Buffer overflows: qué son y cómo evitarlos

Ricardo J. Rodríguez

All wrongs reversed

rjrodriguez@fi.upm.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

Universidad Politécnica de Madrid
Madrid, Spain

29 de Noviembre, 2013

BetaBeers

Zaragoza (España)

$whoami

- Miembro de CLS desde los principios (2001)
- Ph.D. por la Universidad de Zaragoza (2013)
- Actualmente trabajando para la UPM

$whoami

- Miembro de CLS desde los principios (2001)
- Ph.D. por la Universidad de Zaragoza (2013)
- Actualmente trabajando para la UPM
  - Análisis de rendimiento de sistemas complejos
  - Ingeniería del Software segura
  - Sistemas Tolerantes a Fallos (diseño y análisis)
  - Análisis malware (técnicas, morfología, etc.)
  - Análisis *safety* en sistemas basados en componentes

$whoami

- Miembro de CLS desde los principios (2001)
- Ph.D. por la Universidad de Zaragoza (2013)
- Actualmente trabajando para la UPM
  - Análisis de rendimiento de sistemas complejos
  - Ingeniería del Software segura
  - Sistemas Tolerantes a Fallos (diseño y análisis)
  - Análisis malware (técnicas, morfología, etc.)
  - Análisis *safety* en sistemas basados en componentes
- Formador en NcN, RootedCON, HIP...
- Ponente en NcN, HackLU, RootedCON, STIC CCN-CERT, HIP...

Agenda

Agenda

¿Qué es un BOF? (I)

```
void readName()
{
  char username[256];
  printf("Nombre de usuario: ");
  scanf("%s", username);
}
```

¿Qué es un BOF? (I)

```
void readName()
{
  char username[256];
  printf("Nombre de usuario: ");
  scanf("%s", username);
}
```

```
void copyBuffers(char *org, char *dst)
{
  char buffer[5000];
  strcpy(buffer, org);
  // Do some stuff into your buffer
  strcpy(dst, buffer);
}
```

¿Qué es un BOF? (I)

```c
void readName()
{
  char username[256];
  printf("Nombre de usuario: ");
  scanf("%s", username);
}
```

```c
void copyBuffers(char *org, char *dst)
{
  char buffer[5000];
  strcpy(buffer, org);
  // Do some stuff into your buffer
  strcpy(dst, buffer);
}
```

Buffer Overflow (BOF)

- Desbordamiento (*overflow*) del buffer (zona de memoria)

¿Qué es un BOF? (I)

```
void readName()
{
  char username[256];
  printf("Nombre de usuario: ");
  scanf("%s", username);
}
```

```
void copyBuffers(char *org, char *dst)
{
  char buffer[5000];
  strcpy(buffer, org);
  // Do some stuff into your buffer
  strcpy(dst, buffer);
}
```

Buffer Overflow (BOF)

- Desbordamiento (*overflow*) del buffer (zona de memoria)
- Consecuencia habitual: ejecución de código arbitrario
  - Código arbitrario: cualquier código, indiferente

¿Qué es un BOF? (I)

```
void readName()
{
  char username[256];
  printf("Nombre de usuario: ");
  scanf("%s", username);
}
```

```
void copyBuffers(char *org, char *dst)
{
  char buffer[5000];
  strcpy(buffer, org);
  // Do some stuff into your buffer
  strcpy(dst, buffer);
}
```

Buffer Overflow (BOF)

- Desbordamiento (*overflow*) del buffer (zona de memoria)
- Consecuencia habitual: ejecución de código arbitrario
  - Código arbitrario: cualquier código, indiferente
- ¿Se usa el BOF?
  - Método habitual para ejecución de código malicioso (aka *malware*)

¿Qué es un BOF? (II)

¿Algo más?

