# Cuando ROP conoce a Turing: de BOF a LOL en 4 sencillos pasos

**Ricardo J. Rodríguez**



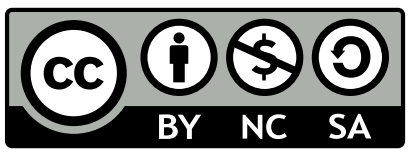
rjrodriguez@unizar.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

11 de noviembre, 2021

**Salón de Innovación y Emprendimiento**
Huesca, España



# $whoami

- **Profesor Contratado Doctor**, área *Lenguajes y Sistemas Informáticos*
- **Líneas de investigación**:
  - Análisis de seguridad y rendimiento (modelos formales)
  - Análisis de software (seguridad ofensiva y defensiva)
  - Análisis forense digital
- **Si tienes interés en ciberseguridad, ¡síguenos!** 🙂
  - `https://reversea.me`
  - `https://twitter.com/reverseame/`
  - `https://t.me/reverseame`

*Miguel Martín-Pérez*
Estudiante PhD.

*Daniel Uroz*
Estudiante PhD.

*Razvan Raducu*
Estudiante PhD.

*Pedro Fernández*
Técnico

Universidad Zaragoza

# Agenda



Universidad Zaragoza

# Motivación

```c
#include <stdio.h>

void special_trick(void)
{
    __asm__("jmp *%esp");
}

void secret(){
    puts("YOU WIN!");
}

void echo(){
    char buf[16];
    printf("What is your name?: ");
    scanf("%s", buf); // BOF vuln here

    printf("Hello, %s!", buf);
}

int main(){
    echo();
}
```

- **Flujo de ejecución** de un programa
  - Función principal: `main`
  - Ejecución secuencial (una tras otra)
- **Llamadas a sistema**: funciones externas
  - *Ejemplos*: `printf`, `puts`, `scanf`

Universidad Zaragoza

# Motivación

```c
#include <stdio.h>

void special_trick(void)
{
    __asm__("jmp *%esp");
}

void secret(){
    puts("YOU WIN!");
}

void echo(){
    char buf[16];
    printf("What is your name?: ");
    scanf("%s", buf); // BOF vuln here

    printf("Hello, %s!", buf);
}

int main(){
    echo();
}
```

- **Flujo de ejecución** de un programa
  - Función principal: `main`
  - Ejecución secuencial (una tras otra)
- **Llamadas a sistema**: funciones externas
  - *Ejemplos*: `printf`, `puts`, `scanf`

*¿Cómo sabe dónde "volver" al invocar una llamada a sistema?*

Universidad Zaragoza

# Motivación

## La pila


- También llamada *stack* (del inglés)
- **Memoria para almacenar datos abstractos** (cualquier tipo) en orden LIFO
- Guarda temporalmente valores que pueden ser necesitados más tarde

# Motivación

## La pila


- También llamada *stack* (del inglés)
- **Memoria para almacenar datos abstractos** (cualquier tipo) en orden LIFO
- Guarda temporalmente valores que pueden ser necesitados más tarde
- **Variables locales** (en el ejemplo, `buf`)
- **Dirección de retorno**: donde hay que "volver" al invocar una llamada
  - Por ejemplo: al llamar a `scanf`, se guarda en la pila que la ejecución tiene que volver a la línea 12 del código

# Motivación

## Vulnerabilidades de desbordamiento en pila

### Problemas del código anterior

- `scanf` es una función insegura: **no** controla el tamaño límite del destino
- Imagina que tienes un vaso de medio litro, y lo llenas con una botella de litro y medio (con embudo). *¿Qué pasa?*

Universidad Zaragoza

# Motivación

## Vulnerabilidades de desbordamiento en pila

### Problemas del código anterior

- `scanf` es una función insegura: **no** controla el tamaño límite del destino
- Imagina que tienes un vaso de medio litro, y lo llenas con una botella de litro y medio (con embudo). *¿Qué pasa?* En este ejemplo:
  - El embudo es `scanf`
  - La variable `buf` es el vaso
  - La botella de litro y medio la entrada del usuario

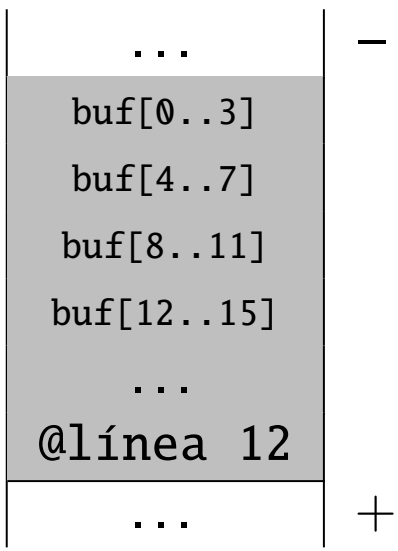
Universidad Zaragoza

# Motivación

## Vulnerabilidades de desbordamiento en pila

### Problemas del código anterior

- `scanf` es una función insegura: **no** controla el tamaño límite del destino
- Imagina que tienes un vaso de medio litro, y lo llenas con una botella de litro y medio (con embudo). *¿Qué pasa?* En este ejemplo:
  - El embudo es `scanf`
  - La variable `buf` es el vaso
  - La botella de litro y medio la entrada del usuario




- Si el usuario da una entrada lo suficientemente larga, **¡llegará a sobrescribir la dirección de retorno!**

# Motivación


- *Demo 1*: redirección de ejecución (a función ya existente)
- *Demo 2*: ejecución de código arbitrario (inserción de código nuevo)

Universidad Zaragoza

# Motivación


Universidad Zaragoza

# Motivación

## Defensas software

- **Canarios de pila**
  - Inserta un valor en la pila que se comprueba al final de la función
  - Si no coincide, se para la ejecución del programa con error
  - Evita el problema visto en la primera y segunda demostración

Universidad Zaragoza

# Motivación

## Defensas software

- **Canarios de pila**
  - Inserta un valor en la pila que se comprueba al final de la función
  - Si no coincide, se para la ejecución del programa con error
  - Evita el problema visto en la primera y segunda demostración
- **Zona de memoria de pila no ejecutable**
  - Páginas de memoria con permisos W⊕X
  - Error si se intenta ejecutar código en una página sin permisos
  - Evita el problema visto en la segunda demostración

Universidad Zaragoza
1542

# Motivación

## Defensas software

- **Canarios de pila**
  - Inserta un valor en la pila que se comprueba al final de la función
  - Si no coincide, se para la ejecución del programa con error
  - Evita el problema visto en la primera y segunda demostración
- **Zona de memoria de pila no ejecutable**
  - Páginas de memoria con permisos W⊕X
  - Error si se intenta ejecutar código en una página sin permisos
  - Evita el problema visto en la segunda demostración

***¿Se puede ejecutar código arbitrario sin insertar código?***

Universidad Zaragoza

# Agenda



Universidad Zaragoza

# Ataques ROP

- **Ataques `ret2libc`, `ret2text`, `ret2code`, y variantes**
  - Ataques de reutilización de código: redirección de la ejecución a código existente
  - *Similar a lo hecho en la primera demostración, pero aprovechándose del código existente en las librerías de sistema o la propia aplicación*

# Ataques ROP

- **Ataques `ret2libc`, `ret2text`, `ret2code`, y variantes**
  - Ataques de reutilización de código: redirección de la ejecución a código existente
  - *Similar a lo hecho en la primera demostración, pero aprovechándose del código existente en las librerías de sistema o la propia aplicación*
- **Evolución**: ataques ROP (*Return-Oriented-Programming*)
  - Presentado en 2007 para arquitecturas Intel x86
  - Existen en la mayoría de arquitecturas (x64, RISC, SPARC, RISC-V)
  - **Definiciones**:
    - **Gadget ROP**: trozos de código ya presentes en la memoria del proceso, acaban en una instrucción que cambia el flujo de ejecución
    - Si el atacante controla la escritura en la pila, puede controlar cómo se ejecutan estas secuencias de gadgets: **cadena ROP** (o *ROP chain*)

Universidad Zaragoza 1542

# Ataques ROP

## ¿Y por qué aparecen?

- **Longitud de instrucciones variable**
  - Cambia la interpretación, ¡según dónde se comience a leer!

