Fundamentos del Software: introducción a la programación verificada

Programación segura

Cadenas de formato

Roberto Blanco[†] & Ricardo J. Rodríguez[‡]

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Universidad Zaragoza

[†]Max Plank Institute for Security and Privacy
Bochum, Alemania

[‡]Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza (España)

Septiembre 2020

Universidad de Zaragoza

Índice

Universidad Zaragoza

Funciones con cadenas de formato

- **Cadena de formato + número variable de argumentos**
  - **La cadena de formato indica cómo se tienen que interpretar los argumentos por la función**
- Controlando el contenido de la cadena de formato, **un usuario puede controlar la salida formateada por la función**

Funciones con cadenas de formato

- **Cadena de formato + número variable de argumentos**
  - **La cadena de formato indica cómo se tienen que interpretar los argumentos por la función**
- Controlando el contenido de la cadena de formato, **un usuario puede controlar la salida formateada por la función**

Cadena de formato

- **Secuencia de caracteres y especificadores de conversión**
- **Los caracteres se copian sin más al flujo de salida**
- **Los especificadores de conversión indican cómo interpretar y escribir los argumentos al flujo de salida**
  - Empiezan con el carácter %, interpretándose de izquierda a derecha

Universidad Zaragoza

Funciones con cadenas de formato

Cadena de formato

```
%[flags] [width] [.precision] [{length-modifier}] conversion-specifier
```

- **Si hay más argumentos que especificadores de conversión, se ignoran**
- **Si hay menos, el resultado es indefinido**

Universidad Zaragoza

Funciones con cadenas de formato

Cadena de formato

Especificación de la conversión

- **Campos opcionales** (*flags*, *width*, *precision*, y *length modifier*)
- **Especificador de formato**

- **Ejemplo**: `%-10.8ld`
  - `-` es una flag
  - `10` es la anchura
  - `8` es la precisión
  - La letra `l` es el modificador de la longitud
  - `d` es el especificador de conversión: *imprime un argumento de tipo* `long int` *en notación decimal, con un mínimo de 8 dígitos alineados a la izquierda, considerando al menos 10 caracteres de anchura*

- **La especificación más simple contiene únicamente el carácter % y un especificador de conversión** (por ejemplo, %s)

Universidad Zaragoza

Funciones con cadenas de formato

Especificadores de conversión

Especificador	Salida	Ejemplo	Paso por...
`%d` o `i`	Entero decimal con signo	392	valor
`%u`	Entero decimal sin signo	7235	valor
`%o`	Entero octal (sin signo)	610	valor
`%x`	Entero hexadecimal (sin signo)	7fa	valor
`%X`	Entero hexadecimal (sin signo), en mayúsculas	7FA	valor
`%f`	Número real, en minúsculas	392.65	valor
`%F`	Número real, en mayúsculas	392.65	valor
`%e`	Notación científica (mantisa/exponente), en minúsculas	3.9265e+2	valor
`%E`	Notación científica (mantisa/exponente), en mayúsculas	3.9265E+2	valor
`%g`	Usa la notación más compacta: `%e` o `%f`	392.65	valor
`%G`	Usa la notación más compactan: `%E` o `%F`	392.65	valor
`%a`	Número real en hexadecimal, en minúsculas	-0xc.90fep-2	valor
`%A`	Número real en hexadecimal, en mayúsculas	-0XC.90FEP-2	valor
`%c`	Carácter	a	valor
`%s`	Cadena de caracteres	sample	referencia
`%p`	Dirección de memoria	b8000000	valor
`%n`	No imprime nada. El argumento correspondiente ha de ser un puntero a `signed int`. El número de caracteres que se ha escrito hasta el momento se guarda en el argumento.		referencia
`%%`	Para lograr escribir % en la salida.	%	–

