza
1542

Buffer overflow

Función vulnerable

```
void website() {
    char buf[10]; // sub esp, 40

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```



Buffer overflow

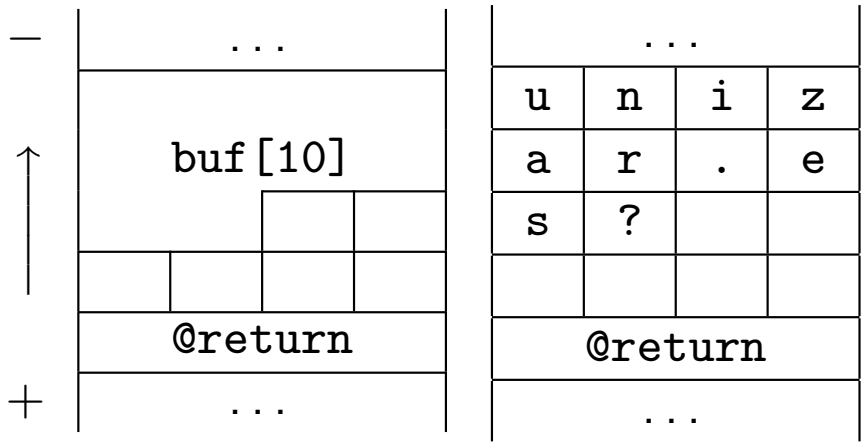
Función vulnerable

```
void website() {
    char buf[10]; // sub esp, 40

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```

Ejecución

```
Enter your website: unizar.es
Your website is unizar.es
```



Universidad Zaragoza
1542

Buffer overflow

Función vulnerable

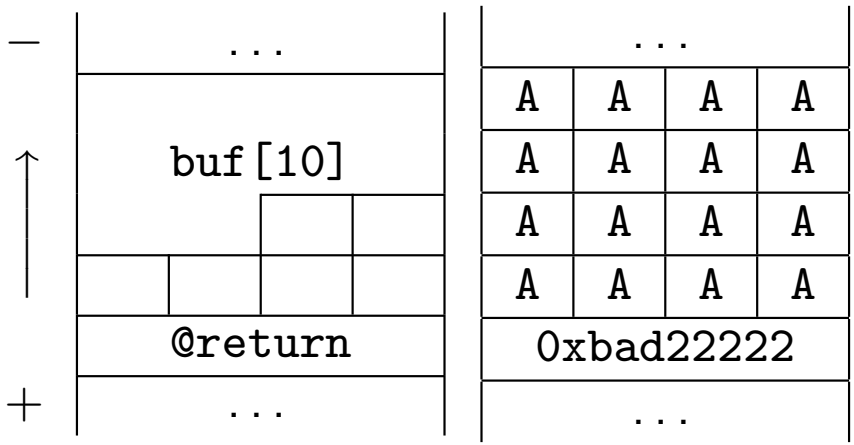
```
void website() {
    char buf[10]; // sub esp, 40

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```

Ejecución

Enter your website: AAAAAAAA AAAAAAAA°Ò”””

Your website is AAAAAAAAAAAA AAAAA°Ò”””



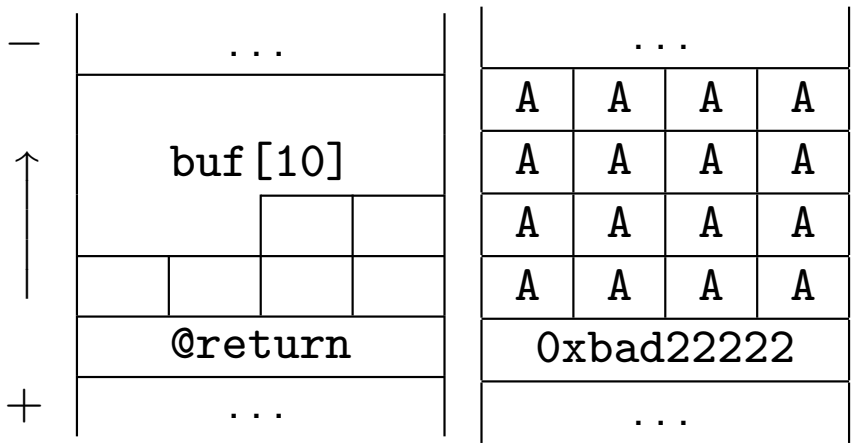
Universidad Zaragoza
1542

Buffer overflow

Función vulnerable