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


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

# Ataques ROP

## ¿Y por qué aparecen?

- **Longitud de instrucciones variable**
  - Cambia la interpretación, ¡según dónde se comience a leer!

```
b8 89 41 08 c3      mov eax, 0xc3084189
89 41 08            mov [ecx+8], eax
c3                  ret
```

| | | |
|---|---|---|
| | ... | |
| esp → | 0x7c37638d | → pop ecx; ret |
| | 0xBADCOFFE | |
| | 0x7c341591 | → pop edx; ret |
| | 0xBAADF00D | |
| | 0x7c367042 | → xor eax, eax; ret |
| | 0x7c34779f | → add eax, ecx; ret |
| | 0x7c347f97 | → mov ebx, eax; ret |
| | ... | |

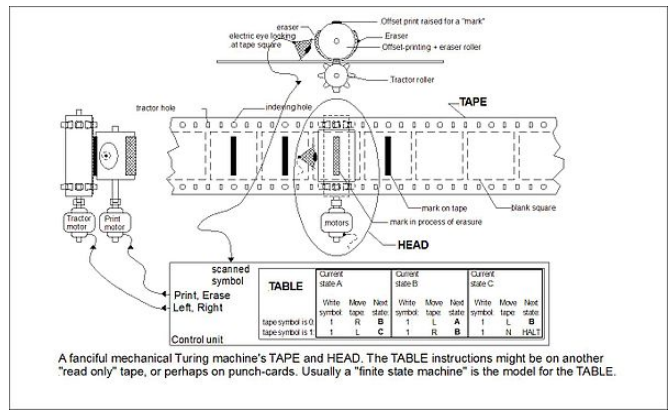
  - Esta secuencia de ejemplo de la derecha está haciendo lo siguiente:
    1. Coloca el valor `0xBADCOFFE` en el registro `ecx`
    2. Coloca el valor `0xBAADF00D` en el registro `edx`
    3. Deja el registro `eax` al valor 0
    4. Suma `eax` con `ecx`, dejando el resultado en `eax`
    5. Copia el valor de `eax` al registro `ebx`

Universidad Zaragoza

# Ataques ROP


**ROP es Turing-completo**

- Se puede realizar cualquier cálculo arbitrario
- En otras palabras, permite simular una máquina de Turing



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Universidad Zaragoza 1542

# Ataques ROP

## Tesis de Church-Turing

- *Cualquier computación se puede traducir a una máquina de Turing*
  - O sea, se puede construir una máquina de Turing equivalente a una cadena ROP

Universidad Zaragoza

# Ataques ROP

## Tesis de Church-Turing

- *Cualquier computación se puede traducir a una máquina de Turing*
  - O sea, se puede construir una máquina de Turing equivalente a una cadena ROP

**Pregunta de interés**

***¿Cuánto poder tiene, en realidad, un atacante con ROP?***

Universidad Zaragoza

# Ataques ROP

## Tesis de Church-Turing

- *Cualquier computación se puede traducir a una máquina de Turing*
  - O sea, se puede construir una máquina de Turing equivalente a una cadena ROP

## Pregunta de interés

***¿Cuánto poder tiene, en realidad, un atacante con ROP?***

1. **En programas reales, ¿cuál es la prevalencia de gadgets ROP para cualquier operación?**
2. **¿Es posible, en programas reales, hacer cualquier cosa con ROP?**
   **¿Puede un atacante realmente construir cualquier algoritmo con ROP?**

Universidad Zaragoza
1542

# Agenda



Universidad Zaragoza
1542

# El lenguaje virtual ROPLang

## Operaciones virtuales

- Simuladas mediante gadgets ROPs
- Notación similar al ensamblador de Intel (usamos sintaxis Intel)

Universidad Zaragoza

# El lenguaje virtual ROPLANG

## Operaciones virtuales

- Simuladas mediante gadgets ROPs
- Notación similar al ensamblador de Intel (usamos sintaxis Intel)

### *Categorías de operaciones*

- **Aritméticas**: suma (`add`), resta (`sub`), and negación (`neg`)
- **Asignación**: para dar valores a variables (serán registros de la CPU)
- **Dereferencia**: para lectura/escritura de una dirección de memoria (`ld`, `st`)
- **Lógicas**: `xor`, `and`, `or`, y `not`
  - Por las Leyes de Morgan, se pueden simplificar al conjunto de {`and`, `or`} y {`xor`, `not`, `neg`}
- **Saltos**: condicionales y no condicionales
  - Los saltos condicionales requieren algunos trucos para que funcionen

Universidad Zaragoza
1542

# El lenguaje virtual ROPLANG

## Operaciones aritméticas

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `add(dst, src)` | `add dst, src` |
| | `clc`<br>`adc dst, src` |
| | `inc dst` |
| `sub(dst, src)` | `sub dst, src` |
| | `clc`<br>`sbb dst, src` |
| | `dec dst` |
| `neg(dst)` | `xor REG1, REG1`<br>`sub REG1, dst`<br>`mov(dst, REG1)` |
| | `neg dst` |

*La instrucción `ret` (al final de cada gadget ROP) se ha omitido de forma deliberada*

Universidad Zaragoza

# El lenguaje virtual ROPLang

## Operaciones aritméticas

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `add(dst, src)` | `add dst, src` |
| | `clc`<br>`adc dst, src` |
| | `inc dst` |
| `sub(dst, src)` | `sub dst, src` |
| | `clc`<br>`sbb dst, src` |
| | `dec dst` |
| `neg(dst)` | `xor REG1, REG1`<br>`sub REG1, dst`<br>`mov(dst, REG1)` |
| | `neg dst` |

## Operaciones de asignación

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `mov(dst, src)` | `mov dst, src` |
| | `xchg dst, src` |
| | `xor dst, dst`<br>`add dst, src` |
| | `xor dst, dst`<br>`not dst`<br>`and dst, src` |
| | `clc`<br>`cmovnc dst, src` |
| | `stc`<br>`cmovc dst, src` |
| | `push src`<br>`pop dst` |
| `lc(dst, value)` | `pop dst`; *el valor se coge de la pila* |
| | `popad`; *el valor se coge de la pila* |

*La instrucción `ret` (al final de cada gadget ROP) se ha omitido de forma deliberada*

Universidad Zaragoza 1542

# El lenguaje virtual ROPLang

## Operaciones aritméticas

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `add(dst, src)` | `add dst, src` |
| | `clc`<br>`adc dst, src` |
| | `inc dst` |
| `sub(dst, src)` | `sub dst, src` |
| | `clc`<br>`sbb dst, src` |
| | `dec dst` |
| `neg(dst)` | `xor REG1, REG1`<br>`sub REG1, dst`<br>`mov(dst, REG1)` |
| | `neg dst` |

