

PA193 - Secure coding principles and practices

**LABS: Language level vulnerabilities:
Buffer overflow, type overflow, strings**



Petr Švenda svenda@fi.muni.cz

CRCS

Centre for Research on
Cryptography and Security

```

// Note: GCC and MSVC uses different memory alignment
// Try "12345678DevilEvecosia" as a password for gcc build
// Try "1234567812345678Devil I am. Ha Ha" as a password for MSVC debug build

void demoBufferOverflowData() {
    int                unused_variable = 30;
#define NORMAL_USER   'n'
#define ADMIN_USER    'a'
    int                userRights = NORMAL_USER;
#define USER_INPUT_MAX_LENGTH 8
    char               userName[USER_INPUT_MAX_LENGTH];
    char               passwd[USER_INPUT_MAX_LENGTH];

    // print some info about variables
    printf("%-20s: %p\n", "userName", userName);
    printf("%-20s: %p\n", "passwd", passwd);
    printf("%-20s: %p\n", "unused_variable", &unused_variable);
    printf("%-20s: %p\n", "userRights", &userRights);
    printf("\n");

    // Get user name
    memset(userName, 1, USER_INPUT_MAX_LENGTH);
    memset(passwd, 2, USER_INPUT_MAX_LENGTH);
    printf("login as: ");
    fflush(stdout);
    gets(userName);

    // Get password
    printf("%s@vulnerable.machine.com: ", userName);
    fflush(stdout);
    gets(passwd);

    // Check user rights (set to NORMAL_USER and not changed in code)
    if (userRights == NORMAL_USER) {
        printf("\nWelcome, normal user '%s', your rights are limited.\n\n", userName);
        fflush(stdout);
    }
    if (userRights == ADMIN_USER) {
        printf("\nWelcome, all mighty admin user '%s'!\n", userName);
        fflush(stdout);
    }

    // How to FIX:
    //memset(userName, 0, USER_INPUT_MAX_LENGTH);
    //fgets(userName, USER_INPUT_MAX_LENGTH - 1, stdin);
    //memset(passwd, 0, USER_INPUT_MAX_LENGTH);
    //fgets(passwd, USER_INPUT_MAX_LENGTH - 1, stdin);
}

```

Setup

- Create new Visual Studio 2015 Project
 - File->New->Project->VisualC++->Win32 Console app
 - Turn off 'Precompiled header' and 'SDL checks'
- Paste BufferOverflow.cpp from IS instead of project's main file
- Try to compile (disable warning on gets() function)
 - #define _CRT_SECURE_NO_WARNINGS
- Insert breakpoint (begin of demoBufferOverflowData()) – F9
- Run program in debug mode – F5
- Execute next step of program – F10
- Display memory
 - Debug → Windows → Memory
 - Program must be in debugging session and running!

```

void demoBufferOverflowData() {
    int        unused_variable = 30;
#define NORMAL_USER    'n'
#define ADMIN_USER     'a'
    int        userRights = NORMAL_USER;
#define USER_INPUT_MAX_LENGTH 8
    char        userName[USER_INPUT_MAX_LENGTH];
    char        passwd[USER_INPUT_MAX_LENGTH];

    // print some info about variables
    printf("%-20s: %p\n", "userName", userName);
    printf("%-20s: %p\n", "passwd", passwd);
    printf("%-20s: %p\n", "unused_variable", &unused_variable);
    printf("%-20s: %p\n", "userRights", &userRights);
    printf("\n");

    // Get user name
    printf("login as: ");
    gets(userName);

    // Get password
    printf("%s@vulnerable.machine.com: ", userName);
    gets(passwd);

    // Check user rights (set to NORMAL_USER and not changed in code)
    if (userRights == NORMAL_USER) {
        printf("\nWelcome, normal user '%s', your rights are limited.\n\n", userName);
    }
    if (userRights == ADMIN_USER) {
        printf("\nWelcome, all mighty admin user '%s'!\n", userName);
    }
}

```

Variable containing current access rights

Array with fixed length (will be overwritten)

Help output of address of local variables stored on the stack

Reading username and password (no length checking)

