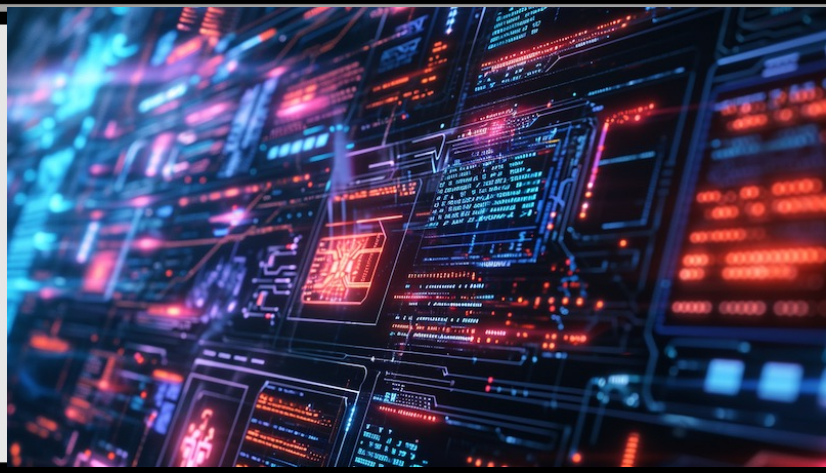
Class 7

CSC 583 Advanced Topics in Computer Security

CVE-2006-3439 Stack Overflow

Si Chen (schen@wcupa.edu)



“Memory Corruption”

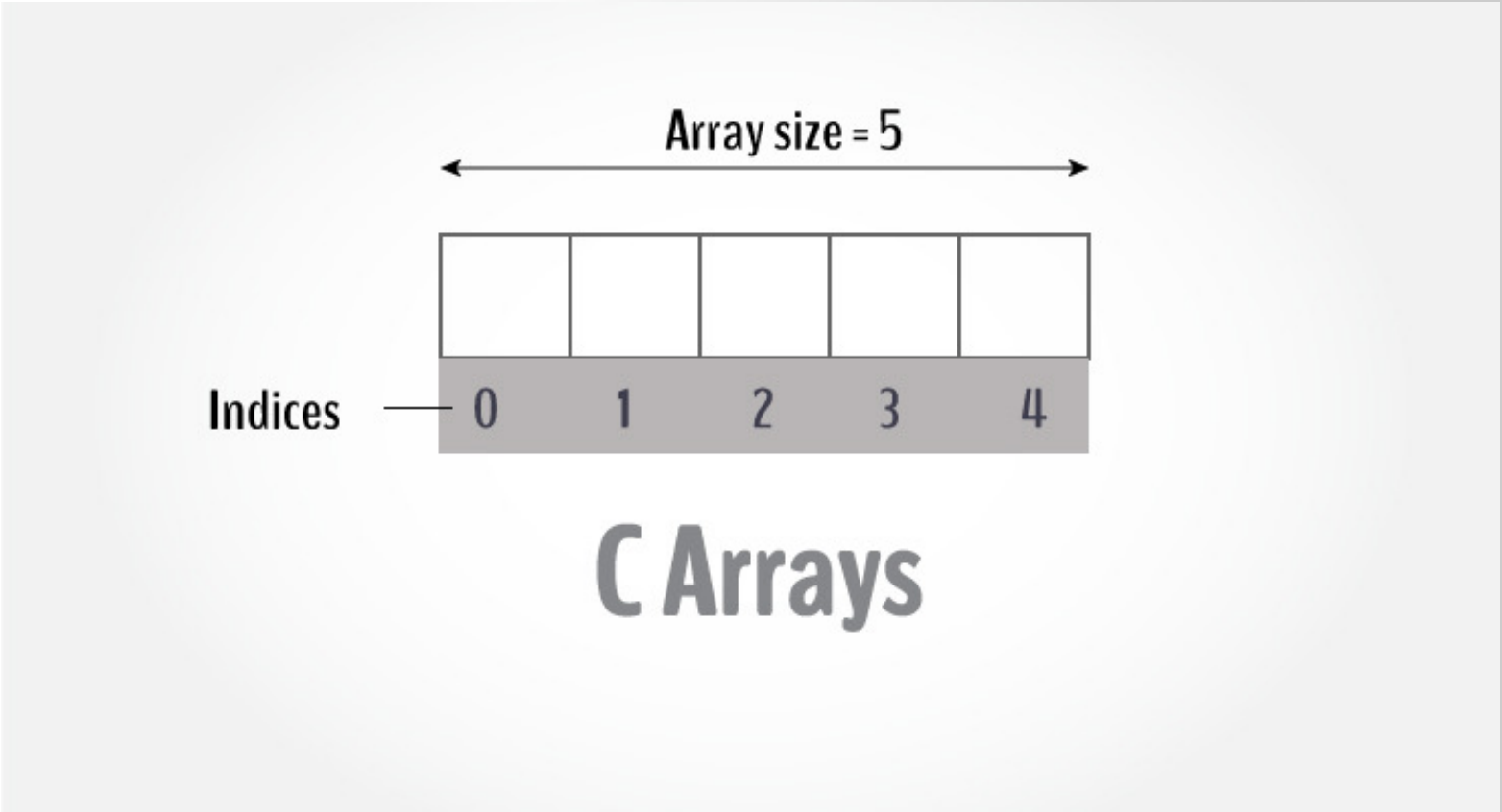
- What is it?

“Memory Corruption”

- Modifying a binary’s memory in a way that was not intended
- Broad umbrella term for most of what the rest of this class will be
- The vast majority of system-level exploits (real-world and competition) involve memory corruption

Buffers

- A buffer is defined as a limited, contiguously allocated set of memory. The most common buffer in C is an array.



A novice C programmer mistake

```c
#include <stdio.h>
#include <string.h>

int main()
{
    int array[5] = {1, 2, 3, 4, 5};
    printf("%d\n", array[5]);
}
```

```
quake0day@quakes-iMac  ~/Documents/Sync/CSC495 Software Security/ch5  cc buffer.c
buffer.c:7:17: warning: array index 5 is past the end of the array (which contains 5 elements) [-Warray-bounds]
        printf("%d\n", array[5]);
                       ^     ~
buffer.c:6:2: note: array 'array' declared here
        int array[5] = {1, 2, 3, 4, 5};
        ^
1 warning generated.
quake0day@quakes-iMac  ~/Documents/Sync/CSC495 Software Security/ch5  ./a.out
32767
```

This example shows how easy it is to read past the end of a buffer; C provides no built-in protection.