## Operaciones de asignación

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `mov(dst, src)` | `mov dst, src` |
| | `xchg dst, src` |
| | `xor dst, dst`<br>`add dst, src` |
| | `xor dst, dst`<br>`not dst`<br>`and dst, src` |
| | `clc`<br>`cmovnc dst, src` |
| | `stc`<br>`cmovc dst, src` |
| | `push src`<br>`pop dst` |
| `lc(dst, value)` | `pop dst`; *el valor se coge de la pila* |
| | `popad`; *el valor se coge de la pila* |

## Operaciones de dereferencia

| Operación | Gadgets ROP |
|---|---|
| `ld(dst, src)` | `mov dst, [src]` |
| `st(dst, src)` | `mov [dst], src` |

*La instrucción `ret` (al final de cada gadget ROP) se ha omitido de forma deliberada*

Universidad Zaragoza

# El lenguaje virtual ROPLang

## Operaciones aritméticas

| Operación | Gadgets ROP/Operaciones |
|---|---|
| add(dst, src) | add dst, src |
| | clc<br>adc dst, src |
| | inc dst |
| sub(dst, src) | sub dst, src |
| | clc<br>sbb dst, src |
| | dec dst |
| neg(dst) | xor REG1, REG1<br>sub REG1, dst<br>mov(dst, REG1) |
| | neg dst |

## Operaciones lógicas

| Operación | Gadgets ROP/Operaciones |
|---|---|
| xor(dst, src) | xor dst, src |
| and(dst, src) | and dst, src |
| or(dst, src) | or dst, src |
| not(dst) | not dst |
| | xor dst, 0xFFFFFFFF |

## Operaciones de asignación

| Operación | Gadgets ROP/Operaciones |
|---|---|
| mov(dst, src) | mov dst, src |
| | xchg dst, src |
| | xor dst, dst<br>add dst, src |
| | xor dst, dst<br>not dst<br>and dst, src |
| | clc<br>cmovnc dst, src |
| | stc<br>cmovc dst, src |
| | push src<br>pop dst |
| lc(dst, *value*) | pop dst; *el valor se coge de la pila* |
| | popad; *el valor se coge de la pila* |

## Operaciones de dereferencia

| Operación | Gadgets ROP |
|---|---|
| ld(dst, src) | mov dst, [src] |
| st(dst, src) | mov [dst], src |

*La instrucción `ret` (al final de cada gadget ROP) se ha omitido de forma deliberada*

Universidad Zaragoza 1542

# El lenguaje virtual ROPLANG

## Operaciones de comparación

| Operación | Operaciones |
|---|---|
| `eqc(dst, src)` | `sub(dst, src)`<br>`neg(dst)` |
| `ltc(dst, src)` | `sub(dst, src)` |

## Saltos condicionales

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `gcf(dst`$_{CF}$`, cop(dst, src))` | `lc(REG1, 0)`<br>*Operación de comparación* cop(dst, src)<br>`adc dst`$_{CF}$`, REG1` |
| | `lc(REG1, 0)`<br>*Operación de comparación* cop(dst, src)<br>`sbb dst`$_{CF}$`, REG1`<br>`neg(dst`$_{CF}$`)` |
| | `lc(dst`$_{CF}$`, 0)`<br>*Operación de comparación* cop(dst, src)<br>`rcl dst`$_{CF}$`, 1` |
| `lsd(dst`$_{CF}$`,` $\delta$`)` | `lc(REG1,` $\delta$`)`<br>`neg(dst`$_{CF}$`)`<br>`and(dst`$_{CF}$`, REG1)` |
| `spa(src)` | `add(REG_SP, src)` |
| `sps(src)` | `sub(REG_SP, src)` |

## Saltos no condicionales

| Operación | Gadgets ROP/Operaciones |
|---|---|
| `jmp(dst,` $\delta$`)` | `lc(dst,` $\delta$`)`<br>`spa(dst)` |

*La instrucción `ret` (al final de cada gadget ROP) se ha omitido de forma deliberada*

Universidad Zaragoza 1542

# El lenguaje virtual ROPLANG

*Algunas notas...*

- **Lista de gadgets ROP no exhaustivas**
- **Algunas operaciones son operaciones virtuales, mientras que otras son ROP gadgets**
  - Nuestro lenguaje admite la mezcla de tipos de operaciones
- **Suposición**: *no ocurren efectos secundarios entre secuencias de operaciones virtuales*

## ROPLANG es Turing-completo

- **Simulación una máquina de Turing clásica con ROPLANG en:**
  - Uroz, D. & Rodríguez, R. J. *Evaluation of the Executional Power in Windows using Return Oriented Programming*. Proceedings of the 15th IEEE Workshop on Offensive Technologies (WOOT), IEEE, 2021, 361-372. DOI: 10.1109/SPW53761.2021.00056

Universidad Zaragoza 1542

# El lenguaje virtual ROPLANG

## La herramienta ROP3

ROP3

- **Desarrollada en Python, usa Capstone** para desensamblar los ficheros
- **Soporta todas las operaciones virtuales de** ROPLANG
- **Definición de operaciones usando sintaxis YAML**
  - Permite definir operaciones personalizadas (como un único o múltiples ficheros YAML)
  - Registros lógicos CPU específicos o máscaras
  - Valores arbitrarios
- **La búsqueda de gadgets ROP es similar al algoritmo Galileo**
  - Algoritmo original de Shacham (definición original de ROP)

Universidad Zaragoza 1542

# El lenguaje virtual ROPLang

## La herramienta ROP3 – ejemplos de ficheros YAML

```yaml
# Add values
add:
  # add dst, src
  -
    - mnemonic: add
      op1: dst
      op2: src

  # clc
  # adc dst, src
  -
    - mnemonic: clc
    - mnemonic: adc
      op1: dst
      op2: src
```

```yaml
# NOT value
not:
  # not dst
  -
    - mnemonic: not
      op1: dst

  # xor dst, src (src = 0xFFFFFFFF)
  -
    - mnemonic: xor
      op1: dst
      op2:
        reg: src
        value: 0xFFFFFFFF
```