```
void website() {
    char buf[10]; // sub esp, 40

    printf("Enter your website: ");
    scanf("%s", buf);
    printf("Your website is %s", buf);
}
```



Ejecución

Enter your website: AAAAAAAA AAAAAAAA°Ò”””

Your website is AAAAAAAAAAA AAAAA°Ò”””

- Defensa: Write-XOR-eXecute (W⊕X)
  - Memoria marcada como escribible o ejecutable
- Evolución: ataques ROP

Universidad Zaragoza 1542

Características de ROP

- *Return Oriented-Programming* (ROP)

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)

Universidad Zaragoza
1542

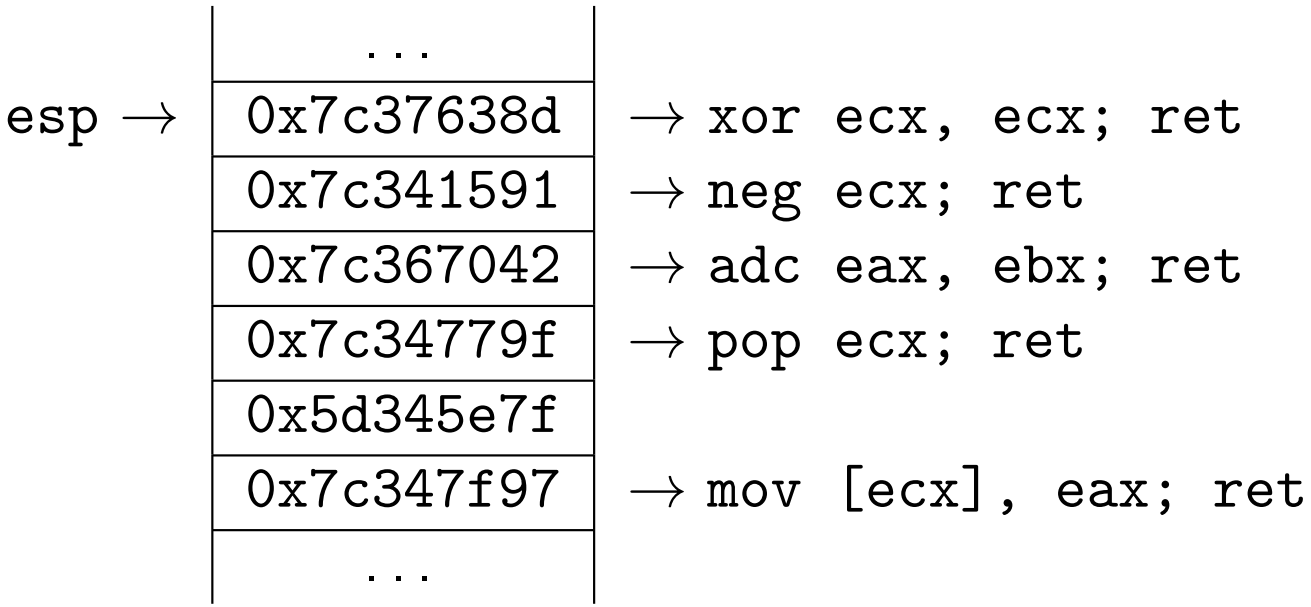
Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`

Universidad Zaragoza
1542

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`



Universidad Zaragoza
1542

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`
- Permite evadir W⊕X

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`
- Permite evadir W⊕X
- En arquitecturas de tamaño de instrucción variable (p. ej. x86, x64):
  - Instrucciones no alineadas en memoria → instrucciones no intencionadas

```
b8 89 41 08 c3          mov eax, 0xc3084189
```

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`
- Permite evadir W⊕X
- En arquitecturas de tamaño de instrucción variable (p. ej. x86, x64):
  - Instrucciones no alineadas en memoria → instrucciones no intencionadas