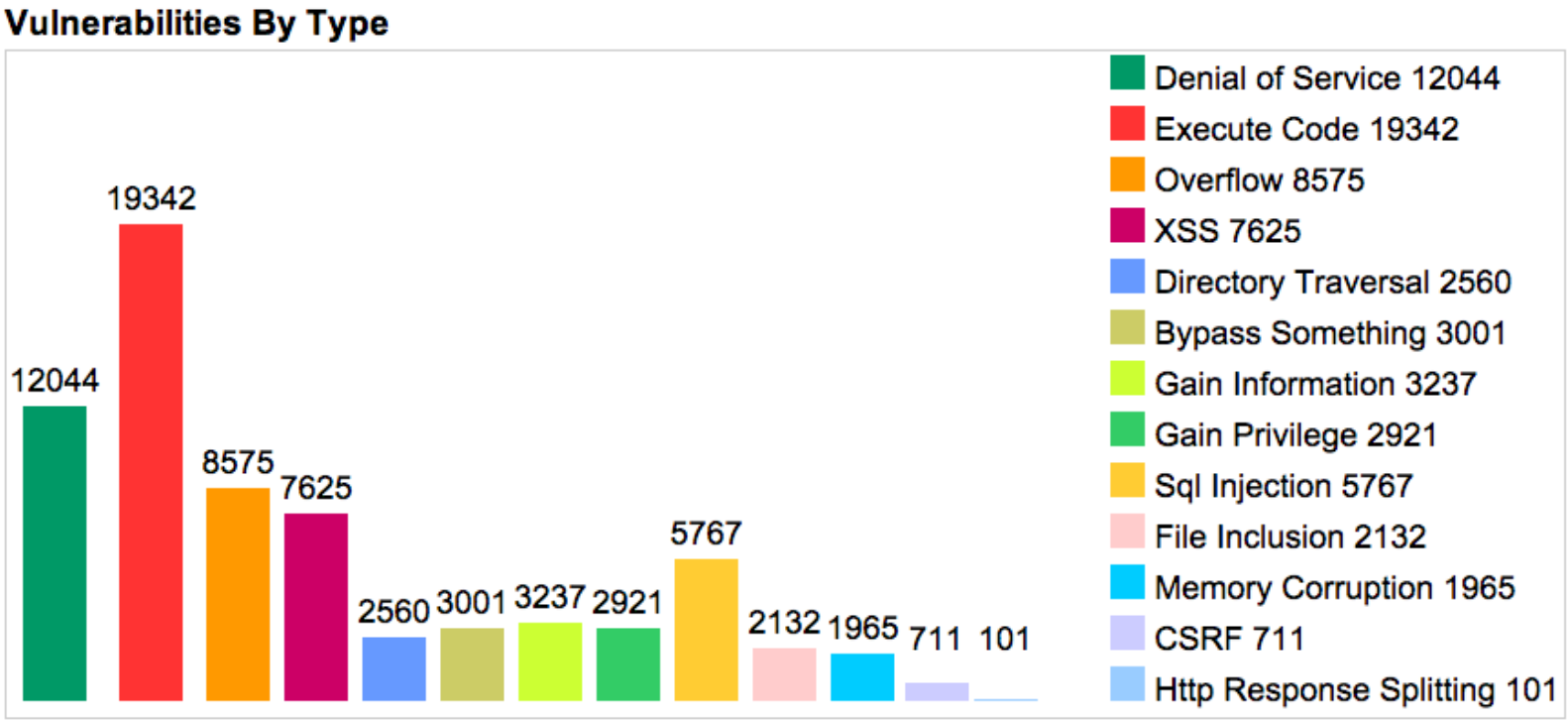
- Causa DoS (denegación de servicio de la aplicación)
  - La aplicación termina de forma no controlada, se "rompe"

¿Qué es un BOF? (II)

¿Algo más?

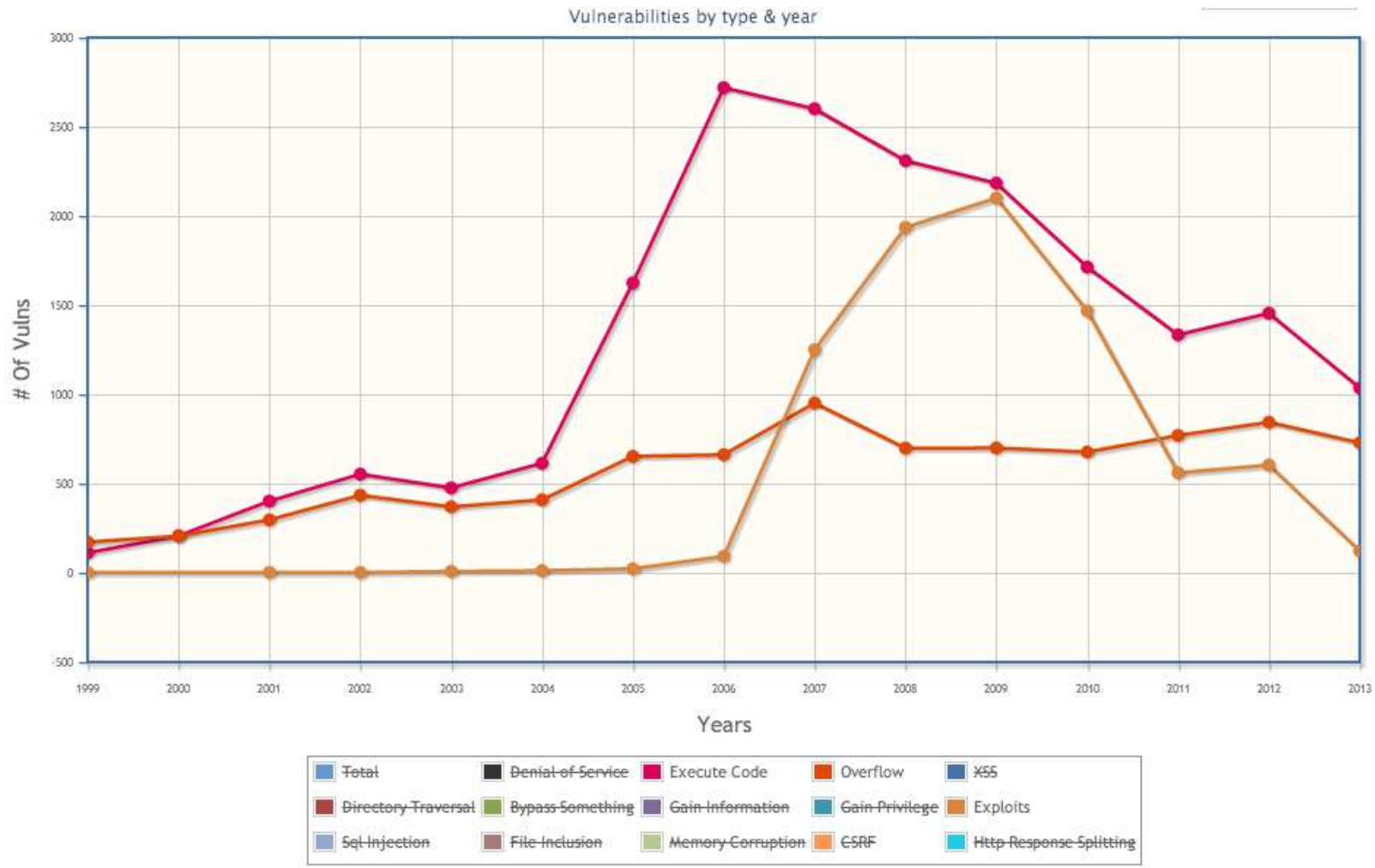
- Causa DoS (denegación de servicio de la aplicación)
  - La aplicación termina de forma no controlada, se "rompe"
- Definición Wikipedia (overflow):
  - *"a buffer overflow, or buffer overrun, is an anomaly where a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory. This is a special case of violation of memory safety' '*
- Problema creciente