Universidad Zaragoza

# El lenguaje virtual ROPLANG

## La herramienta ROP3 – construcción de cadenas ROP

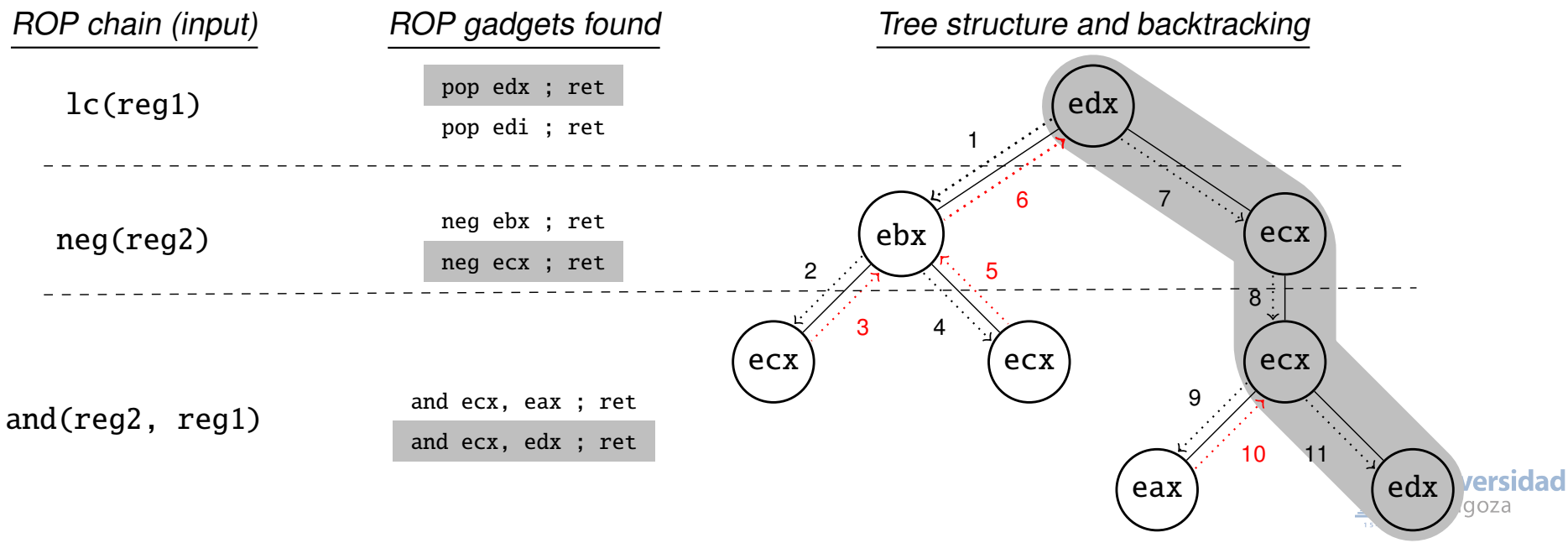
- **Se especifica mediante operaciones de** ROPLANG
- **Algoritmo de búsqueda**:
  1. Busca todos aquellos gadgets que cumplan cada operación ROPLANG
  2. Construye una estructura de árbol, según el orden de las operaciones definido en la cadena
  3. Se resuelven las dependencias entre operaciones **recorriendo el árbol de manera recursiva (en profundidad con *backtracking*)**
- **Efectos secundarios:** inhabilita la rama del árbol en conflicto

Universidad Zaragoza

# El lenguaje virtual ROPLANG

## La herramienta ROP3 – construcción de cadenas ROP

- **Se especifica mediante operaciones de** ROPLANG
- **Algoritmo de búsqueda**:
  1. Busca todos aquellos gadgets que cumplan cada operación ROPLANG
  2. Construye una estructura de árbol, según el orden de las operaciones definido en la cadena
  3. Se resuelven las dependencias entre operaciones **recorriendo el árbol de manera recursiva (en profundidad con *backtracking*)**
- **Efectos secundarios:** inhabilita la rama del árbol en conflicto



# El lenguaje virtual ROPLang

## La herramienta ROP3

- **Liberada bajo licencia GNU/GPLv3**
- Acepta **múltiples parámetros**:
  - Tamaño máximo de los gadgets ROP (en bytes)
  - Instrucciones finales de los gadgets (`ret`, `jmp`, `retf`)
  - ...
- Es también una librería de Python3

`https://github.com/reverseame/rop3`

Universidad Zaragoza

# Agenda



Universidad Zaragoza

# Evaluación

## Conjunto de pruebas

- **Parte de las librerías de sistema de `KnownDlls`**
  - Todas las DLLs comunes en las versiones de Windows consideradas en experimentación
- Windows sobre Oracle VirtualBox, versiones de 32 y 64 bits
  - Windows 10 Education 10.0.14393 Build 14393 (32-bit) y Windows 10 Pro 1703 Build 15063.726 (64-bit)
  - Windows 7 Professional 6.1.7601 Service Pack 1 Build 7601

*Sobre las gráficas...*

- **Gráfico de calor de ocurrencia (en %) para cada operación en cada DLL**
- **Los números de resultados están anotados**
  - El dígito más significativo y el orden de magnitud cuando el número de resultados es $\geq 10^4$
- **DLLs ordenadas por tamaño** (menos a más)

Universidad Zaragoza 1542

# Evaluación

## Configuración de ROP3

- Gadgets ROP de 10 bytes de tamaño (máximo)
- Sólo la instrucción `ret` como instrucción final