Another C programmer mistake

```c
#include <stdio.h>
#include <string.h>

int main()
{
    int array[5];
    int i;
    for(i = 0; i <= 255; i ++)
    {
        array[i] = 10;
    }
}
```

```
quake0day@quakes-iMac  ~/Documents/Sync/CSC495_Software_Security/ch5  cc buffer2.c
quake0day@quakes-iMac  ~/Documents/Sync/CSC495_Software_Security/ch5  ./a.out
[1]    26905 abort      ./a.out
```

Crash report

Console

Now | Activities | Clear | Reload | Info

All Messages | Errors and Faults

Devices
- quake's iMac

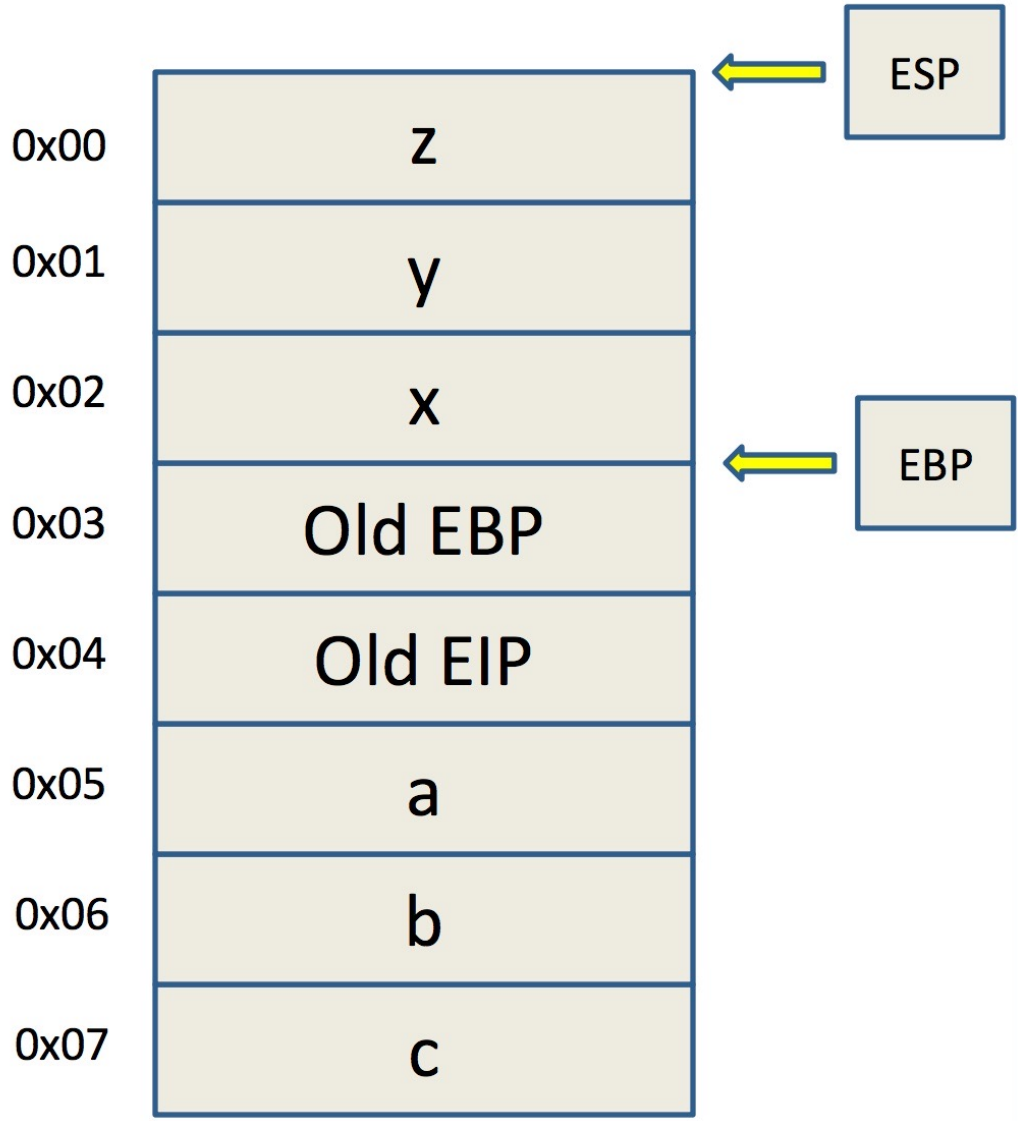
Reports
- Mac Analytics...
- System Reports
- User Reports
- system.log
- ~/Library/Logs
- /Library/Logs
- /var/log

- a.out_2017-0...es-iMac.crash
- a.out_2017-0...es-iMac.crash
- a.out_2017-0...es-iMac.crash
- CEPHtmlEngi...es-iMac.crash
- random_num_...s-iMac.crash
- random_num_...s-iMac.crash