Fuente: `http://www.cplusplus.com/reference/cstdio/printf/`

Universidad Zaragoza

Funciones con cadenas de formato

Elementos

- **Especificador de conversión**
  - Indica el tipo de conversión a aplicar
  - Único campo obligatorio, aparece después de los opcionales (siempre es el último)
- **Flags**
  - Justifica el texto de salida e imprime el signo, espacios en blanco, puntos decimales, prefijos octales/hexadecimales
  - Puede aparecer más de una
- **Width**
  - Mínimo de caracteres a escribir en la salida Specifies the minimum number of characters to output
  - Si el número a escribir es menor, se rellena con caracteres vacíos
  - Si la anchura es menor que el número, no pasa nada: no se trunca la salida
  - Si `width` es *, indica que el valor se coge de la lista de argumentos (tiene que preceder al valor que se quiere formatear)
- **Precision**
  - Especifica el número de caracteres (o decimales o dígitos significativos) que se van a imprimir
  - Puede truncar la salida (o redondearla)
  - Si `precision` es *, indica que el valor se coge de la lista de argumentos (tiene que preceder al valor que se quiere formatear)

Índice

Universidad Zaragoza

Vulnerabilidad de cadenas de formato

Comparativa con BOF

	Desbordamiento de buffer	**Cadenas de formato**
Conocido desde	mediados de 1980s	Junio 1999
Exploado desde	1990s	Junio 2000
Considerado como	Amenaza de seguridad	Error de programación
Técnicas de explotación	Avanzadas	Básicas
Visibilidad	Difícil (algunas veces)	Fácil

Adaptado de `https://crypto.stanford.edu/cs155/papers/formatstring-1.2.pdf`

Vulnerabilidad de cadenas de formato

Ejemplos reales

Aplicación	Encontrado por	Impacto	Duración
`wu-ftpd 2.*`	security.is	remote root	> 6
`Linux rpc.statd`	security.is	remote root	> 4
`IRIX telnetd`	LSD	remote root	> 8
`Qualcomm Popper 2.53`	security.is	remote user	> 3
`Apache + PHP3`	security.is	remote user	> 2
`NLS / locale`	CORE SDI	local root	?
`screen`	Jouko Pynn¨onen	local root	> 5
`BSD chpass`	TESO	local root	?
`OpenBSD fstat`	ktwo	local root	?

Adaptado de `https://crypto.stanford.edu/cs155/papers/formatstring-1.2.pdf`

Universidad Zaragoza

Vulnerabilidad de cadenas de formato

¿Cuándo aparece?

Cuando un usuario es capaz de proporcionar una cadena de formato (en parte o en su totalidad) a una función de formato

Vulnerabilidad de cadenas de formato

¿Cuándo aparece?

Cuando un usuario es capaz de proporcionar una cadena de formato (en parte o en su totalidad) a una función de formato

```
void error(char *s)
{
        fprintf(stderr, s);
}
```

¿Qué ocurre aquí si `s` es "%s %s %s %s %s %s"?

Vulnerabilidad de cadenas de formato

¿Cuándo aparece?

Cuando un usuario es capaz de proporcionar una cadena de formato (en parte o en su totalidad) a una función de formato

```
void error(char *s)
{
        fprintf(stderr, s);
}
```

¿Qué ocurre aquí si `s` es "%s %s %s %s %s %s"?

- **El programa acabará con error (muy probablemente)**: Denial-of-Service
- **Si no, se imprime el contenido de la memoria**: *problemas de privacidad*

Vulnerabilidad de cadenas de formato

Funciones vulnerables

- `fprintf`
- `printf`
- `sprintf`
- `snprintf`
- `vfprintf`
- `vprintf`
- `vsprintf`
- `vsnprintf`
- `setproctitle`
- `syslog`
- Otras como `err*`, `verr*`, `warn*`, `vwarn*`

Universidad Zaragoza

Vulnerabilidad de cadenas de formato

- **Funcionalidad**
  - Conversión a cadena de los tipos de datos simples de C
  - La representación del formato se puede especificar
  - La cadena resultado se procesa
- **¿Cómo funciona una función de cadenas de formato?**
  - **La cadena de formato controla el comportamiento de la función**
  - Tipos de parámetros que se van a imprimir
  - **Los parámetros se pasan por pila, directamente (paso por valor) o indirectamente (referencia)**
- **El llamante sabe cuántos parámetros se han metido en la pila, dado que tiene que ajustarla tras la ejecución de la función**