Print information about current user rights

Data in memory

```

void demoBufferOverflowData() {
    int    unused_variable = 30;
#define  NORMAL_USER    'n'
#define  ADMIN_USER    'a'
    int    userRights = NORMAL_USER;
#define  USER_INPUT_MAX_LENGTH  8
    char   userName[USER_INPUT_MAX_LENGTH];
    char   passwd[USER_INPUT_MAX_LENGTH];

    // print some info about variables|
    printf("%-20s: %p\n", "userName", userName);
    printf("%-20s: %p\n", "passwd", passwd);
    printf("%-20s: %p\n", "unused_variable", &unused_variable);
    printf("%-20s: %p\n", "userRights", &userRights);
    printf("\n");

    // Get user name
    memset(userName, 1, USER_INPUT_MAX_LENGTH);
    memset(passwd, 2, USER_INPUT_MAX_LENGTH);
    printf("login as: ");
    fflush(stdout);
}

```

Memory 1

Address: 0x0024FB1B

0x0024FB1B	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	
0x0024FB2C	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FB3D	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FB4E	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FB5F	cc	02	02	02	02	02	02	02	02	02	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FB70	01	01	01	01	01	01	01	01	01	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	6e
0x0024FB81	00	00	00	cc	cc	cc	cc	cc	cc	cc	cc	1e	00	00	00	cc	cc	cc	cc	cc
0x0024FB92	cc	cc	85	20	45	b8	6c	fc	24	00	9a	20	dd	00	00	00	00	cc	cc	cc
0x0024FBA3	00	00	00	00	00	00	e0	fd	7f	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FBB4	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FBC5	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FBD6	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FBE7	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FBF8	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc
0x0024FC09	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc	cc

Autos Locals Memory 1 Threads Modules Watch 1

unused_variable

Running without malicious input

(Global Scope) demoBufferOverflowData()

```
// Get user name
memset(userName, 1, U
memset(passwd, 2, U
printf("login as: "
fflush(stdout);
gets(userName);

// Get password
printf("%s@vulnerab
fflush(stdout);
gets(passwd);

// Check user right
if (userRights == N
    printf("\nWelco
    fflush(stdout);
}
if (userRights == A
    printf("\nWelco
    fflush(stdout);
}
```

Memory 1

Address	Hex	ASCII
0x0013FA36	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0013FA47	cc 74 65 73 74 00 02 02 02 cc cc cc cc cc cc cc cc	itest...iiiiiii
0x0013FA58	70 65 74 72 00 01 01 01 cc cc cc cc cc cc cc cc 6e	petr...iiiiiii
0x0013FA69	00 00 00 cc cc cc cc cc cc cc cc cc cc 1e 00 00 00 cc cc	...iiiiiii...ii
0x0013FA7A	00 cc 9c 2f eb d8 54 fb 13 00 9a 20 f9 00 00 00 00	iïæ/ëøÛ...š ù....
0x0013FA8B	00 00 00 00 00 00 e0 fd 7f cc cc cc cc cc cc cc ccàý.iiiiiii
0x0013FA9C	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0013FAAD	cc	
0x0013FABE	cc	
0x0013FACF	cc	
0x0013FAE0	cc	
0x0013FAF1	cc	
0x0013FB02	cc	
0x0013FB13	cc	
0x0013FB24	cc	

D:\Documents\Develop\PB071_spring2010\pb071_cv11_vs\Debug\pb071_cv11_vs.exe

```
#### demoBufferOverflowData ####
userName      : 0013FA58
passwd        : 0013FA48
unused_variable : 0013FA74
userRights    : 0013FA68

login as: petr
petr@vulnerable.machine.com: test

Welcome, normal user 'petr', your rights are limited.
```

userName

passwd

Running with malicious input – userName

insert 'evil' into userName

```
// Get user name
memset(userName, 1, USER_INPUT_MAX_LENGTH);
memset(passwd, 2, USER_INPUT_MAX_LENGTH);
printf("login as: ");
fflush(stdout);
gets(userName);

// Get password
printf("%s@vulnerable.machine.com: ", userNam
fflush(stdout);
gets(passwd);

// Check user rights (set to NORMAL_USER and
if (userRights == NORMAL_USER) {
    printf("\nWelcome, normal user '%s', your
    fflush(stdout);
}
if (userRights == ADMIN_USER) {
    printf("\nWelcome, all mighty admin user
```