Universidad Zaragoza

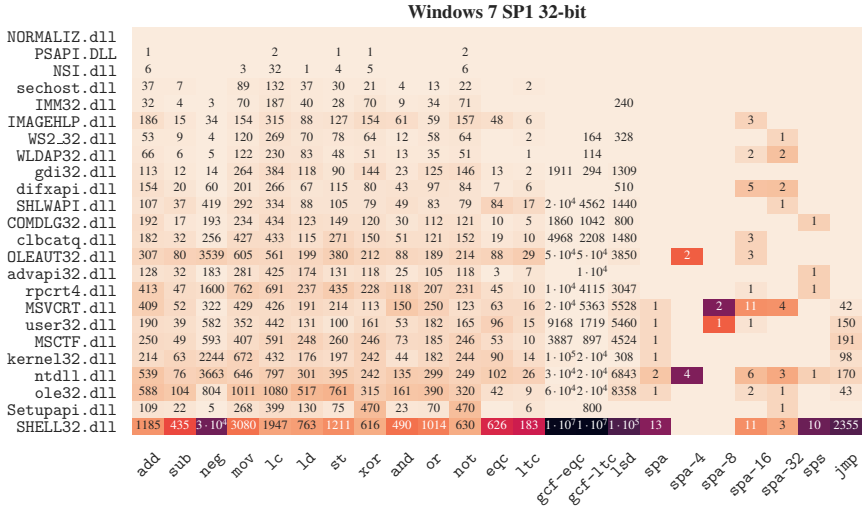
# Evaluación

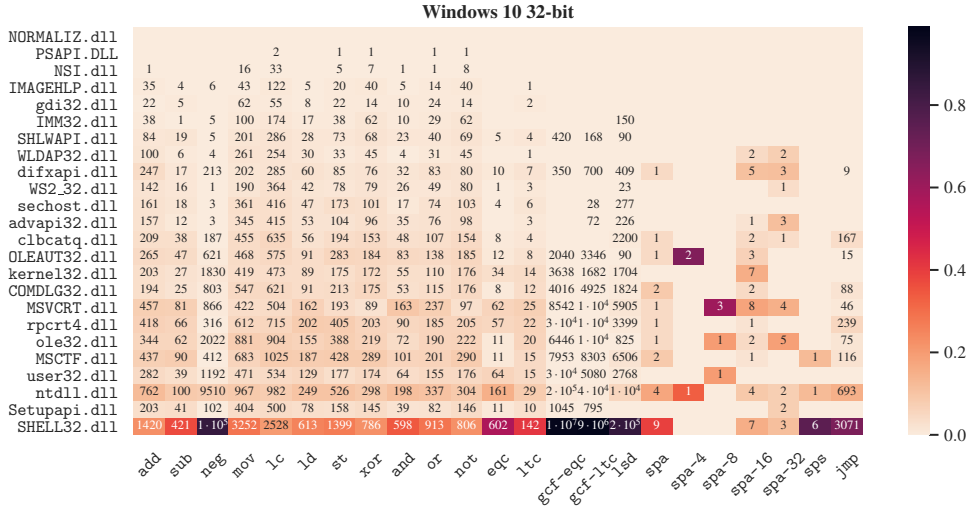
## Configuración de ROP3

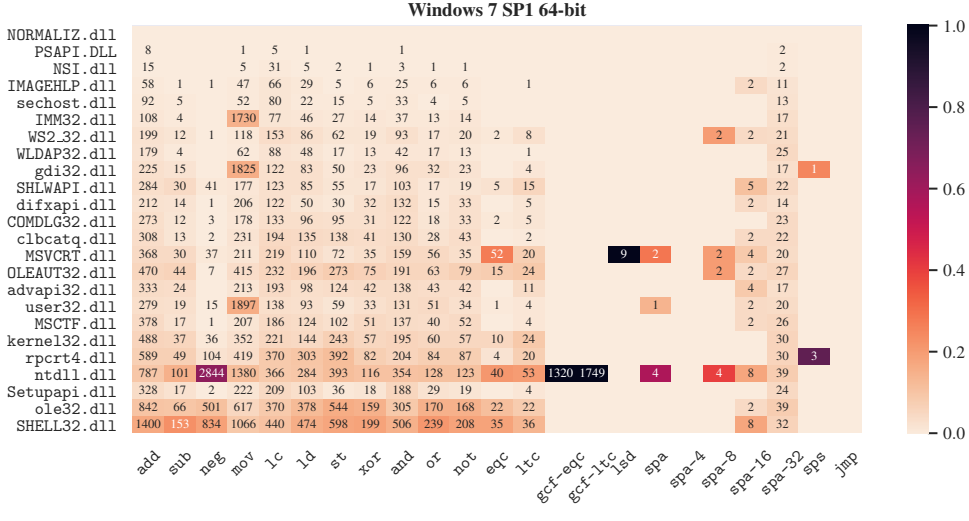
- Gadgets ROP de 10 bytes de tamaño (máximo)
- Sólo la instrucción `ret` como instrucción final
- No se cuentan gadgets ROPs repetidos
- **Sólo las definiciones actuales de las operaciones de ROPLANG**
- **Ampliadas operaciones a:**
  - `spa-4`, `spa-8`, `spa-16` y `spa-32`
  - `gcf` dividida en dos, `gcf-eqc` y `gcf-ltc`

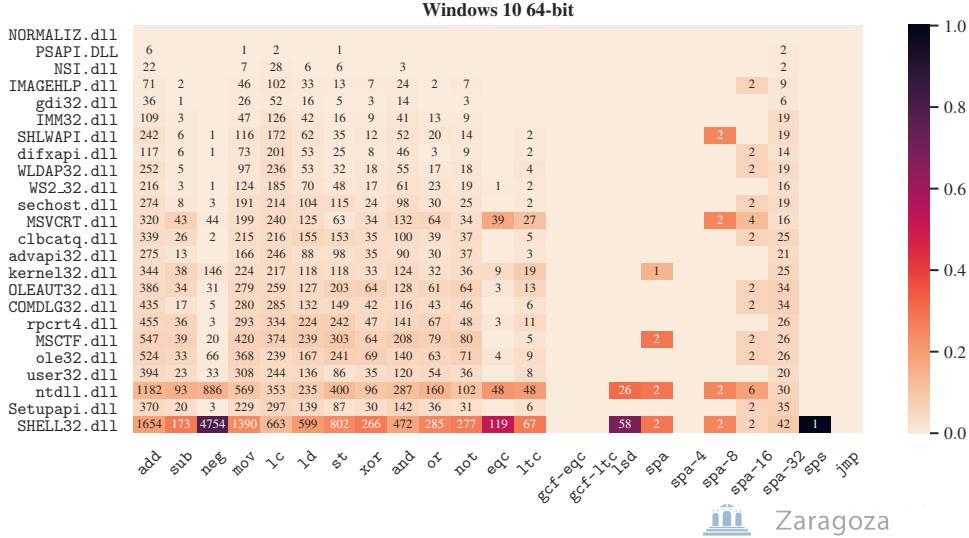
# Evaluación

## Prevalencia de gadgets ROP



# Evaluación

## Prevalence of ROP Gadgets – Discussion

- **Las operaciones de salto son poco frecuentes**, en ambas arquitecturas
  - No hay resultados para saltos incondicionales en 64 bits
- **Resultados variables en las otras operaciones virtuales**
  - Cuanto más grande es la DLL, mayor número de resultados (esperable)

Universidad Zaragoza

# Evaluación

## Prevalence of ROP Gadgets – Discussion

- **Las operaciones de salto son poco frecuentes**, en ambas arquitecturas
  - No hay resultados para saltos incondicionales en 64 bits
- **Resultados variables en las otras operaciones virtuales**
  - Cuanto más grande es la DLL, mayor número de resultados (esperable)
- **El número de resultados en Windows 10 siempre es mayor que en Windows 7, y lo mismo en 64 vs. 32 bits**
  - ¿Diferencias de tamaño en las DLLs?

Universidad Zaragoza

# Evaluación

## Simulación de una máquina de Turing – `mov` intermedio

- **Resultados muy limitados para las operaciones de salto**
  - **Operaciones obligatorias** para simular una máquina de Turing clásica

Universidad Zaragoza

# Evaluación

## Simulación de una máquina de Turing – `mov` intermedio

- **Resultados muy limitados para las operaciones de salto**
  - **Operaciones obligatorias** para simular una máquina de Turing clásica
- IDEA: **relajar dependencia de datos en operaciones simplemente añadiendo operaciones de asignación intermedias**
  Por ejemplo, con `mov(reg1, dst)`:
  - Mayor probabilidad de encontrar una operación `mov(reg1, dst)` adecuada
  - Por contra, incrementa el tamaño de la cadena ROP y por tanto, mayor probabilidad de efectos colaterales
- *Ejemplos de extensión*: `eqc(dst, src)`

```
sub(dst, src)   ⇒   sub(dst, src)
neg(dst)            mov(reg1, dst)
                    neg(reg1)
```

# Evaluación

## Simulación de una máquina de Turing – `mov` intermedio

- **Resultados muy limitados para las operaciones de salto**
  - **Operaciones obligatorias** para simular una máquina de Turing clásica
- IDEA: **relajar dependencia de datos en operaciones simplemente añadiendo operaciones de asignación intermedias**
  Por ejemplo, con `mov(reg1, dst)`:
  - Mayor probabilidad de encontrar una operación `mov(reg1, dst)` adecuada
  - Por contra, incrementa el tamaño de la cadena ROP y por tanto, mayor probabilidad de efectos colaterales
- *Ejemplos de extensión*: `eqc(dst, src)`