¿Qué es un BOF? (III)



(Cortesía de `www.cvedetails.com`, datos de 1999 a 2013)

¿Qué es un BOF? (IV)



(Cortesía de `www.cvedetails.com`, datos de 1999 a 2013)

¿Qué es un BOF? (V)

Tipos de desbordamientos

- Basados en la pila (*stack-based BOF*)
  - La pila: declaración de variables locales, inicialización de variables
  - Datos para el flujo de ejecución
    - Dirección de retorno
    - Manejadores de excepciones

¿Qué es un BOF? (V)

Tipos de desbordamientos

- Basados en la pila (*stack-based BOF*)
  - La pila: declaración de variables locales, inicialización de variables
  - Datos para el flujo de ejecución
    - Dirección de retorno
    - Manejadores de excepciones
  - Consecuencia: ejecución de código arbitrario (redirección de ejecución)

¿Qué es un BOF? (V)

Tipos de desbordamientos

- Basados en la pila (*stack-based BOF*)
  - La pila: declaración de variables locales, inicialización de variables
  - Datos para el flujo de ejecución
    - Dirección de retorno
    - Manejadores de excepciones
  - Consecuencia: ejecución de código arbitrario (redirección de ejecución)
- Basados en el heap (*heap-based BOF*)
  - Sobreescritura de memoria reservada (`malloc`, `allocate`)

¿Qué es un BOF? (V)

Tipos de desbordamientos

- Basados en la pila (*stack-based BOF*)
  - La pila: declaración de variables locales, inicialización de variables
  - Datos para el flujo de ejecución
    - Dirección de retorno
    - Manejadores de excepciones
  - Consecuencia: ejecución de código arbitrario (redirección de ejecución)
- Basados en el heap (*heap-based BOF*)
  - Sobreescritura de memoria reservada (`malloc`, `allocate`)
  - Consecuencia: corrupción de memoria, ejecución de código
- ...

¿Qué es un BOF? (VI)

Tipos de desbordamientos

- ...
- Off-by-one
  - Una iteración de n pasos que se realiza (n - 1) pasos
  - Consecuencia: reescritura (controlada) de un byte del EIP

¿Qué es un BOF? (VI)

Tipos de desbordamientos

- ...
- Off-by-one
  - Una iteración de n pasos que se realiza (n - 1) pasos
  - Consecuencia: reescritura (controlada) de un byte del EIP
- Buffer Overrun
  - Cuello de botella en los bloques de memoria de grabadores de CD/DVD
  - Sobreescritura buffer $\rightarrow$ bonito CD (o DVD) posavasos/espantapájaros

¿Qué es un BOF? (VI)

Tipos de desbordamientos

- ...
- Off-by-one
  - Una iteración de n pasos que se realiza (n - 1) pasos
  - Consecuencia: reescritura (controlada) de un byte del EIP
- Buffer Overrun
  - Cuello de botella en los bloques de memoria de grabadores de CD/DVD
  - Sobreescritura buffer $\rightarrow$ bonito CD (o DVD) posavasos/espantapájaros
- Integer OF

¿Qué es un BOF? (VI)

Tipos de desbordamientos

- ...
- Off-by-one
  - Una iteración de n pasos que se realiza (n - 1) pasos
  - Consecuencia: reescritura (controlada) de un byte del EIP
- Buffer Overrun
  - Cuello de botella en los bloques de memoria de grabadores de CD/DVD
  - Sobreescritura buffer $\rightarrow$ bonito CD (o DVD) posavasos/espantapájaros
- Integer OF

En esta charla, nos centramos en Stack-based BOF

¿Qué es un BOF? (VII)

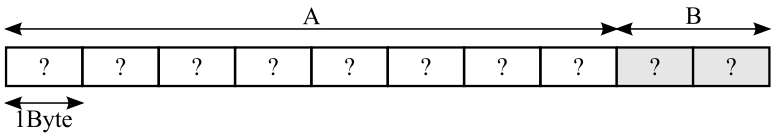
```
char           A[8];
unsigned short B;
```

- Variable A: 8B (1 char $\rightarrow$ 1B)
- Variable B: 2B
  - No inicializadas

¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

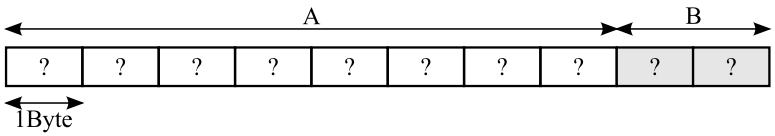
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



