XFOCUS TEAM
from internet,for internet

Advanced shellcode Technique

plan9@jump.net.cn

X'CON 2004

X'con 2004

What will we learn

- Going through Firewall
- Evading NIDS
- Avoiding Application filter
- Defeating stack protect mechanism
- Evading HIDS

Shellcode attack

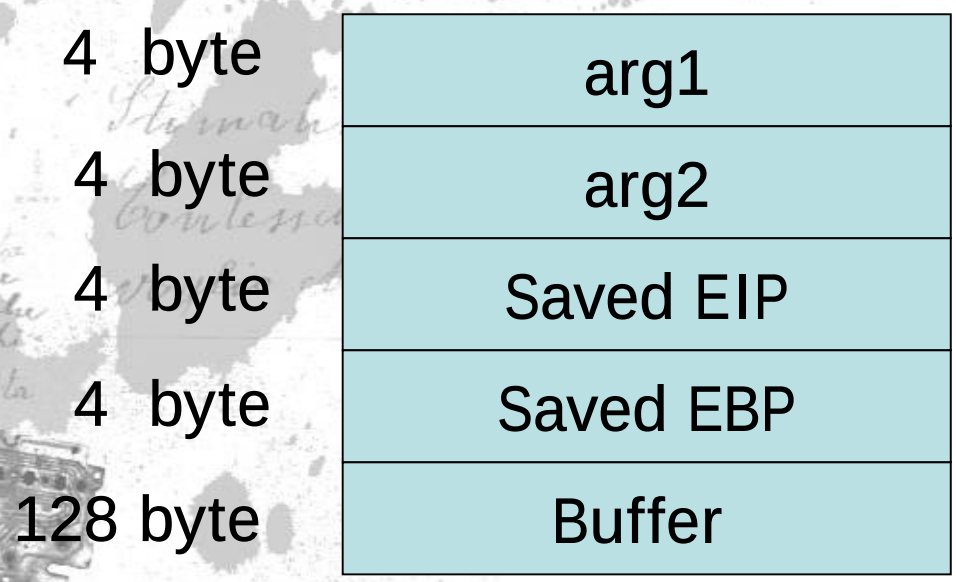
- Hijacking the execution flow of vulnerable process …
- …to owned buffer
  - Arbitrary syscalls execution with the privileges and the execution context of the hijacked process
    - Most time spawans a shell,so called “shellcode”
  - Execution manual constructed function

Shellcode attack(con.)

- Stack overflow

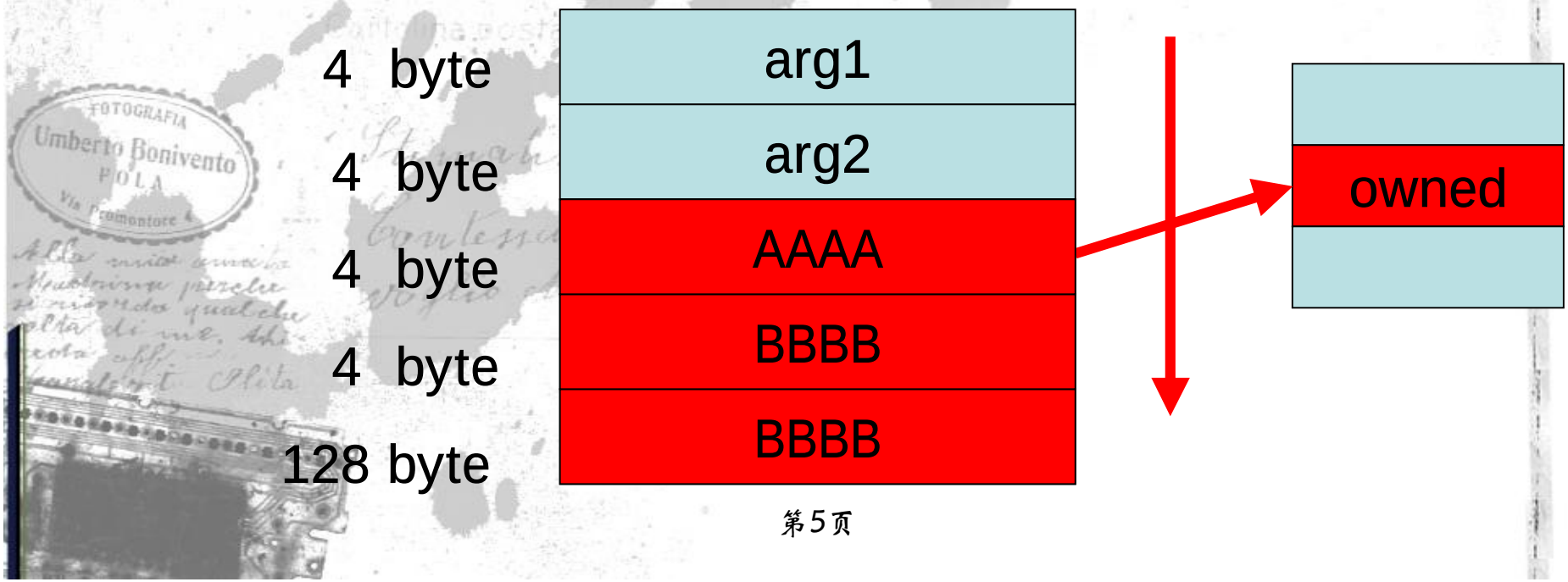
```
void fun(char *arg2,long arg1){
    char buffer[128];
    strcpy(buffer,arg2); /* w/o bounds check */
        return 0;}
```