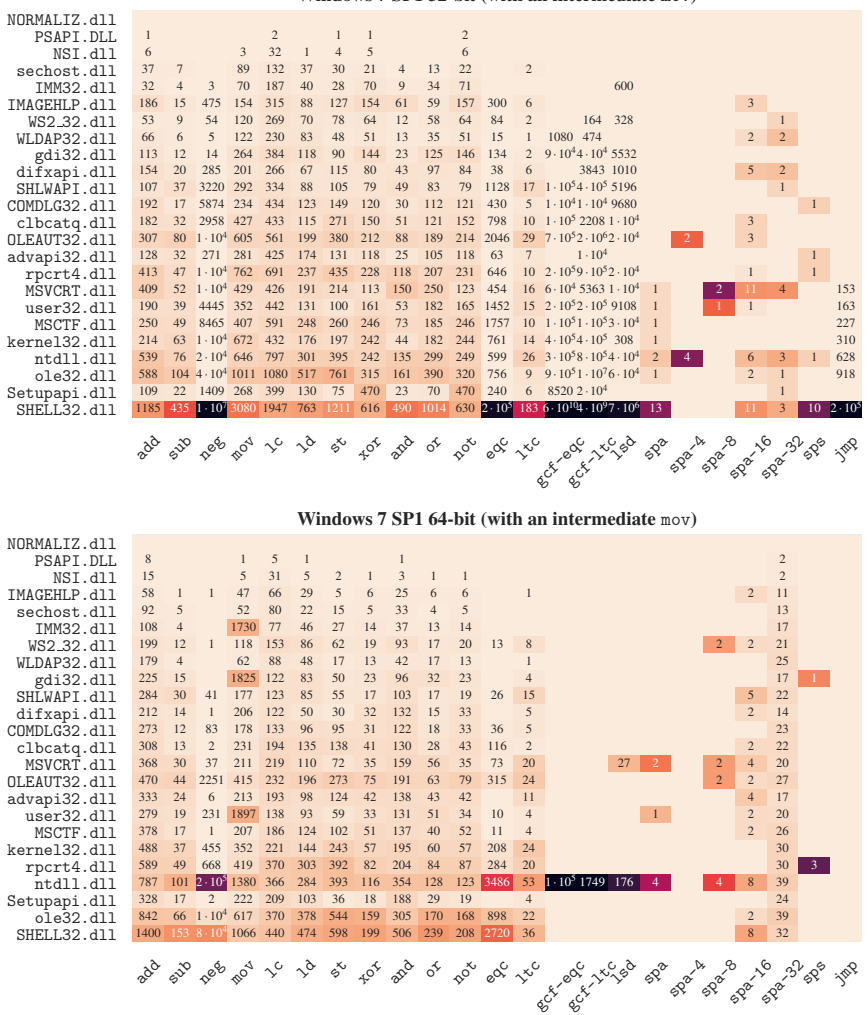
```
sub(dst, src)   ⇒   sub(dst, src)
neg(dst)            mov(reg1, dst)
                    neg(reg1)
```

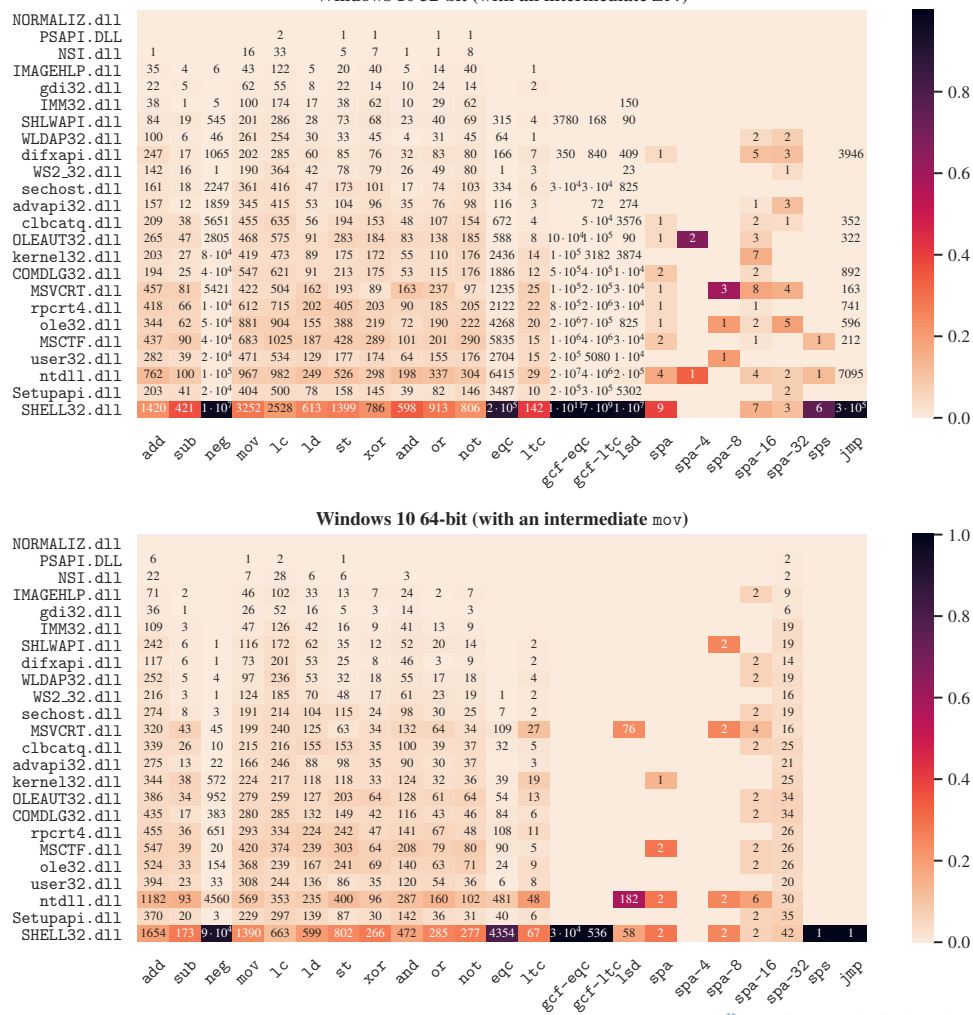
- Operaciones `neg`, `eqc`, `gcf`, `lsd`, y `jmp`
  - **Extendidas con operaciones `mov` intermedias**

Universidad Zaragoza

# Evaluación

## Simulación de una máquina de Turing – `mov` intermedio










Zaragoza
1542

# Evaluación

## Simulación de una máquina de Turing – Discusión

- **Más resultados en sistemas de 32 bits, más discretos en 64 bits**
  - **No hay operaciones de salto incondicional en Windows 7 SP1 64 bits**

Universidad Zaragoza

# Evaluación

## Simulación de una máquina de Turing – Discusión

- **Más resultados en sistemas de 32 bits, más discretos en 64 bits**
  - **No hay operaciones de salto incondicional en Windows 7 SP1 64 bits**

*Una concatenación de operaciones sofisticada incrementa la probabilidad de simular cualquier operación, si no se encuentra directamente*

- Extensión de operaciones virtuales posible y soportada por `ROP3`

Universidad Zaragoza

# Agenda



Universidad Zaragoza

# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333

- **Vulnerabilidad de Microsoft Office**
  - Versiones afectadas: Microsoft Office XP SP3, Office 2003 SP3, Office 2007 SP2, Office 2010, Office 2004 y 2008 para Mac, y Office para Mac 2011
- **Notificada en septiembre de 2010**
- Parche publicado en MS10-087 (publicado el 09 de noviembre, 2010)

Universidad Zaragoza

# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333

- **Vulnerabilidad de Microsoft Office**
  - Versiones afectadas: Microsoft Office XP SP3, Office 2003 SP3, Office 2007 SP2, Office 2010, Office 2004 y 2008 para Mac, y Office para Mac 2011
- **Notificada en septiembre de 2010**
- Parche publicado en MS10-087 (publicado el 09 de noviembre, 2010)
- **Noviembre de 2012: ataque al NATO's Special Operations Headquarters**
  - Vía **spear phishing**, con RTF adjunto explotando CVE-2010-333
  - Un archivo RTF comienza con `{rtf1}` y contiene texto no formateado, palabras de control, símbolos, etc., entre llaves

```
{\rtf1{
....
{\shp{\sp{\sn pFragments}{\sv value}}}
}
}
```