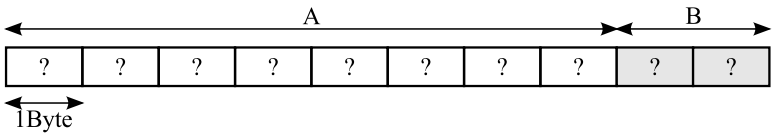
- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

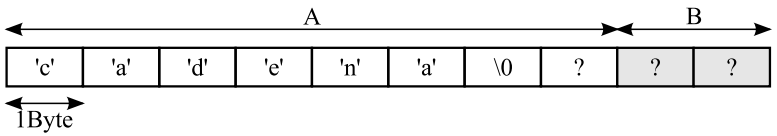
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

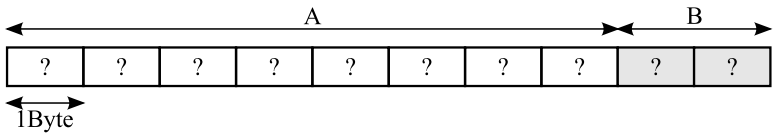
- ¿Cómo queda la memoria?



¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

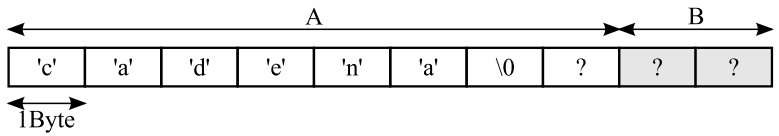
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

- ¿Cómo queda la memoria?



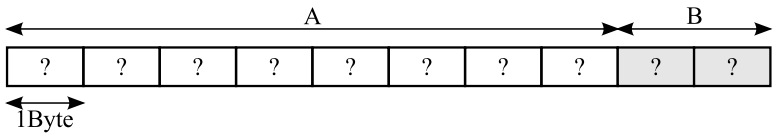
- ¿Y si copiamos una más larga?

```
strcpy(A, "cadena larga");
```

¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

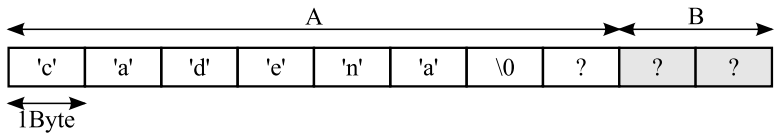
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

- ¿Cómo queda la memoria?



- ¿Y si copiamos una más larga?

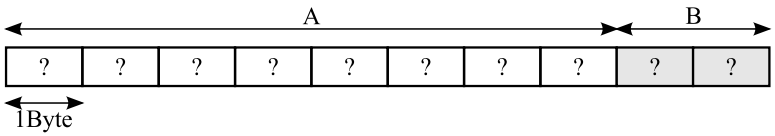
```
strcpy(A, "cadena larga");
```

- ¿Cómo queda la memoria?

¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

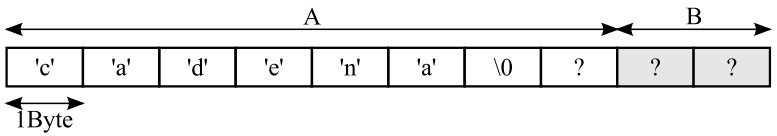
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

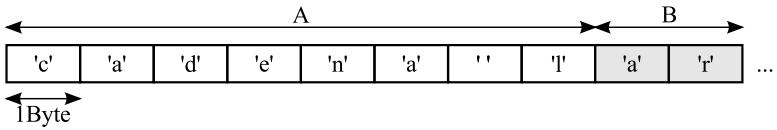
- ¿Cómo queda la memoria?



- ¿Y si copiamos una más larga?

```
strcpy(A, "cadena larga");
```

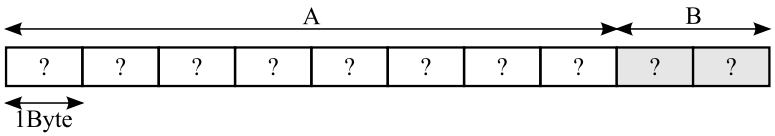
- ¿Cómo queda la memoria?



¿Qué es un BOF? (VII)

```
char           A[8];
unsigned short B;
```

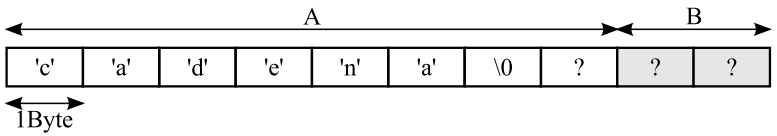
- Variable A: 8B (1 char → 1B)
- Variable B: 2B
  - No inicializadas



- Copiemos una cadena a A...

```
strcpy(A, "cadena");
```

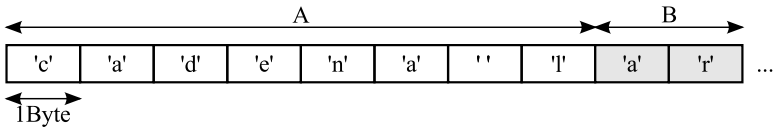
- ¿Cómo queda la memoria?



- ¿Y si copiamos una más larga?

```
strcpy(A, "cadena larga");
```

- ¿Cómo queda la memoria?