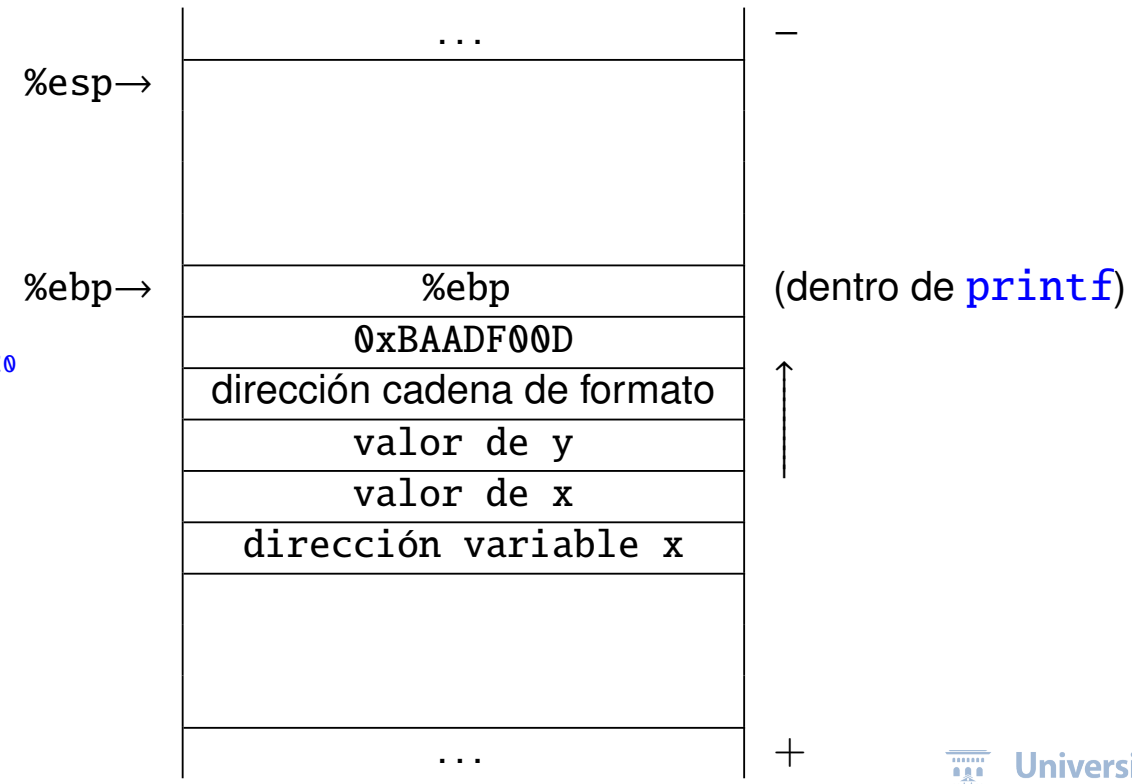
Universidad Zaragoza
1542

Vulnerabilidad de cadenas de formato

Ejemplo

```
printf("Values %d, %d, %08x\n", y, x, &x);
do_stuff(); // this call is in 0xBAADF00D address
```

```
_main:
        .....
        mov     eax, DWORD PTR [esp+24]
        lea     edx, [esp+24]
        mov     DWORD PTR [esp+12], edx
        mov     DWORD PTR [esp+8], eax
        mov     eax, DWORD PTR [esp+28]
        mov     DWORD PTR [esp+4], eax
        mov     DWORD PTR [esp], OFFSET FLAT:LC0
        call    _printf
0xBAADF00D:
        call    _do_stuff
        mov     eax, 0
        leave
        ret
```



Vulnerabilidad de cadenas de formato

Problema de canalización
Dos tipos de canales de información se mezclan en uno solo, y existen caracteres especiales que se usan para distinguir cuál es el canal que se encuentra activo actualmente

- Un canal es el **canal de datos** (no se parsea, sólo se copia)
- El otro canal es el **canal de control: especificadores de formato**

Universidad Zaragoza
1542

Vulnerabilidad de cadenas de formato

Posibles ataques

1. **Denegación de servicio**
   - **La lectura de direcciones de memoria no permitidas rompe la ejecución**

Vulnerabilidad de cadenas de formato

Posibles ataques

1. **Denegación de servicio**
   - **La lectura de direcciones de memoria no permitidas rompe la ejecución**
2. **Read-what-where**
   - Especificadores de formato de interés: `%s`, `%p`
     - **Se puede leer cualquier dirección de memoria válida**

```
printf("%s"); // will print the top of the stack
printf("%5$s"); // will print the 5th element of the stack
```