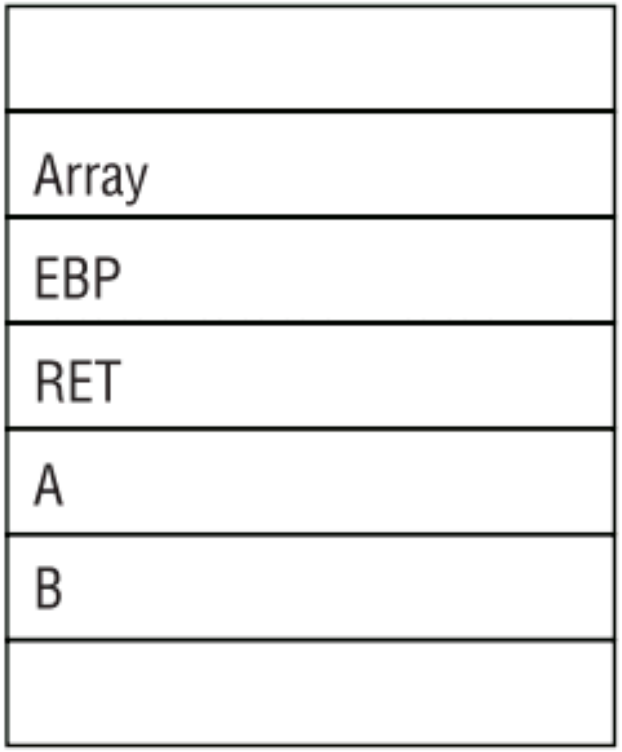
```
Process:               a.out [26905]
Path:                  /Users/USER/Documents/*/a.out
Identifier:            a.out
Version:               0
Code Type:             X86-64 (Native)
Parent Process:        zsh [25194]
Responsible:           a.out [26905]
User ID:               501

Date/Time:             2017-09-12 12:14:39.138 -0400
OS Version:            Mac OS X 10.12.6 (16G29)
Report Version:        12
Anonymous UUID:        FA3D3E94-9D60-E763-CD6E-C784AEA99094


Time Awake Since Boot: 160000 seconds

System Integrity Protection: enabled

Crashed Thread:        0  Dispatch queue: com.apple.main-thread

Exception Type:        EXC_CRASH (SIGABRT)
Exception Codes:       0x0000000000000000, 0x0000000000000000
Exception Note:        EXC_CORPSE_NOTIFY

Application Specific Information:
[26905] stack overflow

Thread 0 Crashed:: Dispatch queue: com.apple.main-thread
0   libsystem_kernel.dylib          0x00007fff92cbfd42 __pthread_kill + 10
1   libsystem_pthread.dylib         0x00007fff92dad457 pthread_kill + 90
2   libsystem_c.dylib               0x00007fff92c254bb __abort + 140
3   libsystem_c.dylib               0x00007fff92c25d7e __stack_chk_fail + 205
4   a.out                           0x00000001003baf96 main + 118
5   ???                             0x0000000a0000000a 0 + 42949672970

Thread 0 crashed with X86 Thread State (64-bit):
  rax: 0x0000000000000000  rbx: 0x0000000000000006  rcx: 0x00007fff5f8458f8  rdx: 0x0000000000000000
  rdi: 0x0000000000000307  rsi: 0x0000000000000006  rbp: 0x00007fff5f845920  rsp: 0x00007fff5f8458f8
   r8: 0x0000000000000000   r9: 0x0000000000000000  r10: 0x0000000008000000  r11: 0x0000000000000206
  r12: 0x0000000000000000  r13: 0x0000000000000000  r14: 0x00007fff9baa43c0  r15: 0x0000000000000000
  rip: 0x00007fff92cbfd42  rfl: 0x0000000000000206  cr2: 0x00007fff92dab31b

Logical CPU:     0
Error Code:      0x02000148
Trap Number:     133


Binary Images:
       0x1003ba000 -        0x1003bafff +a.out (0) <F20F94F3-B9D2-3630-B259-E53B6ED98692> /Users/USER/Documents/*/a.out
       0x106f10000 -        0x106f4ddc7  dyld (433.5) <322C06B7-8878-311D-888C-C8FD2CA96FF3> /usr/lib/dyld
    0x7fff915e7000 -     0x7fff915e8ffb  libSystem.B.dylib (1238.60.2) <F18AC1E7-C6F1-34B1-8069-BE571B3231D4> /usr/lib/libSystem.B.dylib
    0x7fff91721000 -     0x7fff91777ff7  libc++.1.dylib (307.5) <0B43BB5D-E6EB-3464-8DE9-B41AC8ED9D1C> /usr/lib/libc++.1.dylib
    0x7fff91778000 -     0x7fff917a1ff7  libc++abi.dylib (307.4) <BC271AD3-831B-362A-9DA7-E8C51F285FE4> /usr/lib/libc++abi.dylib
    0x7fff92296000 -     0x7fff92668047  libobjc.A.dylib (709.1) <70614861-0340-32E2-85ED-FE65759CDFFA> /usr/lib/libobjc.A.dylib
    0x7fff92ab5000 -     0x7fff92ab9ff7  libcache.dylib (79) <093A4DAB-8385-3D47-A350-E20CB7CCF7BF> /usr/lib/system/libcache.dylib
    0x7fff92aba000 -     0x7fff92ac4fff  libcommonCrypto.dylib (60092.50.5) <BA64D1B0-C70E-385C-92F0-E669079FDA90> /usr/lib/system/libcommonCrypto.dylib
    0x7fff92ac5000 -     0x7fff92accfff  libcompiler_rt.dylib (62) <55DA7421-772A-32AB-B529-1A46C2F43B4D> /usr/lib/system/libcompiler_rt.dylib
    0x7fff92acd000 -     0x7fff92ad5fff  libcopyfile.dylib (138) <8199BEA3C-DF11-3E3D-A1A1-5A51C5BF1961> /usr/lib/system/libcopyfile.dylib
    0x7fff92ad6000 -     0x7fff92b59fdf  libcorecrypto.dylib (442.50.19) <65D7165E-2E71-335D-A2D6-33F7BE2DF0C1> /usr/lib/system/libcorecrypto.dylib
    0x7fff92b5a000 -     0x7fff92b8bfff  libdispatch.dylib (703.50.37) <6582BAD6-ED27-3B30-B620-90B1C5A4AE3C> /usr/lib/system/libdispatch.dylib
    0x7fff92b8c000 -     0x7fff92b91ffb  libdyld.dylib (433.5) <9B2AC56D-107C-3541-A127-9094A751F2C9> /usr/lib/system/libdyld.dylib
    0x7fff92b92000 -     0x7fff92b92ffb  libkeymgr.dylib (28) <7AA011A9-DC21-3488-BF73-3B5B14D1FDD6> /usr/lib/system/libkeymgr.dylib
    0x7fff92ba0000 -     0x7fff92ba0fff  liblaunch.dylib (972.70.1) <B856ABD2-896E-3DE0-B2C8-146A6AF8E2A7> /usr/lib/system/liblaunch.dylib