Sobreescritura de memoria adyacente

Agenda

Stack-based BOFs: de la teoría a la práctica (I)

Stack-based BOF

- La pila: declaración de variables locales

Stack-based BOFs: de la teoría a la práctica (I)

Stack-based BOF

- La pila: declaración de variables locales
- Datos para el flujo de ejecución
  - Dirección de retorno
  - Manejadores de excepciones

Stack-based BOFs: de la teoría a la práctica (I)

Stack-based BOF

- La pila: declaración de variables locales
- Datos para el flujo de ejecución
  - Dirección de retorno
  - Manejadores de excepciones
- Consecuencia: ejecución de código arbitrario (redirección de ejecución)

Stack-based BOFs: de la teoría a la práctica (II)

Vuelta a los clásicos: *classic BOF* (CWE-120)

- `http://cwe.mitre.org/data/definitions/120.html`
- *"the program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow."*

Stack-based BOFs: de la teoría a la práctica (II)

Vuelta a los clásicos: *classic BOF* (CWE-120)

- `http://cwe.mitre.org/data/definitions/120.html`
- *"the program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow."*
- Funciones típicas explotables (lenguaje C)
  - `strcpy()`, `strcat()`
  - `scanf()`, `gets()`
  - Familia `printf()`: `sprintf()`, `vsprintf()`, ...
  - `https://security.web.cern.ch/security/recommendations/en/codetools/c.shtml`

```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                    then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                    Well coding!\n", username);
        return;
}
```

```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                    then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                    Well coding!\n", username);
        return;
}
```

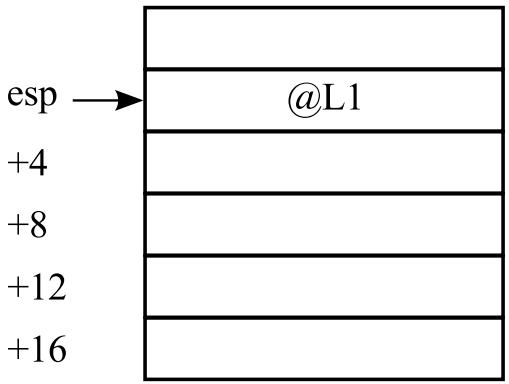
```
LC0: .ascii "Enter your username for login, and ... \0"
LC1: .ascii "%s\0"
LC2: .ascii "Hi %s, welcome back! Well coding!\12\0"
        .text
_readCredentials:
        push    ebp
        mov     ebp, esp
        sub     esp, 40
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC2
        call    _printf
        leave
        ret
L1:
```

```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                    then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                    Well coding!\n", username);
        return;
}
```

```
LC0: .ascii "Enter your username for login, and ... \0"
LC1: .ascii "%s\0"
LC2: .ascii "Hi %s, welcome back! Well coding!\12\0"
        .text
_readCredentials:
        push    ebp
        mov     ebp, esp
        sub     esp, 40
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC2
        call    _printf
        leave
        ret
L1:
```

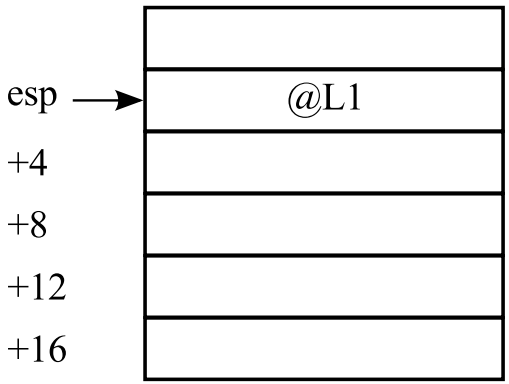
```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                Well coding!\n", username);
        return;
}
```

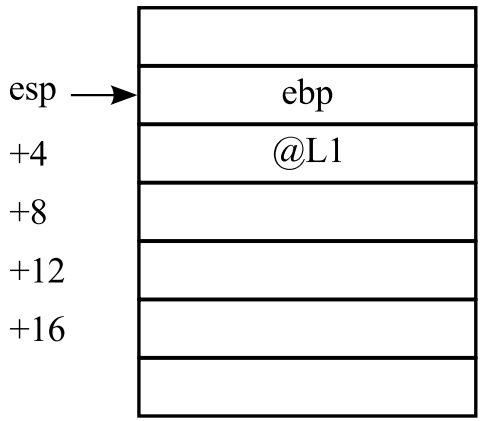
```
LC0: .ascii "Enter your username for login, and ... \0"
LC1: .ascii "%s\0"
LC2: .ascii "Hi %s, welcome back! Well coding!\12\0"
        .text
_readCredentials:
        push    ebp
        mov     ebp, esp
        sub     esp, 40
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC2
        call    _printf
        leave
        ret
L1:
```



```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                Well coding!\n", username);
        return;
}
```

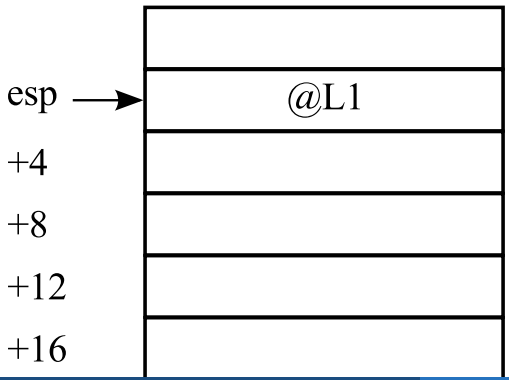
```
LC0: .ascii "Enter your username for login, and ... \0"
LC1: .ascii "%s\0"
LC2: .ascii "Hi %s, welcome back! Well coding!\12\0"
        .text
_readCredentials:
        push    ebp
        mov     ebp, esp
        sub     esp, 40
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC2
        call    _printf
        leave
        ret
L1:
```