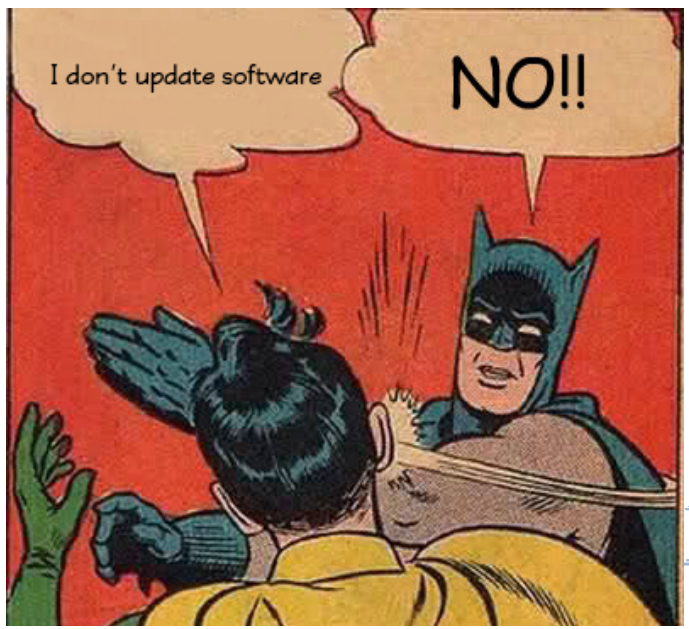
Universidad Zaragoza

# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333

- **Vulnerabilidad de Microsoft Office**
  - Versiones afectadas: Microsoft Office XP SP3, Office 2003 SP3, Office 2007 SP2, Office 2010, Office 2004 y 2008 para Mac, y Office para Mac 2011
- **Notificada en septiembre de 2010**
- Parche publicado en MS10-087 (publicado el 09 de noviembre, 2010)
- **Noviembre de 2012: ataque al NATO's Special Operations Headquarters**
  - Vía **spear phishing**, con RTF adjunto explotando CVE-2010-333
  - Un archivo RTF comienza con `{rtf1}` y contiene texto no formateado, palabras de control, símbolos, etc., entre llaves

```
{\rtf1{
....
{\shp{\sp{\sn pFragments}{\sv value}}}
}
}
```




Universidad Zaragoza

# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333

- **BOF de pila, en la función encargada de interpretar el fichero RTF**
- DLL de interés: `MSO.DLL` 11.0.5606
  - MD5 251C11444F614DE5FA47ECF7275E7BF1
  - Microsoft Office 2003 suite

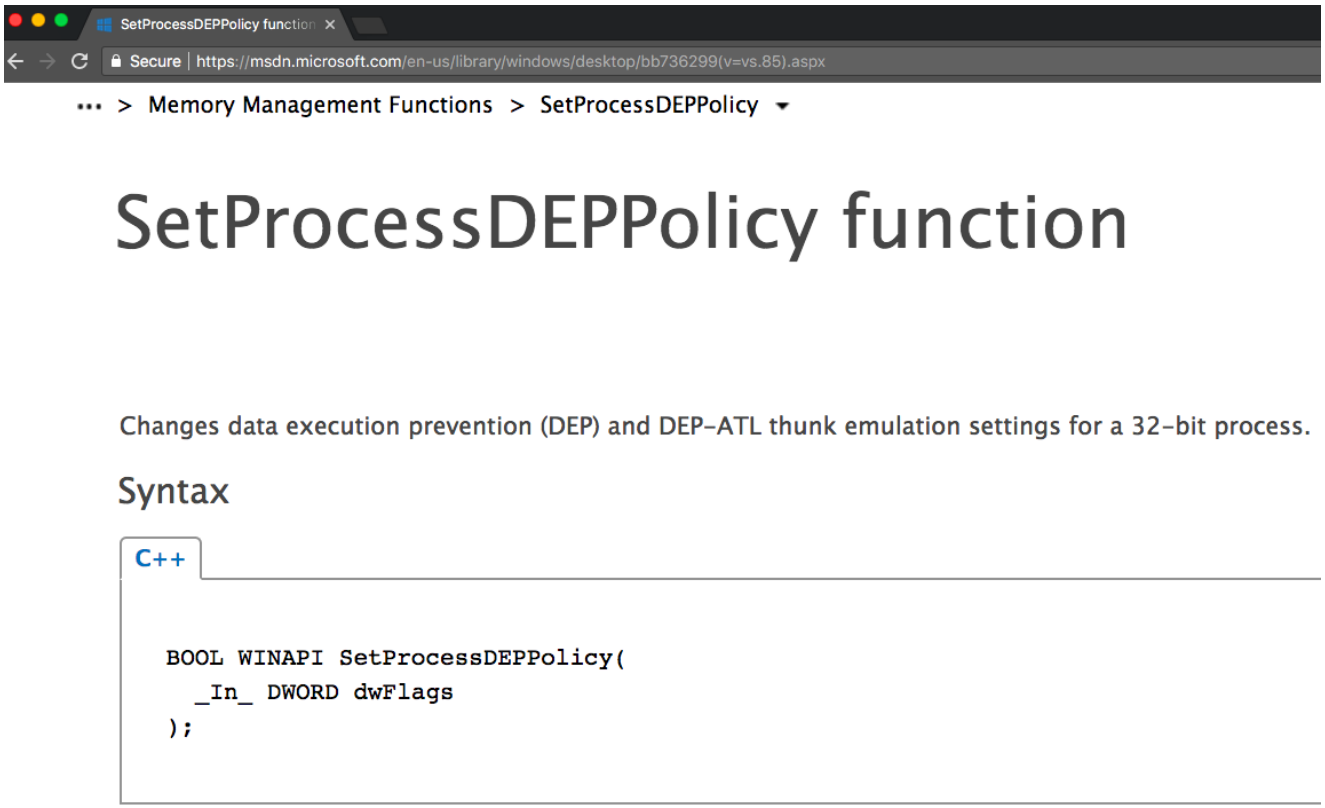
```
0x30f4cc5d  push ebp
0x30f4cc5e  mov ebp, esp
0x30f4cc60  sub esp, 0x14
(...)
0x30f4cc93  call dword [eax + 0x1c] ;  calls to MSO.30e9eb62
0x30f4cc96  mov eax, dword [ebp + 0x14]
0x30f4cc99  push dword [ebp + 0x18]
0x30f4cc9c  mov edx, dword [ebp - 0x10]
0x30f4cc9f  neg eax
0x30f4cca1  sbb eax, eax
0x30f4cca3  lea ecx, [ebp - 8]
0x30f4cca6  and eax, ecx
0x30f4cca8  push eax
0x30f4cca9  push dword [ebp + 8]
0x30f4ccac  call 0x30f4cb1d
0x30f4ccb1  test al, al
0x30f4ccb3  je 0x30f4cd51
(...)
0x30f4cd51  pop esi
0x30f4cd52  pop ebx
0x30f4cd53  pop edi
0x30f4cd54  leave
0x30f4cd55  ret 0x14
```

```
0x30e9eb62  push edi
0x30e9eb63  mov edi, dword [esp + 0xc]
0x30e9eb67  test edi, edi
0x30e9eb69  je 0x30e9eb92
0x30e9eb6b  mov eax, dword [esp + 8]
0x30e9eb6f  mov ecx, dword [eax + 8]
0x30e9eb72  and ecx, 0xffff
0x30e9eb78  push esi
0x30e9eb79  mov esi, ecx
0x30e9eb7b  imul esi, dword [esp + 0x14]
0x30e9eb80  add esi, dword [eax + 0x10]
0x30e9eb83  mov eax, ecx
0x30e9eb85  shr ecx, 2
0x30e9eb88  rep movsd es:[edi], dword ptr [esi]
0x30e9eb8a  mov ecx, eax
0x30e9eb8c  and ecx, 3
0x30e9eb8f  rep movsb es:[edi], byte ptr [esi]
0x30e9eb91  pop esi
0x30e9eb92  pop edi
0x30e9eb93  ret 0xc
```