Address	Hex	ASCII
0x0024FB1B	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FB2C	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FB3D	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FB4E	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FB5F	c 02 02 02 02 02 02 02 cc cc cc cc cc cc cc cc	i.....iiiiiiii
0x0024FB70	65 76 69 6c 00 01 01 01 cc cc cc cc cc cc cc cc	evil...iiiiiiii
0x0024FB81	00 00 00 cc cc cc cc cc cc cc cc cc cc 1e 00 00 cc	...iiiiiiii...ii
0x0024FB92	cc cc 85 20 45 b8 6c fc 24 00 9a 20 dd 00 00 00 00	ii. E,lü\$.š Ý...
0x0024FBA3	00 00 00 00 00 00 e0 fd 7f cc cc cc cc cc cc ccàý.iiiiiiii
0x0024FBB4	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FBC5	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FBD6	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FBE7	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FBF8	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii
0x0024FC09	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiiiiiii

Running with malicious input received

Insert
'1234567812345678Devil I am. Ha Ha'
into passwd

```

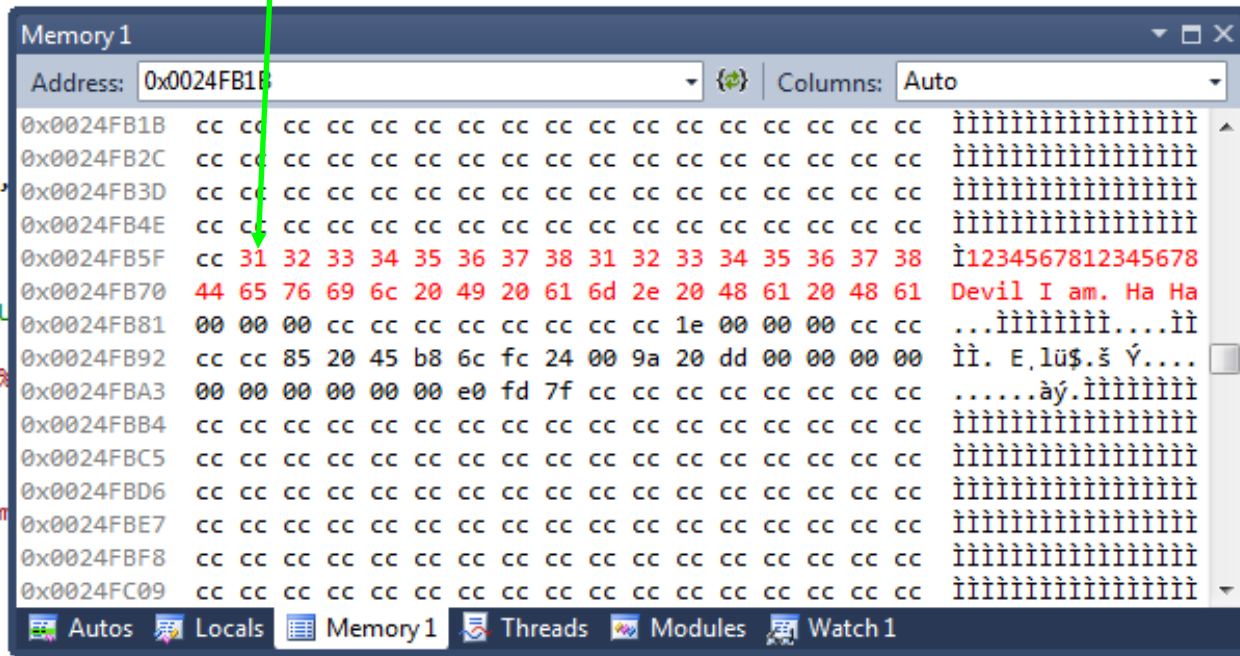
printf("login as: ");
fflush(stdout);
gets(userName);

// Get password
printf("%s@vulnerable.machine.com: ",
fflush(stdout);
gets(passwd);

// Check user rights (set to NORMAL_U
if (userRights == NORMAL_USER) {
    printf("\nWelcome, normal user '%s'
    fflush(stdout);
}
if (userRights == ADMIN_USER) {
    printf("\nWelcome, all mighty adm
    fflush(stdout);
}

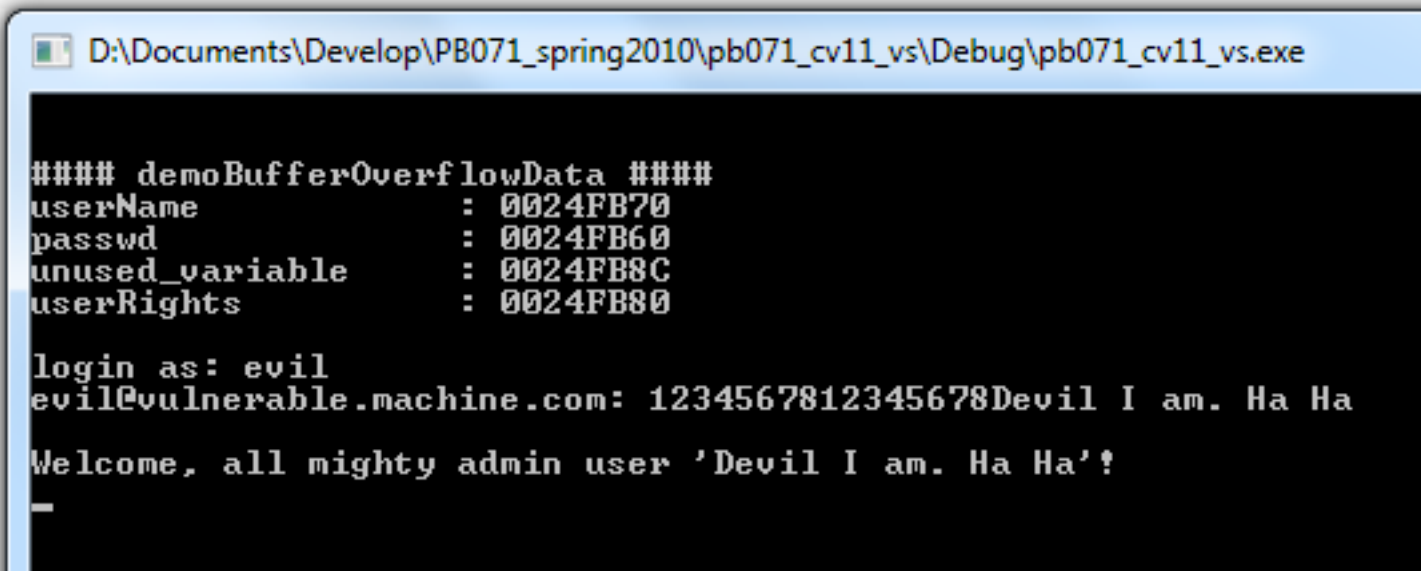
// How to FIX:

```



- Too long password overflow `userName` and `userRights`

Running with attacker input - result



```
D:\Documents\Develop\PB071_spring2010\pb071_cv11_vs\Debug\pb071_cv11_vs.exe

#### demoBufferOverflowData ####
userName      : 0024FB70
passwd        : 0024FB60
unused_variable : 0024FB8C
userRights    : 0024FB80

login as: evil
evil@vulnerable.machine.com: 1234567812345678Devil I am. Ha Ha

Welcome, all mighty admin user 'Devil I am. Ha Ha'!
```

Questions (debug mode)

- How are `userName`, `password` and `userRights` positioned in memory?
- How you will find memory location (address) of *userRights* variable?
- How many bytes you need to write into *userName* variable to change *userRights* ?
- Can you get admin rights by changing `userName` only?

Questions (debug mode)

- Why is program throwing debugger exception when finishing function `demoBufferOverflowData()`?
- How program was able to detect memory corruption?
- Why 0xcc bytes are here? How you can type 0xcc into terminal?
- Can you get admin rights without raising runtime exception (*memory around userName variable corrupted*) when leaving `demoBufferOverflowData()`?
- Where you can find return address?
- What should be the return address value?
 - Try R-Click -> Go to Disassembly

Questions (release mode)

- Release mode, /GS on
 - What is memory layout with respect to debug mode?
 - Can you still execute buffer overflow and change userRights?
 - What is the value of canary word?
- Release mode, /GS off
 - What is the influence of /GS disabled?
 - What is the impact on addresses of variables?
 - Can you be admin in Release? Why?