Vulnerabilidad de cadenas de formato

Posibles ataques

1. **Denegación de servicio**
   - **La lectura de direcciones de memoria no permitidas rompe la ejecución**
2. **Read-what-where**
   - Especificadores de formato de interés: `%s`, `%p`
     - **Se puede leer cualquier dirección de memoria válida**

```
printf("%s"); // will print the top of the stack
printf("%5$s"); // will print the 5th element of the stack
```

3. **Write-what-where**
   - Especificadores de formato de interés: `%n`
     - Escribe en el argumento **el número de bytes escritos hasta el momento**

```
int i;

printf("foobar%n", (int *)&i); // after, i = 6
```

Universidad Zaragoza

Vulnerabilidad de cadenas de formato

Algunos ejemplos: 2011 BMW 330i

Fuente: `https://twitter.com/__0bzy__/status/864704956116254720`

Índice

Universidad Zaragoza

Estrategias de mitigación

Cadenas de formato dinámicas

- **Diseña el código de modo que se seleccione una cadena de formato preexistente, en vez de permitir que la entrada del usuario entre a formar parte de la caena de formato directamente**

```
int x, y;
char format[256] = "%d * %d = ";

x = atoi(argv[1]);
y = atoi(argv[2]);
if (strcmp(argv[3], "hex") == 0){
      strcat(format, "0x%x\n");
}else{
      strcat(format, "%d\n");
}
printf(format, x, y, x * y);
```

Universidad Zaragoza

Estrategias de mitigación

Restricción del número de bytes escritos

- **Puede haber desbordamiento de buffers cuando hay cadenas; para evitarlos, se puede restringir el número de bytes a escribir**
- **En el caso de `%s`, el número de bytes a escribir se restringe mediante el campo de precisión**
  - `sprintf(buf, "Wrong command: %s\n", user);` ⇒
    `sprintf(buf, "Wrong command: %.495s\n", user);`
- **Sería mejor incluso usar la función `snprintf`:**

```
snprintf(buf, 512, "Wrong command: %s\n", user);
```

  - El campo precisión especifica el máximo número de bytes que se van a escribir para las conversiones de cadenas
  - Pueden usarse versiones más seguras, menos propensas a desbordamientos: `snprintf` mejor que `sprintf`; `vsnprintf` mejor que `vsprintf`
  - Especifican el máximo número de bytes que se van a escribir, incluyendo el byte nulo terminador

Universidad Zaragoza

Estrategias de mitigación

Anexo K C+11

- **Funciones `fprintf_s`, `printf_s`, `snprintf_s`, `sprintf_s`, `vfprintf_s`, `vprintf_s`, `vsnprintf_s`, `vsprintf_s`**
- Se diferencias de sus respectivas no-_s en que...
  - **No admiten el especificador `%n`**
  - Error en ejecución si los punteros son nulos o si la cadena de formato es inválida
- **Estas funciones pueden seguir usándose para *read-what-where* y para DoS**

Estrategias de mitigación

Avisos del compilador

Flags del compilador GNU C

- `-Wformat`
  - Incluido en `-Wall`
  - **Comprueba todas las llamadas a funciones de formato, examinando la cadena de formato y verificando el número y tipo de los argumentos dados**
    - **Cuidado**: no avisa si no hay coincidencias entre especificadores enteros con/sin signo y sus respectivos argumentos
- `-Wformat-nonliteral`
  - Hace lo mismo que `-Wformat`, pero **añade avisos si la cadena de formato no es un literal de cadena y no se puede comprobar**
- `-Wformat-security`
  - Hace lo mismo que `-Wformat`, pero **añade avisos sobre las funciones de cadenas de formato que pueden presentar problemas de seguridad**

Universidad Zaragoza

Fundamentos del Software: introducción a la programación verificada

Programación segura

Cadenas de formato

Roberto Blanco[†] & Ricardo J. Rodríguez[‡]

MAX PLANCK INSTITUTE
FOR SECURITY AND PRIVACY

Universidad Zaragoza

[†]Max Plank Institute for Security and Privacy
Bochum, Alemania

[‡]Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza (España)

Septiembre 2020

Universidad de Zaragoza