```
b8 89 41 08 c3          mov eax, 0xc3084189
```

```
89 41 08                mov [ecx+8], eax
c3                      ret
```

Universidad Zaragoza

Características de ROP

- *Return Oriented-Programming* (ROP)
- Control de la pila de programa (p.ej. mediante *buffer overflow*)
- *Gadget*: instrucciones acabadas en `ret`
- Permite evadir W⊕X
- En arquitecturas de tamaño de instrucción variable (p. ej. x86, x64):
  - Instrucciones no alineadas en memoria → instrucciones no intencionadas

```
b8 89 41 08 c3          mov eax, 0xc3084189
```

```
89 41 08                mov [ecx+8], eax
c3                      ret
```

- Nuevas defensas que monitorizan la instrucción `ret`
  - Evolución: *Jump-Oriented Programming* (JOP)
    - Substitución de `ret` por `pop x, jmp *x`

Universidad Zaragoza

Contenidos

Universidad Zaragoza

Sistemas Turing-completos

- Máquina universal de Turing: resuelve cualquier problema computacional
- Sistema Turing-completo: imita su funcionamiento

Sistemas Turing-completos

- Máquina universal de Turing: resuelve cualquier problema computacional
- Sistema Turing-completo: imita su funcionamiento
- Puede realizar las operaciones:
  - Cargar una constante
  - Mover un valor de posición
  - Cargar un valor desde memoria
  - Guardar un valor en memoria
  - Operaciones aritméticas
  - Operaciones lógicas
  - Saltos condicionales

Universidad Zaragoza
1542

Conjuntos ROP

- Operaciones definidas mediante conjuntos ROP

`xchg dst, src;`
`ret;`
	`push src;`
	`pop dst;`
	`ret;`
		`xor dst, dst;`
		`ret;`
		`add dst, src;`
		`ret;`
			`xor dst, dst;`
			`ret;`