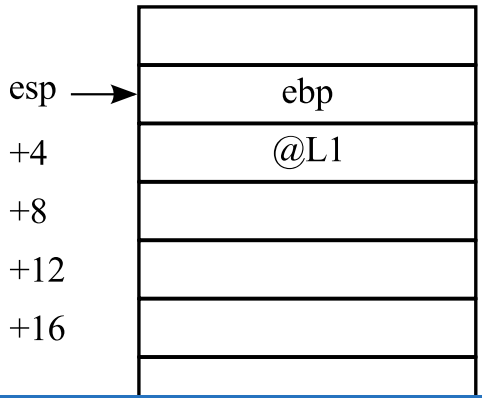


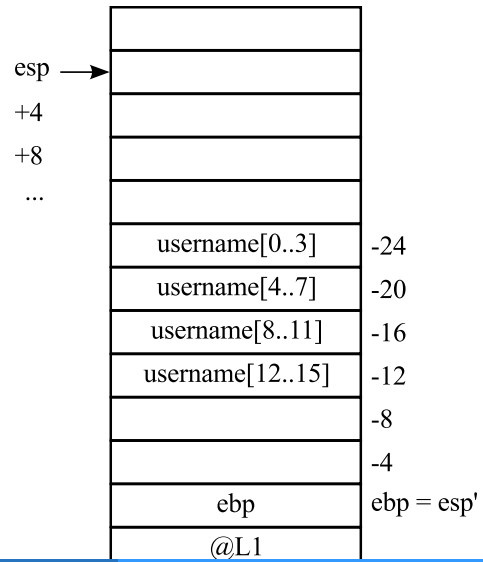


```
void readCredentials()
{
        /* Create an array for storing
                some dummy data */
        char username[16];
        printf("Enter your username for login, and
                then press <Enter>: ");
        scanf("%s", username);
        printf("Hi %s, welcome back!
                Well coding!\n", username);
        return;
}
```

```
LC0: .ascii "Enter your username for login, and ... \0"
LC1: .ascii "%s\0"
LC2: .ascii "Hi %s, welcome back! Well coding!\12\0"
        .text
_readCredentials:
        push    ebp
        mov     ebp, esp
        sub     esp, 40
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC1
        call    _scanf
        lea     eax, [ebp-24]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC2
        call    _printf
        leave
        ret
L1:
```







It's demo time!

Agenda

Mecanismos para Evitación de Stack-based BOFs (I)

Stack Cookies

- También llamado Stack Canaries
- Opción de compilación (`/GSswitch`)

Mecanismos para Evitación de Stack-based BOFs (I)

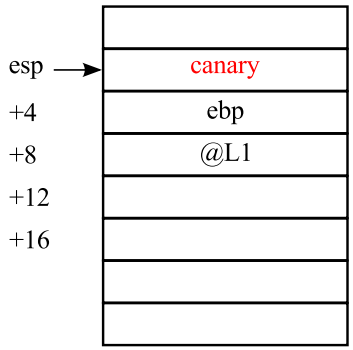
Stack Cookies

- También llamado Stack Canaries
- Opción de compilación (`/GSswitch`)
- Añade prólogo y epílogo a las funciones
  - Cálculo de cookie (dword, unsigned int)
  - Copiado a la pila, y comprobado al final

Mecanismos para Evitación de Stack-based BOFs (I)

Stack Cookies

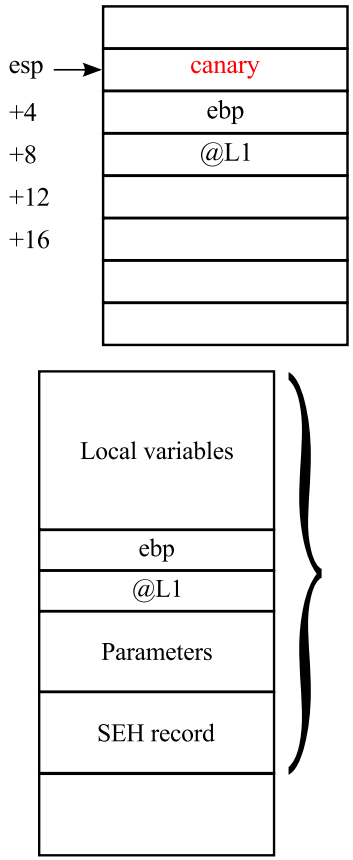
- También llamado Stack Canaries
- Opción de compilación (`/GSswitch`)
- Añade prólogo y epílogo a las funciones
  - Cálculo de cookie (dword, unsigned int)
  - Copiado a la pila, y comprobado al final
- ¿Cómo evitarla? (algunas técnicas)
  - Explotación de manejadores de excepciones



Mecanismos para Evitación de Stack-based BOFs (I)

Stack Cookies

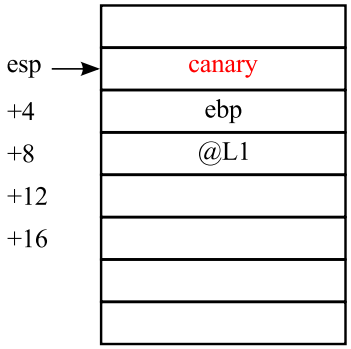
- También llamado Stack Canaries
- Opción de compilación (`/GSswitch`)
- Añade prólogo y epílogo a las funciones
  - Cálculo de cookie (dword, unsigned int)
  - Copiado a la pila, y comprobado al final
- ¿Cómo evitarla? (algunas técnicas)
  - Explotación de manejadores de excepciones