Lab – compiler protections

- GCC (e.g., QT Creator) & MSVC (Visual Studio)
 - list of compiler flags, release mode
- Compile program with/without compiler protection
 - `bufferoverflowdemo.cpp::demoBufferOverflowData()`
 - download from IS materials
 - return pointer smash behavior (crash, exception)
- Disassembly display of resulting binary
 - instruction-wise mode in IDE (Visual Studio), OllyDbg
 - existence of canary word (function with/without GS buffer)
- Display address of variable, function...,
 - run program multiple times – memory randomization (ASLR)

Compiler flags

- Locate all flags discussed during lecture
- Visual Studio Projects Settings
- Observe memory layout for stack frame with and without the flag
 - what is changing?
 - what is missing?

Compiler settings for /DEP and /ASLR

The screenshot shows the Microsoft Visual Studio IDE with the 'BufferOverflow Property Pages' dialog open. The dialog is configured for the 'Active(Debug)' configuration on the 'Active(Win32)' platform. The 'Code Generation' section is selected, and the 'Security Check' option is highlighted, showing it is set to 'Yes (/GS)'. Other options in the 'Code Generation' section include 'Enable String Pooling', 'Enable Minimal Rebuild', 'Enable C++ Exceptions', 'Smaller Type Check', 'Basic Runtime Checks', 'Runtime Library', 'Struct Member Alignment', 'Enable Function-Level Linking', 'Enable Parallel Code Generation', 'Enable Enhanced Instruction Set', 'Floating Point Model', 'Enable Floating Point Exceptions', and 'Create Hotpatchable Image'.

The background code in the editor shows the following preprocessor directives:

```
#include "stdio.h"
#include "memory.h"
// Note: GCC and MSVC uses different memory alignment
// Try "12345678DevilEvecosia" as a password for gcc build
// Try "1234567812345678Devil I am. Ha Ha" as a password for MSVC debug build
```

Deeper look into disassembly

BufferOverflow (Debugging) - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP

Process: [0x1A58] BufferOverflow.exe Thread: [0x21D0] Main Thread Stack Frame: demoBufferOverflowData

Solution Explorer: BufferOverflow (1 proj)

- External Dependencies
- Header Files
 - stdafx.h
 - targetver.h
- Resource Files
- Source Files
 - BufferOverflow.cpp
 - stdafx.cpp
 - ReadMe.txt

Disassembly: BufferOverflow.cpp

Address: demoBufferOverflowData(void)

Viewing Options

```

00EC152B call    __RTC_CheckEsp (0EC114Ah)
          gets(userName);
00EC1530 mov     esi,esp
00EC1532 lea   eax,[userName]
00EC1535 push  eax
00EC1536 call  dword ptr ds:[0EC92D4h]
00EC153C add   esp,4
00EC153F cmp   esi,esp
00EC1541 call  __RTC_CheckEsp (0EC114Ah)

// Get password
printf("%s@vulnerable.machine.com: ", userName);
00EC1546 mov     esi,esp
00EC1548 lea   eax,[userName]
00EC154B push  eax
00EC154C push  0EC58B4h
00EC1551 call  dword ptr ds:[0EC92D0h]
          .....
```

```

// Get user name
memset(userName, 1, USER_INPUT_MAX_LENGTH);
memset(passwd, 2, USER_INPUT_MAX_LENGTH);
printf("login as: ");
fflush(stdout);
gets(userName);

// Get password
printf("%s@vulnerable.machine.com: ", userName);
fflush(stdout);
gets(passwd);

// Check user rights (set to NORMAL_USER and not changed in code)
if (userRights == NORMAL_USER) {
    printf("\nWelcome, normal user '%s', your rights are limited.\n\n", userName);
    fflush(stdout);
}
```

Memory 1

Address: 0x0019FC54

Address	Hex	ASCII
0x0019FC54	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FC63	cc 6e 00 00 00 cc cc cc cc cc cc cc cc 1e 00	In...iiiiiiii..
0x0019FC72	00 00 cc cc cc cc cc 3b a6 7b d4 50 fd 19 00 e3	..iiii;{0Py..ä
0x0019FC81	17 ec 00 00 00 00 00 00 00 00 00 00 e0 fd 7e	.i.....äý~
0x0019FC90	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FC9F	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FCAE	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FCBD	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FCCE	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FCDB	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii
0x0019FCEA	cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc cc	iiiiiiiiiiiiiiii

Call Stack

Name
BufferOverflow.exeldemoBufferOverflowData() Line 25
BufferOverflow.exe!main() Line 56
BufferOverflow.exe!_tmainCRTStartup() Line 536
BufferOverflow.exe!mainCRTStartup() Line 377
kernel32.dll!754833aa()
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]
ntdll.dll!77139f72()
ntdll.dll!77139f45()

Deeper look into disassembly (cont.)