			`neg src;`
			`ret;`
			`sub dst, src;`
			`ret;`

Mover un valor de posición

Universidad Zaragoza

Conjuntos ROP

- Operaciones definidas mediante conjuntos ROP

`xchg dst, src;`
`ret;`
	`push src;`
	`pop dst;`
	`ret;`
		`xor dst, dst;`
		`ret;`
		`add dst, src;`
		`ret;`
			`xor dst, dst;`
			`ret;`
			`neg src;`
			`ret;`
			`sub dst, src;`
			`ret;`

Mover un valor de posición

- Al menos un conjunto por operación $\rightarrow$ resuelve cualquier problema computacional

Universidad Zaragoza

Contenidos

Universidad Zaragoza
1542

Análisis

Requisitos funcionales

RF1 Obtener gadgets en un binario de Windows (PE)

RF2 Modificar el tamaño en bytes en la búsqueda

RF3 Filtrado de gadgets: `retf`, `jmp`, `call`

Universidad Zaragoza

Análisis

Requisitos funcionales

RF1 Obtener gadgets en un binario de Windows (PE)

RF2 Modificar el tamaño en bytes en la búsqueda

RF3 Filtrado de gadgets: `retf`, `jmp`, `call`

RF4 Buscar una operación

RF5 Especificar el registro fuente y/o destino en una operación

RF6 Búsqueda de operaciones mediante *ROP chains*

Universidad Zaragoza

Análisis

Requisitos funcionales

- RF1 Obtener gadgets en un binario de Windows (PE)
- RF2 Modificar el tamaño en bytes en la búsqueda
- RF3 Filtrado de gadgets: `retf`, `jmp`, `call`
- RF4 Buscar una operación
- RF5 Especificar el registro fuente y/o destino en una operación
- RF6 Búsqueda de operaciones mediante *ROP chains*
- RF7 Saber si un binario puede formar una máquina universal de Turing
- RF8 Saber si un S.O. puede formar una máquina universal de Turing
- RF9 Automatización de ataques ROP mediante secuencias de operaciones