    0x7fff92ba1000 -     0x7fff92ba6ff3  libmacho.dylib (898) <17D5D855-F6C3-3B04-B680-E9BF02EF8AED> /usr/lib/system/libmacho.dylib
    0x7fff92ba7000 -     0x7fff92ba9ff3  libquarantine.dylib (85.50.1) <12448CC2-378E-35F3-BE33-9DC395A5B970> /usr/lib/system/libquarantine.dylib
    0x7fff92baa000 -     0x7fff92babffb  libremovefile.dylib (45) <38D4CB9C-10CD-30D3-8B7B-A515EC75FE85> /usr/lib/system/libremovefile.dylib
    0x7fff92bac000 -     0x7fff92bc4ff7  libsystem_asl.dylib (349.50.5) <896E4228-3B7C-30A6-8B13-EC989A644994A> /usr/lib/system/libsystem_asl.dylib
    0x7fff92bc5000 -     0x7fff92bc5ff7  libsystem_blocks.dylib (67) <10DC5404-73AB-35B3-A277-A8AFECB476EB> /usr/lib/system/libsystem_blocks.dylib
    0x7fff92bc6000 -     0x7fff92c53fef  libsystem_c.dylib (1158.50.2) <E5AE5244-7D0C-36AC-8BB6-C7AE7EA52A4B> /usr/lib/system/libsystem_c.dylib
    0x7fff92c54000 -     0x7fff92c57ffb  libsystem_configuration.dylib (888.60.2) <BECC01A2-CA8D-31E6-BCDF-D452965FA976> /usr/lib/system/libsystem_configuration.dylib
    0x7fff92c58000 -     0x7fff92c5bfff  libsystem_coreservices.dylib (41.4) <7D26DE79-B424-3450-85E1-F7FAB32714AB> /usr/lib/system/libsystem_coreservices.dylib
    0x7fff92c5c000 -     0x7fff92c74fff  libsystem_coretls.dylib (121.50.4) <EC6FCF07-DCFB-3A03-9CC9-6DD3709974C6> /usr/lib/system/libsystem_coretls.dylib
    0x7fff92c75000 -     0x7fff92c7bfff  libsystem_dnssd.dylib (765.50.9) <CC960215-0B1B-3B22-A13A-3DDE96FA796F> /usr/lib/system/libsystem_dnssd.dylib
    0x7fff92c7c000 -     0x7fff92ca5ff7  libsystem_info.dylib (503.50.4) <611D884C-BF70-3F92-8702-B9F28A900920> /usr/lib/system/libsystem_info.dylib
    0x7fff92ca6000 -     0x7fff92cc8ff7  libsystem_kernel.dylib (3789.70.16) <34B1F16C-BC9C-3C5F-9045-0CAE91CB5914> /usr/lib/system/libsystem_kernel.dylib
    0x7fff92cc9000 -     0x7fff92d10fe7  libsystem_m.dylib (3121.6) <86D49985-BBDC-3D3B-8A4E-97AE8E6672A4> /usr/lib/system/libsystem_m.dylib
    0x7fff92d11000 -     0x7fff92d2fff7  libsystem_malloc.dylib (116.50.8) <A3D15F17-99A6-3367-8C7E-4280E8619C95> /usr/lib/system/libsystem_malloc.dylib
    0x7fff92d30000 -     0x7fff92d89ffb  libsystem_network.dylib (856.60.1) <369D0221-56CA-3C3E-9EDE-94B41CAE77B7> /usr/lib/system/libsystem_network.dylib
    0x7fff92d8a000 -     0x7fff92d93ff3  libsystem_networkextension.dylib (563.60.2) <B021F2B3-8A75-3633-ABB0-FC012B8E9B0C> /usr/lib/system/libsystem_networkextension.dylib
    0x7fff92d94000 -     0x7fff92d9dff3  libsystem_notify.dylib (165.20.1) <BB160190-A069-3B3A-BDF6-2AA408221FAE> /usr/lib/system/libsystem_notify.dylib
    0x7fff92d9e000 -     0x7fff92da6fe7  libsystem_platform.dylib (126.50.8) <897462FD-B318-321B-A554-E61982630F7E> /usr/lib/system/libsystem_platform.dylib
    0x7fff92da7000 -     0x7fff92db1ff7  libsystem_pthread.dylib (218.60.3) <B8FB5E20-3295-39E2-B5EB-A464D1D4B104> /usr/lib/system/libsystem_pthread.dylib
    0x7fff92db2000 -     0x7fff92db5ff7  libsystem_sandbox.dylib (592.70.1) <4B92EC49-ACD0-36AE-B07A-A2B8152EAF9D> /usr/lib/system/libsystem_sandbox.dylib
    0x7fff92db6000 -     0x7fff92db7ff3  libsystem_secinit.dylib (24.50.4) <F7BB8479-3565-3E4B-98A6-F7AD40392E2D> /usr/lib/system/libsystem_secinit.dylib
    0x7fff92db8000 -     0x7fff92dbfffb  libsystem_symptoms.dylib (532.50.47) <3390E07C-C1CE-348F-ADBD-2C5440B45EAA> /usr/lib/system/libsystem_symptoms.dylib
    0x7fff92dc0000 -     0x7fff92dd3ff7  libsystem_trace.dylib (518.70.1) <AC63A7FE-50D9-3A30-96E6-F6B7FF16E465> /usr/lib/system/libsystem_trace.dylib
    0x7fff92dd4000 -     0x7fff92dd9ffb  libunwind.dylib (35.3) <3D50D8A8-C460-334D-A519-2DA841102C6B> /usr/lib/system/libunwind.dylib
    0x7fff92dda000 -     0x7fff92e03ff7  libxpc.dylib (972.70.1) <BF896DF0-D8E9-31A8-A4B3-01120BFEEE52> /usr/lib/system/libxpc.dylib

External Modification Summary:
  Calls made by other processes targeting this process:
    task_for_pid: 0
    thread_create: 0
    thread_set_state: 0
  Calls made by this process:
    task_for_pid: 0
```