CPU - main thread, module AES_Pola

01101000	\$ 55	PUSH EBP
01101001	. 8BEC	MOV EBP,ESP
01101003	. 83EC 10	SUB ESP,10
01101006	. 56	PUSH ESI
01101007	. 833D 20501D01	CMP DWORD PTR DS:[aes_init_done],0
0110100E	> 75 0F	JNZ SHORT AES_Pola.011D101F
01101010	. E8 1B1A0000	CALL AES_Pola.aes_gen_tables
01101015	. C705 20501D01	MOV DWORD PTR DS:[aes_init_done],1
0110101F	> 8B45 10	MOV EAX,DWORD PTR SS:[EBP+10]
01101022	. 8945 F4	MOV DWORD PTR SS:[EBP-C],EAX
01101025	. 817D F4 000000	CMP DWORD PTR SS:[EBP-C],0
0110102C	> 74 14	JE SHORT AES_Pola.011D1042
0110102E	. 817D F4 C00000	CMP DWORD PTR SS:[EBP-C],0C0
01101035	> 74 16	JE SHORT AES_Pola.011D104D
01101037	. 817D F4 000100	CMP DWORD PTR SS:[EBP-C],100
0110103E	> 74 18	JE SHORT AES_Pola.011D1058
01101040	> EB 21	JMP SHORT AES_Pola.011D1063
01101042	> 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
01101045	> C701 0A000000	MOV DWORD PTR DS:[ECX],0A
01101048	> EB 20	JMP SHORT AES_Pola.011D106D
0110104D	> 8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]
01101050	> C702 0C000000	MOV DWORD PTR DS:[EDX],0C
01101056	> EB 15	JMP SHORT AES_Pola.011D106D
01101058	> 8B45 08	MOV EAX,DWORD PTR SS:[EBP+8]
0110105B	> C700 0E000000	MOV DWORD PTR DS:[EAX],0E
01101061	> EB 0A	JMP SHORT AES_Pola.011D106D
01101063	> B8 00F8FFFF	MOV EAX,-800
01101068	> E9 0B060000	JMP AES_Pola.011D1678
0110106D	> 8B4D 08	MOV ECX,DWORD PTR SS:[EBP+8]
01101070	. 83C1 08	ADD ECX,8
01101073	. 894D FC	MOV DWORD PTR SS:[EBP-4],ECX
01101076	. 8B55 08	MOV EDX,DWORD PTR SS:[EBP+8]
01101079	. 8B45 FC	MOV EAX,DWORD PTR SS:[EBP-4]
0110107C	. 8942 04	MOV DWORD PTR DS:[EDX+4],EAX
0110107F	. C745 F8 000000	MOV DWORD PTR SS:[EBP-8],0
01101086	> EB 09	JMP SHORT AES_Pola.011D1091
01101088	> 8B4D F8	MOV ECX,DWORD PTR SS:[EBP-8]
0110108B	. 83C1 01	ADD ECX,1
0110108E	. 894D F8	MOV DWORD PTR SS:[EBP-8],ECX
01101091	> 8B55 10	MOV EDX,DWORD PTR SS:[EBP+10]
01101094	. C1FA 05	SAR EDX,5
01101097	. 3955 F8	CMP DWORD PTR SS:[EBP-8],EDX