Universidad Zaragoza

Análisis

Requisitos no funcionales

- RNF1 Multiplataforma: Windows, Linux y OS X
- RNF2 Facilitar soporte de distintas fuentes para la búsqueda de operaciones (XML, bases de datos, etc)
- RNF3 Facilitar la extensión a binarios de otras arquitecturas
- RNF4 Facilitar la extensión de operaciones especificadas por los usuarios

Universidad Zaragoza

Análisis

Requisitos no funcionales

RNF1 Multiplataforma: Windows, Linux y OS X

RNF2 Facilitar soporte de distintas fuentes para la búsqueda de operaciones (XML, bases de datos, etc)

RNF3 Facilitar la extensión a binarios de otras arquitecturas

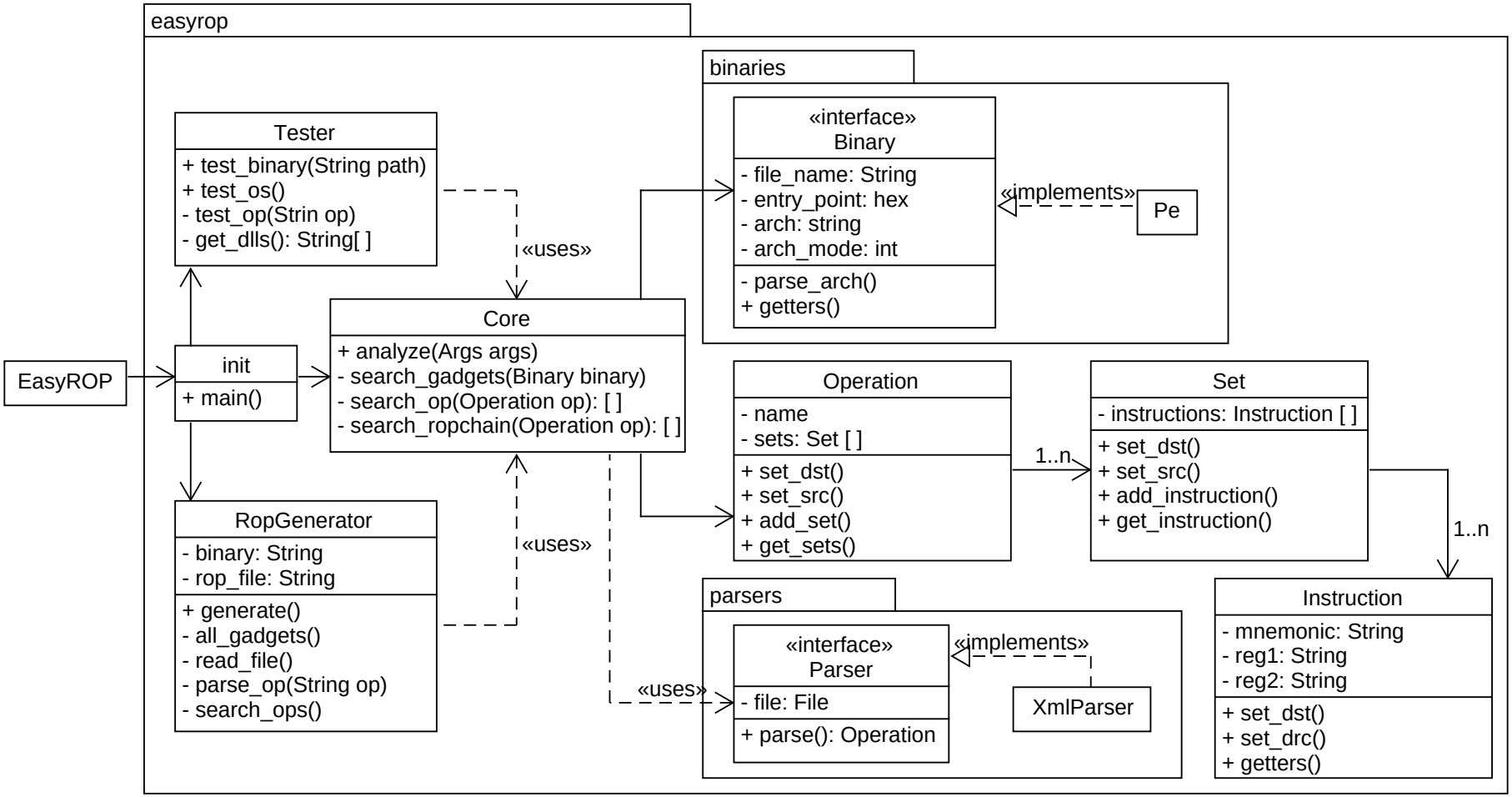
RNF4 Facilitar la extensión de operaciones especificadas por los usuarios

Herramientas

- Python
- Capstone Disassembly Framework
- XML

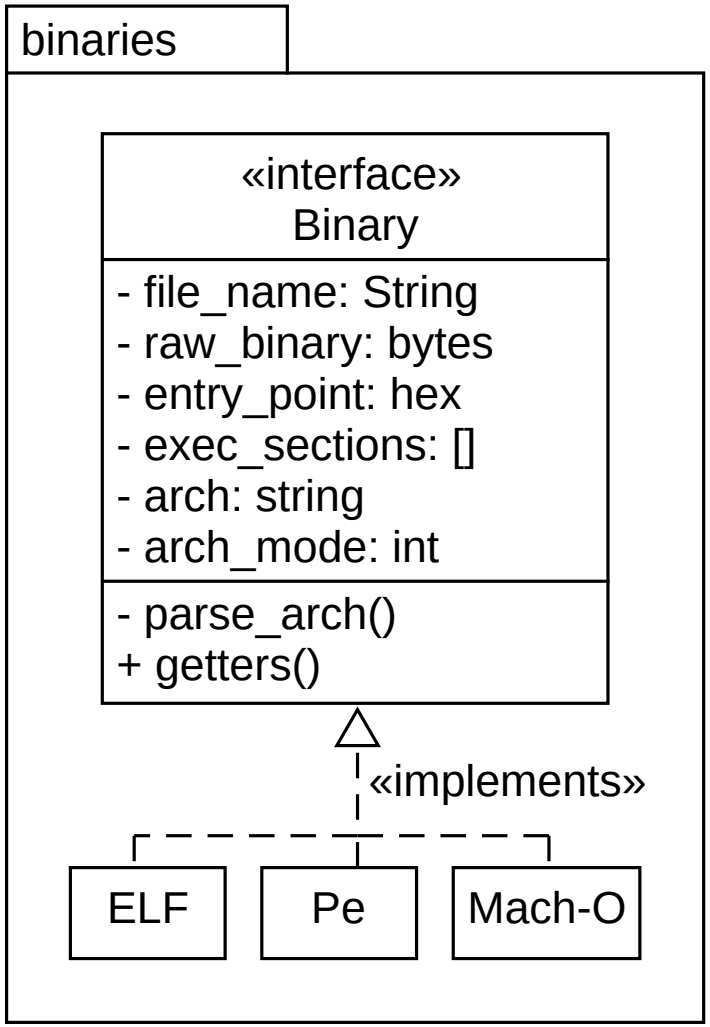
Universidad Zaragoza

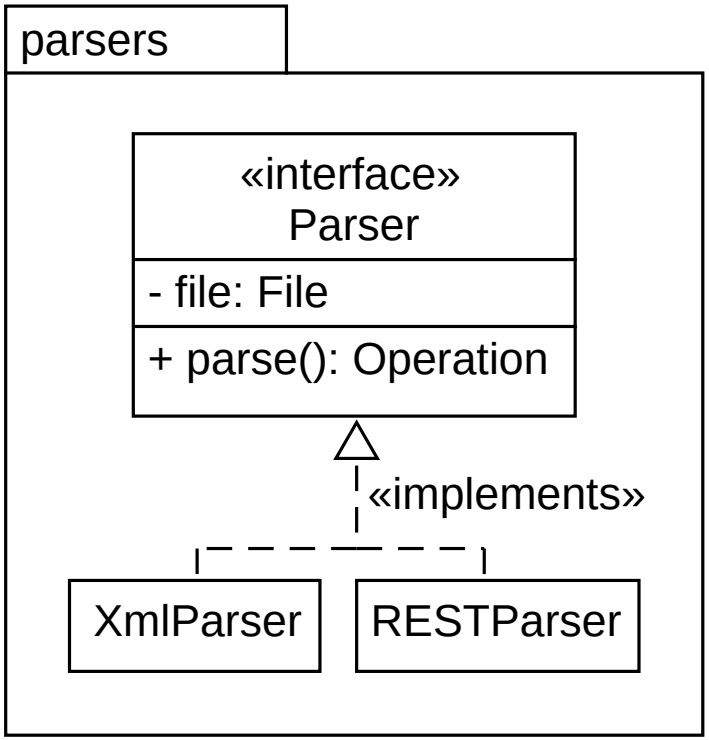
Arquitectura



Universidad Zaragoza
1542

Patrón *Facade*





¿Qué permite?

Creación de operaciones propias mediante XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE operations [
        <!ELEMENT operations (operation)+>
        <!ELEMENT operation (set)+>
        <!ATTLIST operation
                name CDATA #REQUIRED>
        <!ELEMENT set (ins)+>
        <!ELEMENT ins (reg1|reg2)*>
        <!ATTLIST ins
                mnemonic CDATA #REQUIRED>
        <!ELEMENT reg1 (#PCDATA)>
        <!ATTLIST reg1
                value CDATA #IMPLIED>
        <!ELEMENT reg2 (#PCDATA)>
        <!ATTLIST reg2
                value CDATA #IMPLIED>
        ]>
```