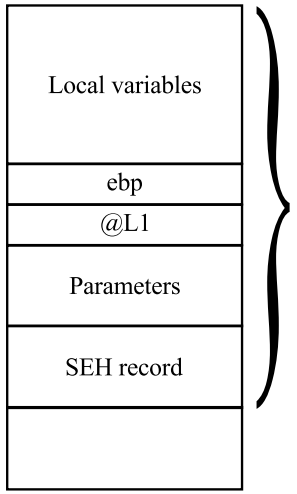


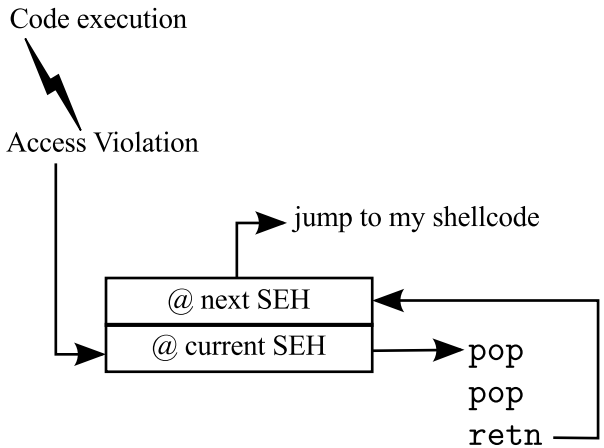
Mecanismos para Evitación de Stack-based BOFs (I)

Stack Cookies

- También llamado Stack Canaries
- Opción de compilación (`/GSswitch`)
- Añade prólogo y epílogo a las funciones
  - Cálculo de cookie (dword, unsigned int)
  - Copiado a la pila, y comprobado al final
- ¿Cómo evitarla? (algunas técnicas)
  - Explotación de manejadores de excepciones







Mecanismos para Evitación de Stack-based BOFs (II)

SafeSEH

- Opción de compilación (`/safeSEH`)
- Aplicable a cualquier módulo ejecutable (exe, dlls, etc.)

Mecanismos para Evitación de Stack-based BOFs (II)

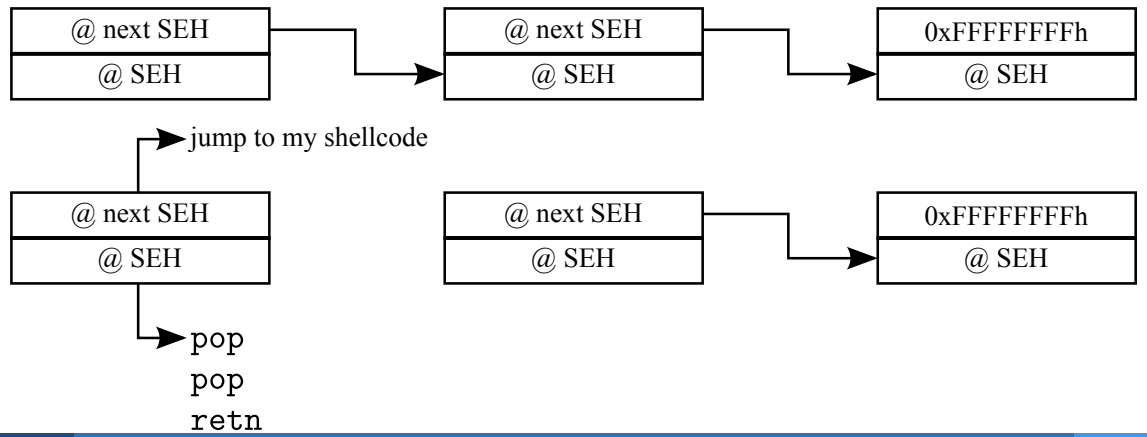
SafeSEH

- Opción de compilación (`/safeSEH`)
- Aplicable a cualquier módulo ejecutable (exe, dlls, etc.)
- SEHOP: Structured Exception Handler Overwrite Protection
  - Protección de la cadena de manejadores de excepciones
  - Detección de modificación en cadena SEH $\rightarrow$ acaba ejecución

Mecanismos para Evitación de Stack-based BOFs (II)

SafeSEH

- Opción de compilación (`/safeSEH`)
- Aplicable a cualquier módulo ejecutable (exe, dlls, etc.)
- SEHOP: Structured Exception Handler Overwrite Protection
  - Protección de la cadena de manejadores de excepciones
  - Detección de modificación en cadena SEH → acaba ejecución
- ¿Cómo evitarla? (algunas técnicas)
  - Módulo cargado sin SafeSEH
  - Redirección a SEH que permite alcanzar código controlable



Mecanismos para Evitación de Stack-based BOFs (III)

Data Execution Prevention (DEP)

- Diferentes opciones:
  - `OptIn`: sólo el núcleo/módulos del sistema protegidos
  - `OptOut`: todo protegido, menos algunas aplicaciones
  - `AlwaysOn`: todo, sin excepción (no deshabilitable en ejecución)
  - `AlwaysOff`: deshabilitación de DEP (no habilitable en ejecución)

Mecanismos para Evitación de Stack-based BOFs (III)

Data Execution Prevention (DEP)

- Diferentes opciones:
  - `OptIn`: sólo el núcleo/módulos del sistema protegidos
  - `OptOut`: todo protegido, menos algunas aplicaciones
  - `AlwaysOn`: todo, sin excepción (no deshabilitable en ejecución)
  - `AlwaysOff`: deshabilitación de DEP (no habilitable en ejecución)
    - Windows 64bits está siempre habilitado
    - IExplorer es una aplicación de 32bits...