Stack

```
int foo(int a, int b, int c)
{
    int x;
    int y;
    int z;

    x=y=z=0;
    z=x+y+a+b+c;
    return z;


}
int main(int argc, char **argv) {


        foo(1,2,3);
}
```



Stack Frame



Low Memory Addresses and Top of the Stack

High Memory Addresses and Bottom of the Stack

Overflow.c

```c
#include <stdio.h>
#include <string.h>

void hacked()
{
    puts("Hacked by Si Chen!!!!");
}

void return_input(void)
{
    char array[30];
    gets(array);
    printf("%s\n", array);
}

main()
{
    return_input();
    return 0;
}
```

```
[quake0day@quake0day-w
AAAAAAAAAA
AAAAAAAAAA
```

```
File  Edit  View  Terminal
[quake0day@quake0da
r -zexecstack
overflow.c: In func
overflow.c:7:2: war
fgets'? [-Wimplicit
  gets(array);
  ^~~~
  fgets
overflow.c: At top
overflow.c:11:1: wa
 main()
 ^~~~
/tmp/cclk5rGr.o: In
overflow.c:(.text+0
t be used.
[quake0day@quake0da
```

Overflow.c

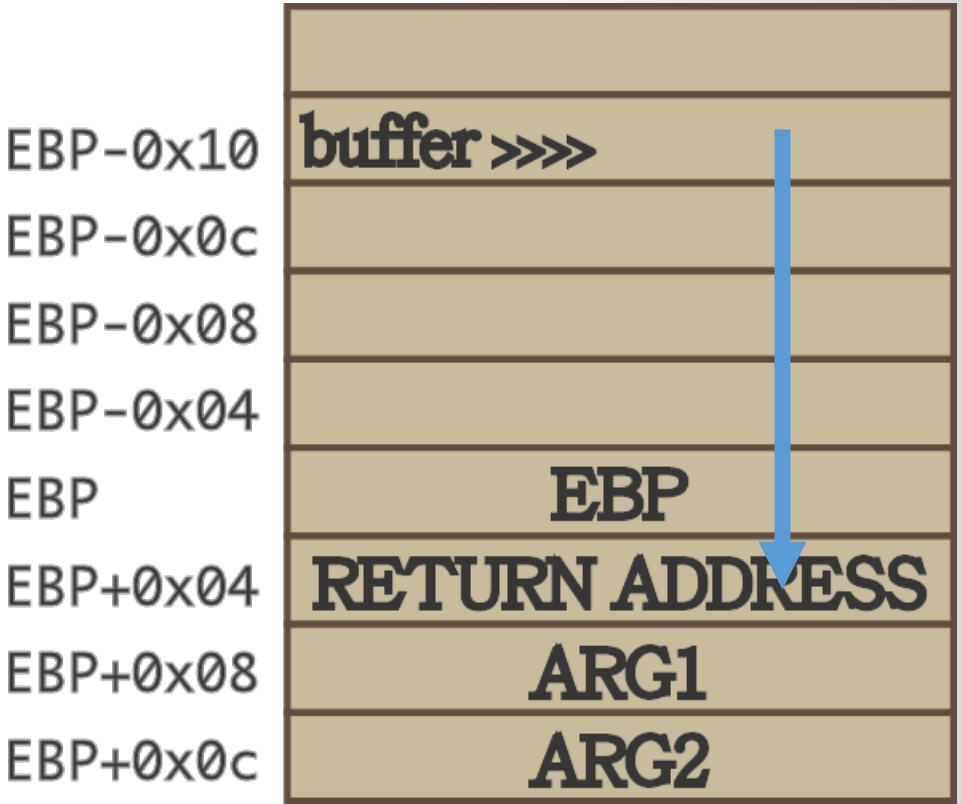
```c
#include <stdio.h>
#include <string.h>

void hacked()
{
    puts("Hacked by Si Chen!!!!");
}

void return_input(void)
{
    char array[30];
    gets(array);
    printf("%s\n", array);
}

main()
{
    return_input();
    return 0;
}
```

```
[quake0day@quake0day-wcu ~]$ ./overflow
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDD
AAAAAAAAAABBBBBBBBBBCCCCCCCCCCDDDDDDDDDD
*** stack smashing detected ***: ./overflow terminated
Segmentation fault (core dumped)
```

```
[quake0day@quake0day-wcu ~]$ ./overflow
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
*** stack smashing detected ***: ./overflow terminated
======= Backtrace: =========
/usr/lib/libc.so.6(+0x6a1e0)[0xb7e5b1e0]
/usr/lib/libc.so.6(__fortify_fail+0x38)[0xb7eefa38]
/usr/lib/libc.so.6(+0xfe9f8)[0xb7eef9f8]
./overflow(+0x6a3)[0x4006a3]
./overflow(+0x5f4)[0x4005f4]
./overflow(main+0x12)[0x40060b]
/usr/lib/libc.so.6(__libc_start_main+0xf3)[0xb7e091d3]
./overflow(+0x4a1)[0x4004a1]
======= Memory map: ========
00400000-00401000 r-xp 00000000 08:01 318658     /home/quake0day/overflow
00401000-00402000 r--p 00000000 08:01 318658     /home/quake0day/overflow
00402000-00403000 rw-p 00001000 08:01 318658     /home/quake0day/overflow
00403000-00424000 rw-p 00000000 00:00 0          [heap]
```

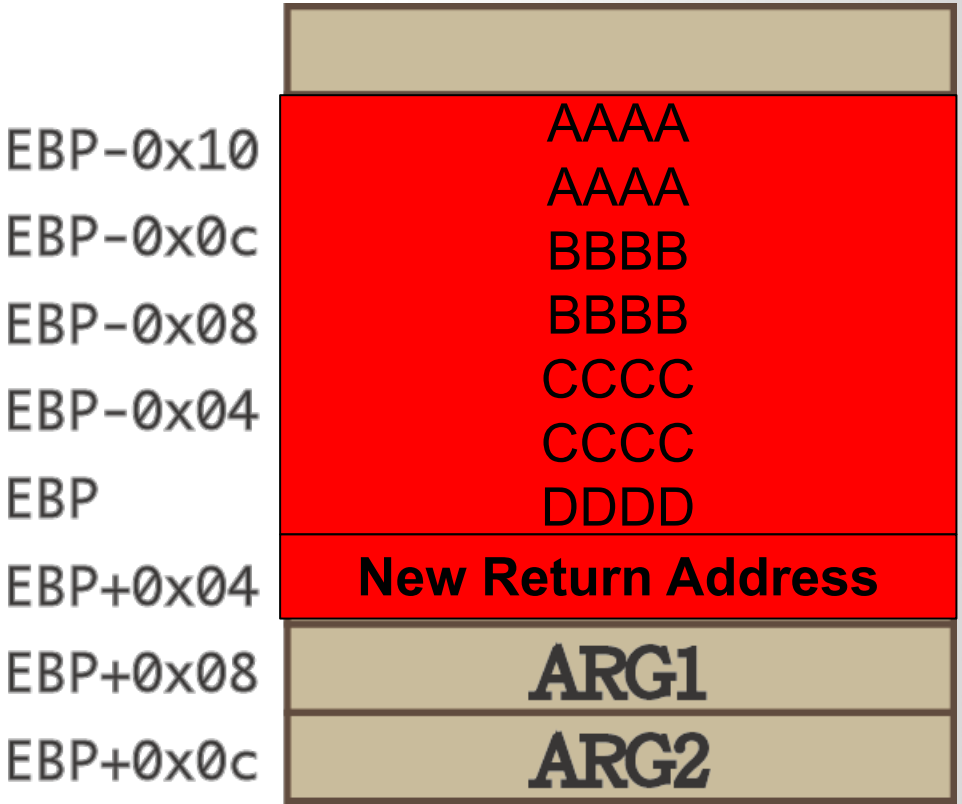
Return Hijack

- The return address will be stored on stack when calling a new function. (EIP)
- The local valuable will be store on the low address
- If the variable is an array, and if we store too many data, it will cover the return address which store on the high address.



From Crash to Hack

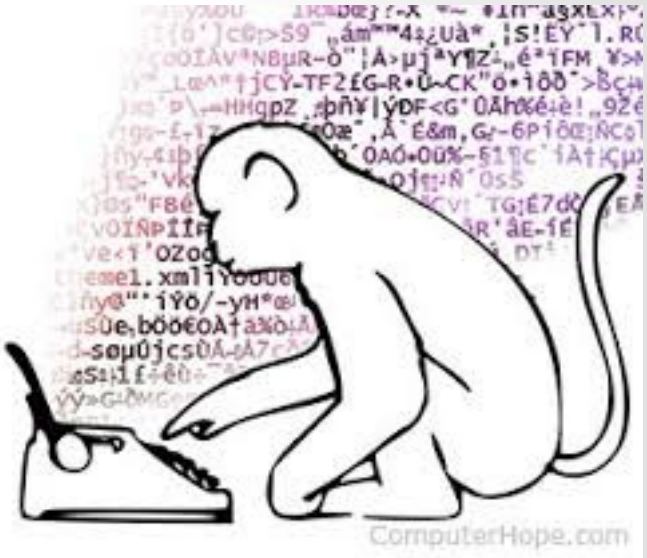
- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
  - The general idea is to overflow a buffer so that it overwrites the return address.