Universidad Zaragoza 1542

¿Qué permite?

Creación de operaciones propias mediante XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE operations [
        <!ELEMENT operations (operation)+>
        <!ELEMENT operation (set)+>
        <!ATTLIST operation
                name CDATA #REQUIRED>
        <!ELEMENT set (ins)+>
        <!ELEMENT ins (reg1|reg2)*>
        <!ATTLIST ins
                mnemonic CDATA #REQUIRED>
        <!ELEMENT reg1 (#PCDATA)>
        <!ATTLIST reg1
                value CDATA #IMPLIED>
        <!ELEMENT reg2 (#PCDATA)>
        <!ATTLIST reg2
                value CDATA #IMPLIED>
        ]>
```

```
<operations>
    <operation name="move">
        <set>
            <ins mnemonic="xor">
                <reg1>dst</reg1>
                <reg2>dst</reg2>
            </ins>
            <ins mnemonic="add">
                <reg1>dst</reg1>
                <reg2>src</reg2>
            </ins>
        </set>
    </operation>
</operations>
```

Universidad Zaragoza

¿Qué permite?

Automatizar la creación de ataques ROP

```
add(reg2, reg1)
lc(reg3)
store(reg3, reg2)
```

¿Qué permite?

Automatizar la creación de ataques ROP

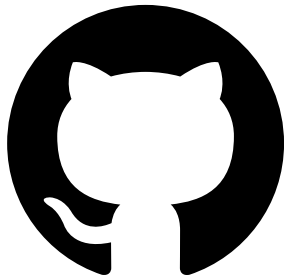
```
add(reg2, reg1)
lc(reg3)                    ⟶      xor ecx, ecx; ret
store(reg3, reg2)                  neg ecx; ret
                                   adc eax, ebx; ret
                                   pop ecx; ret
                                   mov [ecx], eax; ret
```

Universidad Zaragoza

Publicación

- EasyROP liberada bajo la licencia GNU GPLv3 en GitHub:

  `https://github.com/uZetta27/EasyROP`



Universidad Zaragoza

Contenidos

Universidad Zaragoza
1542

Experimentación

Búsqueda de todas las operaciones de Turing

Experimentación

Búsqueda de todas las operaciones de Turing

Preparación

- Subconjunto de KnownDLLs
  - Bibliotecas de enlace dinámico (DLL) más utilizadas por el S.O.

Experimentación

Búsqueda de todas las operaciones de Turing

Preparación

- Subconjunto de KnownDLLs
  - Bibliotecas de enlace dinámico (DLL) más utilizadas por el S.O.
- Entorno de pruebas
  - Intel Core i7, 8GB de RAM, 256 GB de SSD
  - Oracle VirtualBoX: 4GB de RAM, 32GB de disco

Universidad Zaragoza

Experimentación

Búsqueda de todas las operaciones de Turing

Preparación