EST=00000001

Dump - 0020B000..0020FFFF

```

0020F97F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0020F98F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0020F99F  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  .....
0020F9AF  00 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  .0000000000000000
0020F9BF  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  .0000000000000000
0020F9CF  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  .0000000000000000
0020F9DF  01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01  .0000000000000000
0020F9EF  01 53 65 63 75 72 65 50 61 73 73 77 6F 72 64 3A  @SecurePassword
0020F9FF  6E 62 75 31 32 33 00 1D 01 58 72 1D 01 5C 72 1D  nbu123.#0x#0.r#
0020FA0F  01 60 72 1D 01 00 00 00 20 FA 20 00 F8 2F 1D 01 01 00 00 00 08 E1 4F 0* 2+00...iB0
0020FA1F  01 60 FA 20 00 95 32 1D 01 01 00 00 00 08 E1 4F 0* 2+00...iB0
0020FA2F  00 F8 7C 4F 00 7C 75 74 DB 00 00 00 00 00 00 00  .°10.iut█.....
0020FA3F  00 00 E0 FD 7E 00 00 00 34 FA 20 00 58 01 00  .°2°".....4°.X0.
0020FA4F  00 9C FA 20 00 89 36 1D 01 34 CD 49 DA 00 00 00  .: .ë6#04=I r...
0020FA5F  00 6C FA 20 00 AA 33 68 76 00 E0 FD 7E AC FA 20  .L .-3hv.0°2°%
0020FA6F  00 F2 9E EE 76 00 E0 FD 7E 77 18 E1 52 00 00 00  .=x°v.0°2°w†B R...
0020FA7F  00 00 00 00 00 E0 FD 7E 00 00 00 00 00 00 00  ......0°2°".....
0020FA8F  00 00 00 00 78 FA 20 00 00 00 00 FF FF FF  ......X:.....
0020FA9F  FF D5 71 F2 76 EB 27 2C 24 00 00 00 00 C4 FA 20  'q=vU',$,.....-
0020FAAF  00 C5 9E EE 76 6D 32 1D 01 00 E0 FD 7E 00 00 00  .+x°vm2#0.0°2°...

```

BinScope Binary Analyzer

- Download Microsoft SDL's Binscope
 - <https://www.microsoft.com/en-us/download/details.aspx?id=11910>
- Run BinScope Binary Analyzer (cmd or GUI)
 - `binscope.exe`
 - `binscope.exe /o results.xml targetApp.exe`
- Run on the binaries produced with different compiler settings
 - `/GS...`

Homework

- No homework this week, work on parser project
 - Implementation presentation till 13.10. (your seminar)
- What should you have already:
 - Formed group confirmed with me
 - Format for parsing confirmed with me
 - Setup Github repository for project
 - And link send to me!

**OPTIONAL: IF YOU LIKE TO
HAVE MORE FUN!**

Exploiting exercises

- Protostar image (<http://exploit-exercises.com>)
 - pre-prepared virtual machine
 - <http://exploit-exercises.com/protostar> (task description)
- **Important:** site now not available, use this link:
 - <https://web.archive.org/web/20140922114755/http://exploit-exercises.com/protostar>
 - Or protostar.zip in IS
- Login credentials: user / user; root / godmode
- Challenges stored in /opt/protostar/bin/ directory
 - stack0-7
- Run it, supply malformed input leading to crash
- Think about how to fix the source code

Protostar virtual image with exercises

exploit-exercises.com

News

Blog

Download

Exercises ▾

Follow us on twitter

Follow @exploitexercise

Protostar stack0

About

This level introduces the concept that memory can modify program execution.

This level is at `/opt/protostar/bin/stack0`

Source code

```
1#include <stdlib.h>
2#include <unistd.h>
3#include <stdio.h>
4
5int main(int argc, char **argv)
6{
7    volatile int modified;
8    char buffer[64];
9
10   modified = 0;
11   gets(buffer);
12
13   if(modified != 0) {
14       printf("you have changed the 'modified' variable\n");
15   } else {
16       printf("Try again?\n");
17   }
18}
```

Discussion

5 comments

```
EE - protostar 2 [Running] - Oracle VM VirtualBox
Machine View Devices Help
Password:
Linux (none) 2.6.32-5-686 #1 SMP Mon Oct 3 04:15:24 UTC 2011 i686
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
$ cd /opt/protostar/bin
$ ./stack0 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Try again?
$ ./stack0 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
Try again?
$ ./stack0
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
you have changed the 'modified' variable
Segmentation fault
$ _
```

STACK LEVELS

Stack 0

Stack 1

Stack 2

Stack 3

Stack 4

Stack 5

Stack 6

Stack 7

FORMAT STRING LEVELS

Format 0

Format 1

Format 2

Format 3

Format 4

HEAP LEVELS

Heap 0

Heap 1

23

5 | Buff