Guessing Addresses

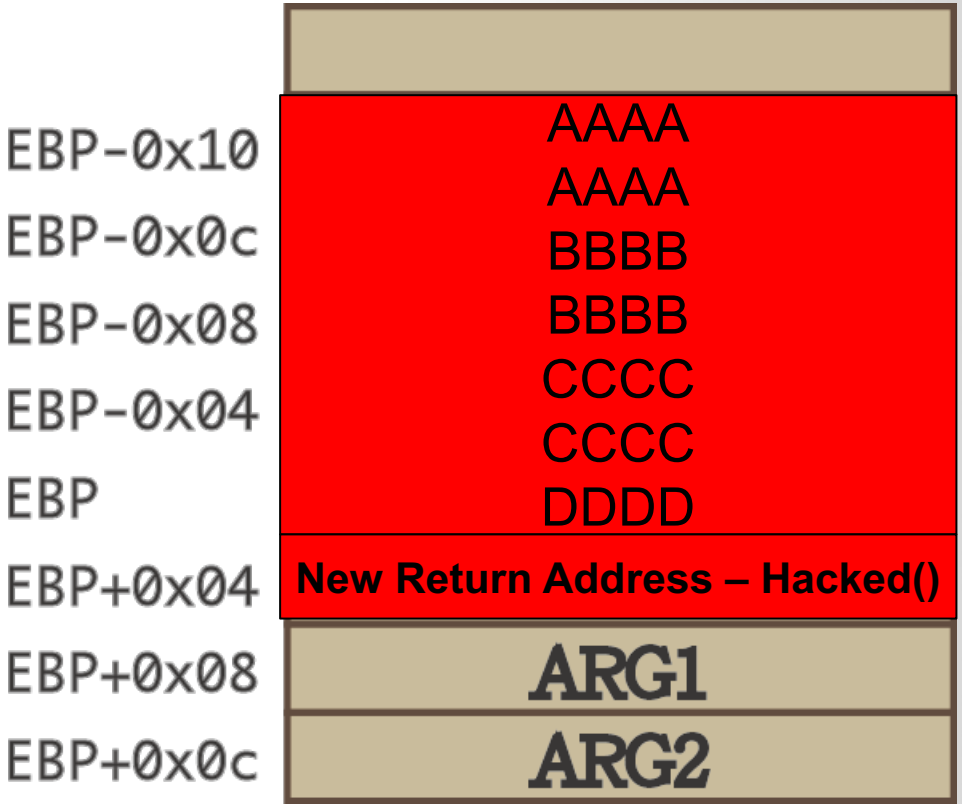
- Typically you need the source code so you can *estimate* the address of both the buffer and the return-address.

- An estimate is often good enough! (more on this in a bit).



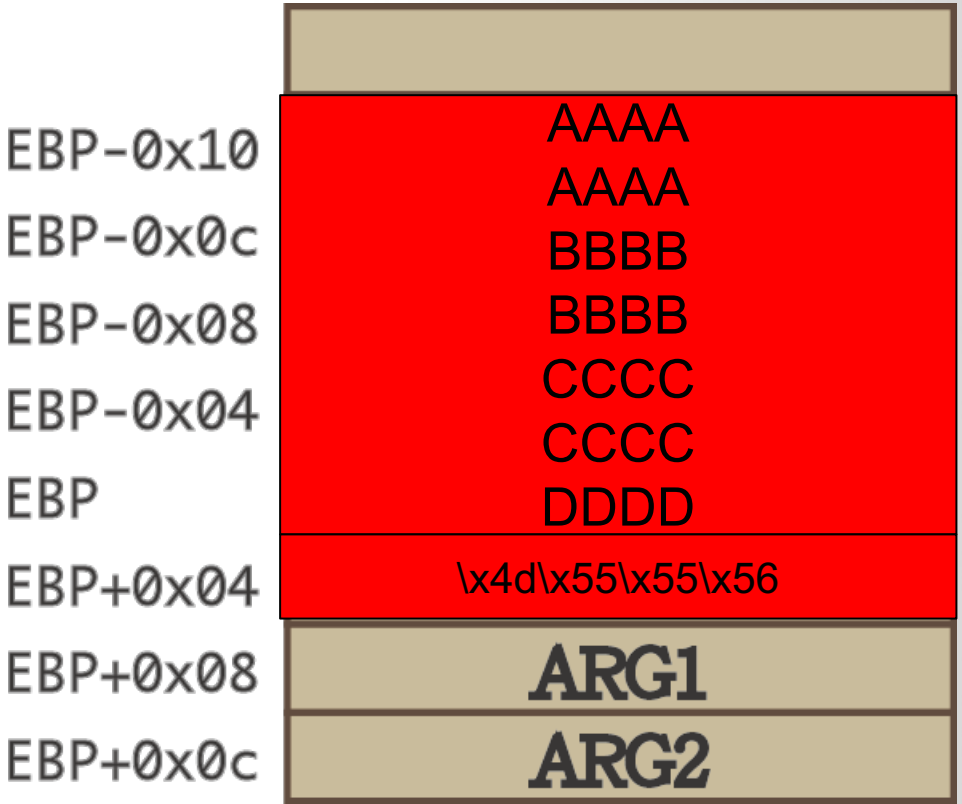
From Crash to Hack

- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
  - The general idea is to overflow a buffer so that it overwrites the return address.



From Crash to Hack

- If the input is larger than the size of the array, normally, the program will crash.
- Need to craft special data to exploit this vulnerability.
  - The general idea is to overflow a buffer so that it overwrites the return address.



Common vulnerabilities and exposures (CVE)

- **CVE system:**
  - Standardized way of identifying and categorizing security vulnerabilities and exposures
  - Maintained by The MITRE Corporation
  - Funded by the US Department of Homeland Security
  - Launched in September 1999
  - Used by the Security Content Automation Protocol
  - CVE IDs listed on Mitre's system and the US National Vulnerability Database

Why CVE-2006-3439?

We should often review past vulnerabilities, especially classic ones, and review the ideas behind their vulnerability analysis, discovery, and exploitation. This will often rekindle sparks of creativity in our thinking and inspire us when researching new problems.