- Subconjunto de KnownDLLs
  - Bibliotecas de enlace dinámico (DLL) más utilizadas por el S.O.
- Entorno de pruebas
  - Intel Core i7, 8GB de RAM, 256 GB de SSD
  - Oracle VirtualBoX: 4GB de RAM, 32GB de disco
- Sistemas operativos (32/64 bits)
  - Windows XP Professional
  - Windows 7 Professional
  - Windows 8.1 Pro
  - Windows 10 Education

Universidad Zaragoza

Resultados

Formación de máquinas de Turing

Versión	**32-bit**	**64-bit**
Windows XP	✗*	✗*
Windows 7	✓	✗*
Windows 8.1	✓	✓
Windows 10	✓	✓

Tiempo: 11h12m

Resultados

Formación de máquinas de Turing

Versión	**32-bit**	**64-bit**
Windows XP	✗*	✗*
Windows 7	✓	✗*
Windows 8.1	✓	✓
Windows 10	✓	✓

Tiempo: 11h12m

Resumen

- 58% del mercado de ordenadores personales (agosto 2016)
  - * todas las operaciones menos saltos condicionales → 78% del mercado
- *shell32.dll* forma todas las operaciones de Turing
  - Proporciona funciones básicas para el manejo del *shell*

Contenidos

Universidad Zaragoza
1542

Trabajo relacionado

Búsqueda de *gadgets*

- [Sha07] combinaciones de *gadgets* escogidas a mano
- [HHF09] automatiza la búsqueda de *gadgets* pero limitado a una instrucción por *gadget*
- [HSL+12] especifica *microgadgets* (tamaño de 2 ó 3 bytes)

Trabajo relacionado

Búsqueda de *gadgets*

- [Sha07] combinaciones de *gadgets* escogidas a mano
- [HHF09] automatiza la búsqueda de *gadgets* pero limitado a una instrucción por *gadget*
- [HSL+12] especifica *microgadgets* (tamaño de 2 ó 3 bytes)

Herramientas ROP

- [Sal] en Python, busca todos los *gadgets* de un binario, multiplataforma
- [Sch] similar a [Sal], añade la creación de *shellcodes* predefinidos

Universidad Zaragoza 1542

Contenidos

Universidad Zaragoza
1542

Conclusiones

Contribución

- EasyROP, primera herramienta que automatiza:
  - La búsqueda de operaciones mediante conjuntos
  - La formación de ataques ROP
- Primer estudio en Windows:
  - Máquinas de Turing en Windows 7 (32-bit), Windows 8.1 (32/64-bit) y Windows 10 (32/64-bit)
  - *shell32.dll*: permite realizar cualquier computación mediante ROP

Universidad Zaragoza
1542

Conclusiones

Contribución

- EasyROP, primera herramienta que automatiza:
  - La búsqueda de operaciones mediante conjuntos
  - La formación de ataques ROP
- Primer estudio en Windows:
  - Máquinas de Turing en Windows 7 (32-bit), Windows 8.1 (32/64-bit) y Windows 10 (32/64-bit)
  - *shell32.dll*: permite realizar cualquier computación mediante ROP

Trabajo futuro

- Supresión automática de efectos colaterales
- Extensión a otras arquitecturas
- Definición de patrones de *shellcodes*

Conclusiones

Contribución

- EasyROP, primera herramienta que automatiza:
  - La búsqueda de operaciones mediante conjuntos
  - La formación de ataques ROP
- Primer estudio en Windows:
  - Máquinas de Turing en Windows 7 (32-bit), Windows 8.1 (32/64-bit) y Windows 10 (32/64-bit)
  - *shell32.dll*: permite realizar cualquier computación mediante ROP

Trabajo futuro

- Supresión automática de efectos colaterales
- Extensión a otras arquitecturas
- Definición de patrones de *shellcodes*
- Propuesta de nuevas defensas

Búsqueda de conjuntos ROP Turing-completos en sistemas Windows

Daniel Uroz Hinarejos

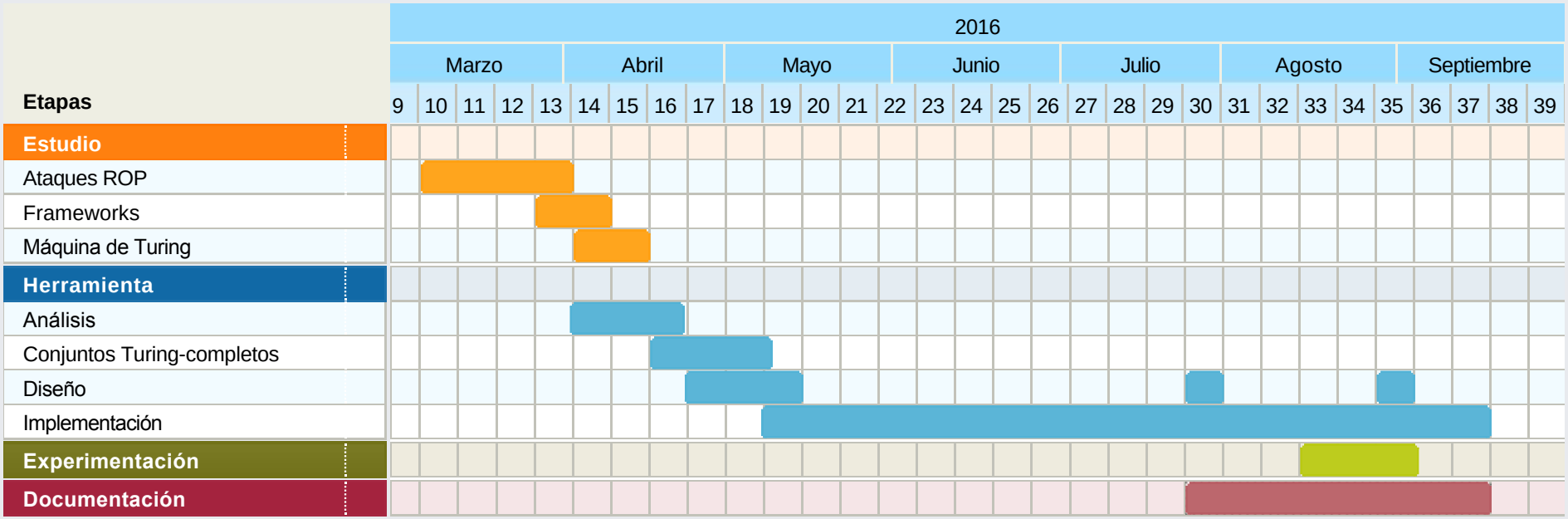
Director: Ricardo J. Rodríguez

Septiembre de 2016
Curso 15/16

Trabajo Fin de Grado – Ingeniería en Informática
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Horas de trabajo

Tareas	Horas
Estudio	40
Análisis/Diseño	33
Implementación	144
Experimentación	44
Documentación	137
Otros	22
Total	**420**



Bibliografía I

Ralf Hund, Thorsten Holz, and Felix C. Freiling, *Return-Oriented Rootkits: Bypassing Kernel Code Integrity Protection Mechanisms*, USENIX Security Symposium (Fabian Monrose, ed.), USENIX Association, 2009, pp. 383–398.

Andrei Homescu, Michael Stewart, Per Larsen, Stefan Brunthaler, and Michael Franz, *Microgadgets: Size Does Matter in Turing-Complete Return-Oriented Programming*, USENIX Workshop on Offensive Technologies (Elie Bursztein and Thomas Dullien, eds.), USENIX Association, 2012, pp. 64–76.

Jonathan Salwan, *ROPgadget Tool*, `https://github.com/JonathanSalwan/ROPgadget`, Accessed: 2016-09-03.

Universidad Zaragoza
1542

Bibliografía II


Sascha Schirra, *Ropper*, `https://github.com/sashs/Ropper`, Accessed: 2016-09-03.


Hovav Shacham, *The geometry of innocent flesh on the bone: return-into-libc without function calls (on the x86)*, ACM Conference on Computer and Communications Security (Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, eds.), ACM, 2007, pp. 552–561.

Universidad Zaragoza
1542