Universidad Zaragoza 1542

# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333




- **Hay que llamar con un 0 para deshabilitar la protección W⊕X** 🙂
  - Vamos a asumir que la dirección de esta función la conocemos

Universidad Zaragoza

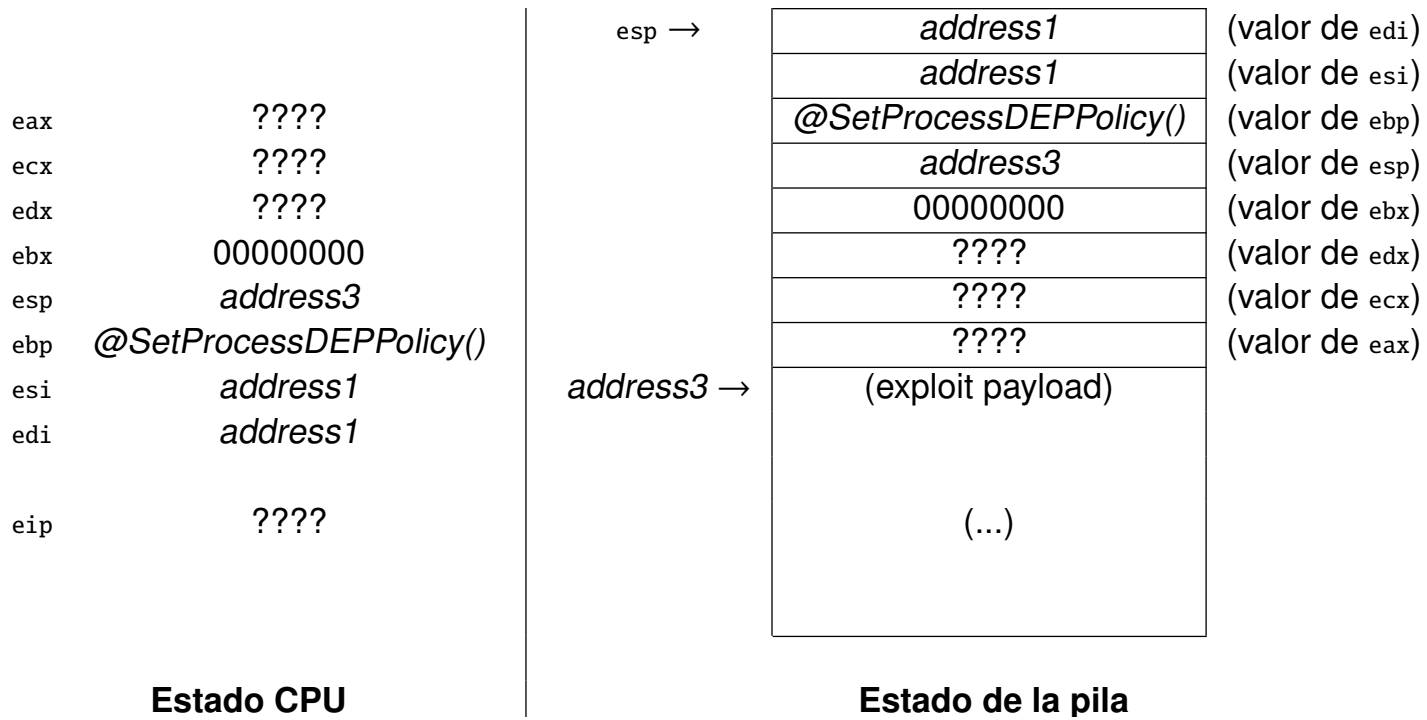
# Demo: explotación de sistema protegido Windows

## Caso de estudio: CVE-2010-3333

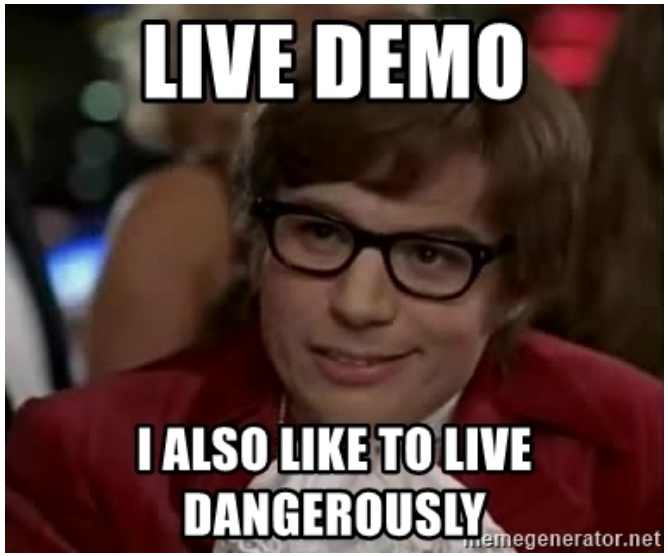
INSTRUCTION SET REFERENCE, N-Z

### PUSHA/PUSHAD—Push All General-Purpose Registers

| Opcode | Instruction | Op/ En | 64-Bit Mode | Compat/ Leg Mode | Description |
|---|---|---|---|---|---|
| 60 | PUSHA | A | Invalid | Valid | Push AX, CX, DX, BX, original SP, BP, SI, and DI. |
| 60 | PUSHAD | A | Invalid | Valid | Push EAX, ECX, EDX, EBX, original ESP, EBP, ESI, and EDI. |



# Demo: explotación de sistema protegido Windows




## Pasos

1. Estudiar la vulnerabilidad y condiciones de explotación
2. Diseñar la cadena ROP, mediante operaciones virtuales de ROPLang
3. Construir la cadena automáticamente con ROP3
4. Crear el fichero de shellcode adecuado

Universidad Zaragoza

# Agenda



Universidad Zaragoza

# Conclusiones

- **Lenguaje virtual,** ROPLANG**, cuyas operaciones mapean gadgets ROP**
- **Herramienta** ROP3**, permite a un usuario encontrar gadgets ROP y construir una cadena ROP especificada mediante** ROPLANG

Universidad Zaragoza

# Conclusiones

- **Lenguaje virtual,** ROPLANG**, cuyas operaciones mapean gadgets ROP**
- **Herramienta** ROP3**, permite a un usuario encontrar gadgets ROP y construir una cadena ROP especificada mediante** ROPLANG

- **Cualquier operación virtual, siendo las de salto las menos frecuentes**
  - **Un entrelazado de operaciones virtuales bien pensado puede servir para encontrar operaciones que no se encuentran directamente**
- El tamaño del fichero impacta en la prevalencia de los gadgets ROP

Universidad Zaragoza
1542

# Conclusiones

- **Lenguaje virtual,** ROPLANG**, cuyas operaciones mapean gadgets ROP**
- **Herramienta** ROP3**, permite a un usuario encontrar gadgets ROP y construir una cadena ROP especificada mediante** ROPLANG

- **Cualquier operación virtual, siendo las de salto las menos frecuentes**
  - **Un entrelazado de operaciones virtuales bien pensado puede servir para encontrar operaciones que no se encuentran directamente**
- El tamaño del fichero impacta en la prevalencia de los gadgets ROP

## Trabajo futuro

- **Evaluar el poder del adversario en otros sistemas operativos**

Universidad Zaragoza

# Cuando ROP conoce a Turing: de BOF a LOL en 4 sencillos pasos

**Ricardo J. Rodríguez**



rjrodriguez@unizar.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

Universidad Zaragoza

Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza

11 de noviembre, 2021

**Salón de Innovación y Emprendimiento**
Huesca, España