-- a Hacker

Background: CVE-2006-3439

Link: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2006-3439

Printer-Friendly View

CVE-ID	
CVE-2006-3439	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • (Information
Description	
Buffer overflow in the Server Service in Microsoft Windows 2000 SP4, XP SP1 and SP2, and Server 2003 SP1 allows remote attackers, including anonymous users, to execute arbitrary code via a crafted RPC message, a different vulnerability than CVE-2006-1314.	

Microsoft Security Bulletin MS06-040 - Critical

https://learn.microsoft.com/en-us/security-updates/Securitybulletins/2006/ms06-040?redirectedfrom=MSDN

CVE-2006-3439 Static Analysis

Use IDA to load NETAPI32.dll

target: sub_7517FC68

Library function | Regular function | Instruction | Data | Unexplored | External symbol | Lumina function

Functions | IDA View-A | Hex View-1 | Structures | Enums | Imports | Exports

Function name

- NetpwNameValidate
- sub_7517ECDD
- sub_7517ECFD
- I_NetServerSetServiceBits
- **NetRemoteComputerSupports**
- I_NetPathType
- I_NetNameValidate
- sub_7517F136
- NetpwPathType
- NetpwNameCanonicalize
- sub_7517F6EA
- **NetUseEnum**
- NetpwPathCanonicalize
- sub_7517F870
- I_NetNameCanonicalize
- **NetUseGetInfo**
- I_NetPathCanonicalize
- sub_7517FB4E
- **NetUseAdd**
- sub_7517FC68
- sub_7517FD81
- sub_7517FDA1
- sub_7517FE34
- NetpwNameCompare
- NetpLogonPutUnicodeString
- sub_7518014B

Line 46 of 1166

Graph overview

```
arg_C= dword ptr  14h
arg_10= dword ptr  18h

push    ebp
mov     ebp, esp
sub     esp, 414h
push    ebx
push    esi
xor     esi, esi
push    edi
cmp     [ebp+String], esi
mov     edi, ds:__imp_wcslen
mov     ebx, 411h
jz      short loc_7517FCED

push    [ebp+String]    ; String
call    edi ; __imp_wcslen
mov     esi, eax
pop     ecx
test    esi, esi
jz      short loc_7517FCF4

cmp     esi, ebx
ja      loc_7517FD3E

push    [ebp+String]    ; Source
lea     eax, [ebp+Destination]
push    eax             ; Destination
call    ds:__imp_wcscpy
mov     ax, [ebp+esi*2+var_416]
pop     ecx
cmp     ax, 5Ch ; '\'
pop     ecx
jz      short loc_7517FCD5

cmp     ax, 2Fh ; '/'
jz      short loc_7517FCD5

lea     eax, [ebp+Destination]
push    offset asc_751717B8 ; "\\"
push    eax             ; Destination
```

100.00% (-773,371) (2,1128) 0000F268 7517FC68: sub_7517FC68 (Synchronized with Hex View-1)

Output

Try to press F5

CVE-2006-3439 Static Analysis

```
; Attributes: bp-based frame

; int __stdcall sub_7517FC68(wchar_t *String, wchar_t *Source, wchar_t *, unsigned int, size_t *)
sub_7517FC68 proc near

var_416= word ptr -416h
Destination= word ptr -414h
String= dword ptr  8
Source= dword ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h
arg_10= dword ptr  18h
```

negative value → local variable

CVE-2006-3439 Static Analysis

```
; Attributes: bp-based frame

; int __stdcall sub_7517FC68(wchar_t *String, wchar_t *Source, wchar_t *, unsigned int, size_t *)
sub_7517FC68 proc near

var_416= word ptr -416h
Destination= word ptr -414h
String= dword ptr  8
Source= dword ptr  0Ch
arg_8= dword ptr  10h
arg_C= dword ptr  14h
arg_10= dword ptr  18h

push    ebp
mov     ebp, esp
sub     esp, 414h
push    ebx
push    esi
xor     esi, esi
push    edi
cmp     [ebp+String], esi
mov     edi, ds:__imp_wcslen
mov     ebx, 411h
jz      short loc_7517FCED
```

CVE-2006-3439 Static Analysis

```
push    [ebp+String]    ; String
call    edi ; __imp_wcslen
mov     esi, eax
pop     ecx
test    esi, esi
jz      short loc_7517FCF4
```

CVE-2006-3439 Static Analysis

```
cmp     esi, ebx
ja      loc_7517FD3E
```

CVE-2006-3439 Static Analysis

```
push    [ebp+String]      ; Source
lea     eax, [ebp+Destination]
push    eax               ; Destination
call    ds:__imp_wcscpy
mov     ax, [ebp+esi*2+var_416]
pop     ecx
cmp     ax, 5Ch ; '\'
pop     ecx
jz      short loc_7517FCD5
```

Exercise Questions

1. How can one identify the number of local variables and parameters within the current function?
2. How can one determine the size of the buffer space allocated by the current function?
3. What is the result of XORing a value with itself?
4. In which register is the return value of a function call generally stored?
5. What is the underlying principle of the comparison operation in cmp?
6. What do the jump instructions like jz, ja, etc., signify?
7. What are the rules for pushing function arguments onto the stack?

Let's pause here and take a moment to do two things. First, let's review the small pieces of knowledge that we've covered up to this point. I've summarized them below for your reference:

1. How to identify the number of local variables and parameters in a current function.
2. How to view the size of the buffer space allocated for a current function.
3. What result is obtained by XORing a value with itself.
4. Where the return value of a function call is typically stored.
5. The principle behind the comparison operation "cmp".
6. What the "jz", "ja", and other jump instructions represent.
7. The rules for how function arguments are pushed onto the stack.

wcslen, wcsnlen_s

Defined in header <wchar.h>

`size_t wcslen( const wchar_t *str );`	(1)	(since C95)
`size_t wcsnlen_s(const wchar_t *str, size_t strsz);`	(2)	(since C11)

1) Returns the length of a wide string, that is the number of non-null wide characters that precede the terminating null wide character.

2) Same as (1), except that the function returns zero if `str` is a null pointer and returns `strsz` if the null wide character was not found in the first `strsz` wide characters of `src`

As with all bounds-checked functions, `wcslen_s` only guaranteed to be available if `__STDC_LIB_EXT1__` is defined by the implementation and if the user defines `__STDC_WANT_LIB_EXT1__` to the integer constant `1` before including `<stdio.h>`..

Parameters

`str` - pointer to the null-terminated wide string to be examined

`strsz` - maximum number of wide characters to examine

Return value

1) The length of the null-terminated wide string `str`.

2) The length of the null-terminated wide string `str` on success, zero if `str` is a null pointer, `strsz` if the null wide character was not found.