Mecanismos para Evitación de Stack-based BOFs (III)

Data Execution Prevention (DEP)

- Diferentes opciones:
  - `OptIn`: sólo el núcleo/módulos del sistema protegidos
  - `OptOut`: todo protegido, menos algunas aplicaciones
  - `AlwaysOn`: todo, sin excepción (no deshabilitable en ejecución)
  - `AlwaysOff`: deshabilitación de DEP (no habilitable en ejecución)
    - Windows 64bits está siempre habilitado
    - IExplorer es una aplicación de 32bits...
- Opción de compilación (`/NXCOMPAT`, *Permanent DEP*)

Mecanismos para Evitación de Stack-based BOFs (III)

Data Execution Prevention (DEP)

- Diferentes opciones:
  - `OptIn`: sólo el núcleo/módulos del sistema protegidos
  - `OptOut`: todo protegido, menos algunas aplicaciones
  - `AlwaysOn`: todo, sin excepción (no deshabilitable en ejecución)
  - `AlwaysOff`: deshabilitación de DEP (no habilitable en ejecución)
    - Windows 64bits está siempre habilitado
    - IExplorer es una aplicación de 32bits...
- Opción de compilación (`/NXCOMPAT`, *Permanent DEP*)
- Cambio en arranque de sistema (`boot.ini`)
- ¿Cómo evitarla? (algunas técnicas)
  - BOFs basados en SEH
  - Retn-Oriented Programming (ROP) (+ `LdrpCheckNXCompatibility`)

Mecanismos para Evitación de Stack-based BOFs (IV)

Address Space Layout Randomization (ASLR)

- Direcciones base de ejecutables/dlls/pila/heap variable
  - Desde Windows Vista / 7 / 2008
  - CAMBIAN en cada reinicio
  - Activado por defecto (excepto Internet Explorer 7)

Mecanismos para Evitación de Stack-based BOFs (IV)

Address Space Layout Randomization (ASLR)

- Direcciones base de ejecutables/dlls/pila/heap variable
  - Desde Windows Vista / 7 / 2008
  - CAMBIAN en cada reinicio
  - Activado por defecto (excepto Internet Explorer 7)
- Opción de compilación `/DYNAMICBASE` (VS2005 SP1 en adelante)
- Para ser efectivo, debe ser complementado con DEP

Mecanismos para Evitación de Stack-based BOFs (IV)

Address Space Layout Randomization (ASLR)

- Direcciones base de ejecutables/dlls/pila/heap variable
  - Desde Windows Vista / 7 / 2008
  - CAMBIAN en cada reinicio
  - Activado por defecto (excepto Internet Explorer 7)
- Opción de compilación `/DYNAMICBASE` (VS2005 SP1 en adelante)
- Para ser efectivo, debe ser complementado con DEP
- ¿Cómo evitarla? (algunas técnicas)
  - Sólo randomiza los bytes altos de una dirección de memoria
  - Módulos que no tiene activado ASLR

Agenda

Conclusiones (I)

- Fallos en programación → puede derivar en BOFs explotables
- Mecanismos de protección
  - Flags del compilador (/GS, /SafeSEH, /NXCOMPAT, /DYNAMICBASE)
  - Sistema Operativo (SEHOP, Hardware-DEP)
  - Otras librerías comerciales/libres (`http://en.wikipedia.org/wiki/Buffer_overflow_protection`)
- Evasión de técnicas: conocidas y aplicables (normalmente)
  - Aisladas: dificultan el proceso
  - Combinadas: garantiza una mayor protección

Conclusiones (II)

Programa (y compila) de forma segura

Conclusiones (II)

Programa (y compila) de forma segura

Recomendaciones finales

- Uso de funciones seguras
- Compilación con todas las flags de protección habilitadas
- ¡Para todos los módulos y ejecutables de tus aplicaciones!

Agenda

Referencias (I)

- Corelan EWT, https://www.corelan.be/index.php/category/security/exploit-writing-tutorials/
- Wikipedia, http://en.wikipedia.org/wiki/Buffer_overflow
- CVE details, http://www.cvedetails.com

Referencias (I)

- Corelan EWT, https://www.corelan.be/index.php/category/security/exploit-writing-tutorials/
- Wikipedia, http://en.wikipedia.org/wiki/Buffer_overflow
- CVE details, http://www.cvedetails.com

- *Practical Malware Analysis*, M. Sikorski, A. Honig, NoStarch, 2012
- *Malware Analyst's Cookbook*, M.H. Ligh, S. Adair, B. Hartstein, M. Richard, Wiley, 2011
- *A Guide to Kernel Exploitation: Attacking the Core*, E. Perla, M. Oldani, Elsevier, 2011
- *Software Security: Building Security In*, G. McGraw, Addison Wesley, 2006
- *Reversing: Secrets of Reverse Engineering*, E. Eilam, Wiley, 2005
- *The Art of Computer Virus Research and Defense*, P. Szor, Addison Wesley, 2005

Buffer overflows: qué son y cómo evitarlos

Ricardo J. Rodríguez

All wrongs reversed

rjrodriguez@fi.upm.es ※ @RicardoJRdez ※ www.ricardojrodriguez.es

Universidad Politécnica de Madrid
Madrid, Spain

29 de Noviembre, 2013

BetaBeers

Zaragoza (España)