Wide character

- A wide character is a computer character datatype that generally has a size greater than the traditional **8-bit** character. The increased datatype size allows for the use of larger coded character sets.
- Early adoption of UCS-2 ("Unicode 1.0") led to common use of UTF-16 in a number of platforms, most notably Microsoft Windows, .NET and Java. In these systems, it is common to have a "wide character" (wchar_t in C/C++; char in Java) type of **16-bits**. These types do not always map directly to one "character", as surrogate pairs are required to store the full range of Unicode (1996, Unicode 2.0).

```
push    [ebp+String]     ; String
call    edi ; __imp_wcslen
mov     esi, eax
pop     ecx
test    esi, esi
jz      short loc_7517FCF4
```

https://en.wikipedia.org/wiki/Wide_character

Check Cross-references

```
; Attributes: bp-based frame

; int __stdcall sub_7517FC68(wchar_t *String, wchar_t *Source, wchar_t *, unsign
sub_7517FC68 proc near

var_416= word ptr -416h
Destination= word ptr -414h
```

xrefs to sub_7517FC68

Direction	Typ	Address	Text
Up	p	NetpwPathCanonicalize+...	call sub_7517FC68

Line 1 of 1

Help Search Cancel OK

```
push    ebx
push    esi
```

```
loc_7517F83B:
cmp     [ebp+arg_8], edi
jnz     short loc_7517F847
```

```
loc_7517F847:
mov     esi, [ebp+arg_4]
push    edi                 ; int
push    [ebp+arg_8]         ; int
mov     [esi], di
push    esi                 ; wchar_t *
push    [ebp+Source]        ; Source
push    ebx                 ; String
call    sub_7517FC68
cmp     eax, edi
jnz     short loc_7517F869
```

```
push    57h ; 'W'
pop     eax
jmp     short loc_7517F869
```

```
mov     eax, 84Bh
jmp     short loc_7517F869
```

```
push    edi                 ; int
push    [ebp+arg_10]        ; int
push    esi                 ; String
call    NetpwPathType
```

```
lea     eax, [ebp+arg_14]
push    edi                 ; int
push    eax                 ; int
push    ebx                 ; String
call    NetpwPathType
cmp     eax, edi
jnz     short loc_7517F869
```

```
loc_7517F83B:
cmp     [ebp+arg_8], edi
jnz     short loc_7517F847
```

```
loc_7517F847:
mov     esi, [ebp+arg_4]
push    edi                 ; int
push    [ebp+arg_8]         ; int
mov     [esi], di
push    esi                 ; wchar_t *
push    [ebp+Source]        ; Source
push    ebx                 ; String
call    sub_7517FC68
```

NetpwPathType()

```
; Exported entry 305. NetpwPathType


; Attributes: bp-based frame

; int __stdcall NetpwPathType(wchar_t *String, int, int)
public NetpwPathType
NetpwPathType proc near

var_14= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
String= dword ptr  8
arg_4= dword ptr  0Ch
arg_8= dword ptr  10h

push    ebp
mov     ebp, esp
sub     esp, 14h
push    esi
push    edi
mov     edi, [ebp+arg_4]
and     dword ptr [edi], 0
test    [ebp+arg_8], 7FFFFFFEh
jz      short loc_7517F4E4
```

```
loc_7517F4E4:
mov     esi, [ebp+String]
test    esi, esi
jz      short loc_7517F4F5
```

```
push    esi                     ; String
call    ds:__imp_wcslen
pop     ecx
jmp     short loc_7517F4F7
```

```
loc_7517F4F5:
xor     eax, eax
```

```
loc_7517F4F7:
test    eax, eax
jz      short loc_7517F54F
```

```
loc_7517F4F7:
test    eax, eax
jz      short loc_7517F54F
```

```
cmp     eax, 103h
ja      short loc_7517F54F
```

```
xor     eax, eax
mov     [ebp+var_14], edi
test    byte ptr [ebp+arg_8], 1
mov     [ebp+var_4], eax
jz      short loc_7517F516
```

```
lea     eax, [ebp+arg_14]
push    edi                     ; int
push    eax                     ; int
push    ebx                     ; String
call    NetpwPathType
cmp     eax, edi
jnz     short loc_7517F869
```

```
cmp     esi, ebx
ja      loc_7517FD3E
```

```
push    [ebp+String]        ; Source
lea     eax, [ebp+Destination]
push    eax                 ; Destination
call    ds:__imp_wcscpy
mov     ax, [ebp+esi*2+var_416]
pop     ecx
cmp     ax, 5Ch ; '\'
pop     ecx
jz      short loc_7517FCD5
```

```
cmp     ax, 2Fh ; '/'
jz      short loc_7517FCD5
```

```
push    [ebp+String]      ; Source
lea     eax, [ebp+Destination]
push    eax               ; Destination
call    ds:__imp_wcscpy
mov     ax, [ebp+esi*2+var_416]
pop     ecx
cmp     ax, 5Ch ; '\'
pop     ecx
jz      short loc_7517FCD5
```

```
cmp     ax, 2Fh ; '/'
jz      short loc_7517FCD5
```

```
lea     eax, [ebp+Destination]
push    offset asc_751717B8 ; "\\"
push    eax               ; Destination
call    ds:__imp_wcscat
pop     ecx
inc     esi
pop     ecx
```

```
loc_7517FCF4:              ; String
push    [ebp+Source]
call    edi ; __imp_wcslen ; calculate Source length (Unicode)
add     eax, esi           ; calculate the combined length of String and Source
pop     ecx
cmp     eax, ebx           ; compare the total length with 0x411
ja      short loc_7517FD3E ; if pass the boundary than quit
```

```
push    [ebp+Source]     ; Source
lea     eax, [ebp+Destination]
push    eax              ; Destination
call    ds:__imp_wcscat  ; concatenate String and Source --> Overflow!!
pop     ecx
lea     eax, [ebp+Destination]
pop     ecx
push    eax
call    sub_7518AE95
lea     eax, [ebp+Destination]
push    eax              ; String
call    sub_7518AEB3
test    eax, eax
jnz     short loc_7517FD43
```

```
call    edi ; __imp_wcslen ; calculate Source length (Unicode)
```

```
; Exported entry 303. NetpwPathCanonicalize


; Attributes: bp-based frame

; int __stdcall NetpwPathCanonicalize(wchar_t *Source, wchar_t *, int, wchar_t *String, int, int)
public NetpwPathCanonicalize
NetpwPathCanonicalize proc near

Source= dword ptr  8
arg_4= dword ptr  0Ch
arg_8= dword ptr  10h
String= dword ptr  14h
arg_10= dword ptr  18h
arg_14= dword ptr  1Ch

push    ebp
mov     ebp, esp
push    ebx
mov     ebx, [ebp+String]
push    esi
push    edi
xor     edi, edi
cmp     ebx, edi
jz      short loc_7517F7FA
```

Q & A