- X86 stack frame



Shellcode attack(con.)

- If arg=BBB…BAAAA=136 bytes
  - After execution strcpy()
- X86 stack frame



Shellcode attack(con.)

- Inject vector **à** Warhead
  - Bug that exploited can hijacking execution flow
    - external : environment variables/global parameter,file, fields of protocol ,socket…any input feeds the buffer!
    - internal: Stack Overflow,Heap overflow,Integer overflow,Unsigned/signed,Format string,Calculate error
- Payload **à** Missile
  - Actual assembly instuctions that do expected stuff
    - Shell,virus,wrom,rootkit and anything you can think!

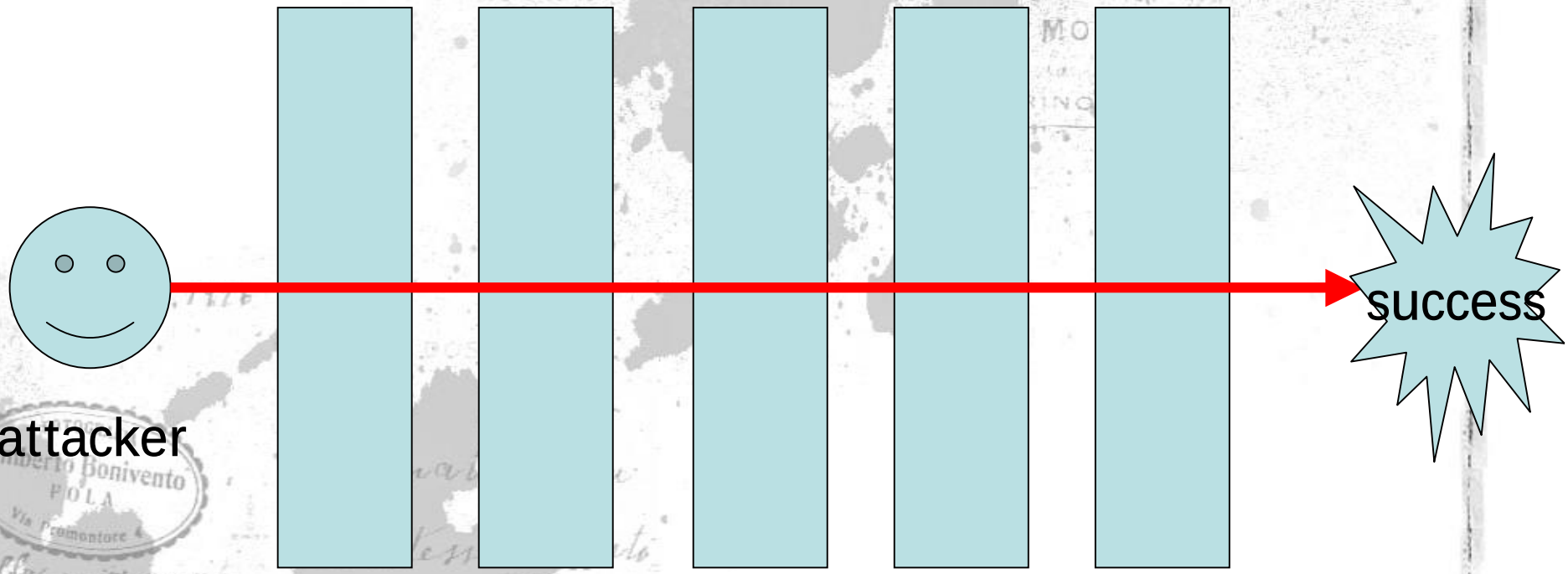
Why learn shellcode technique

- Very common attack vector
  - external :CERT Advisory,BUGTRAP
  - internal :un-security coding,man's error
- Very effective attacks
  - Run attacker's code of choice
- But many stuff set gap to success
  - Firewall,NIDS,HIDS,OS

Shellcode attack thinking

- Targets system police is unknown and strict
  - Can’t write to file,call demanded API…
- But we can exploring process belonging
  - fd,socket,data in memory, privilege,preloaded api…
- We need “small shell”
  - set up channel
  - Upload file and execution command
  - Prepare for further attack

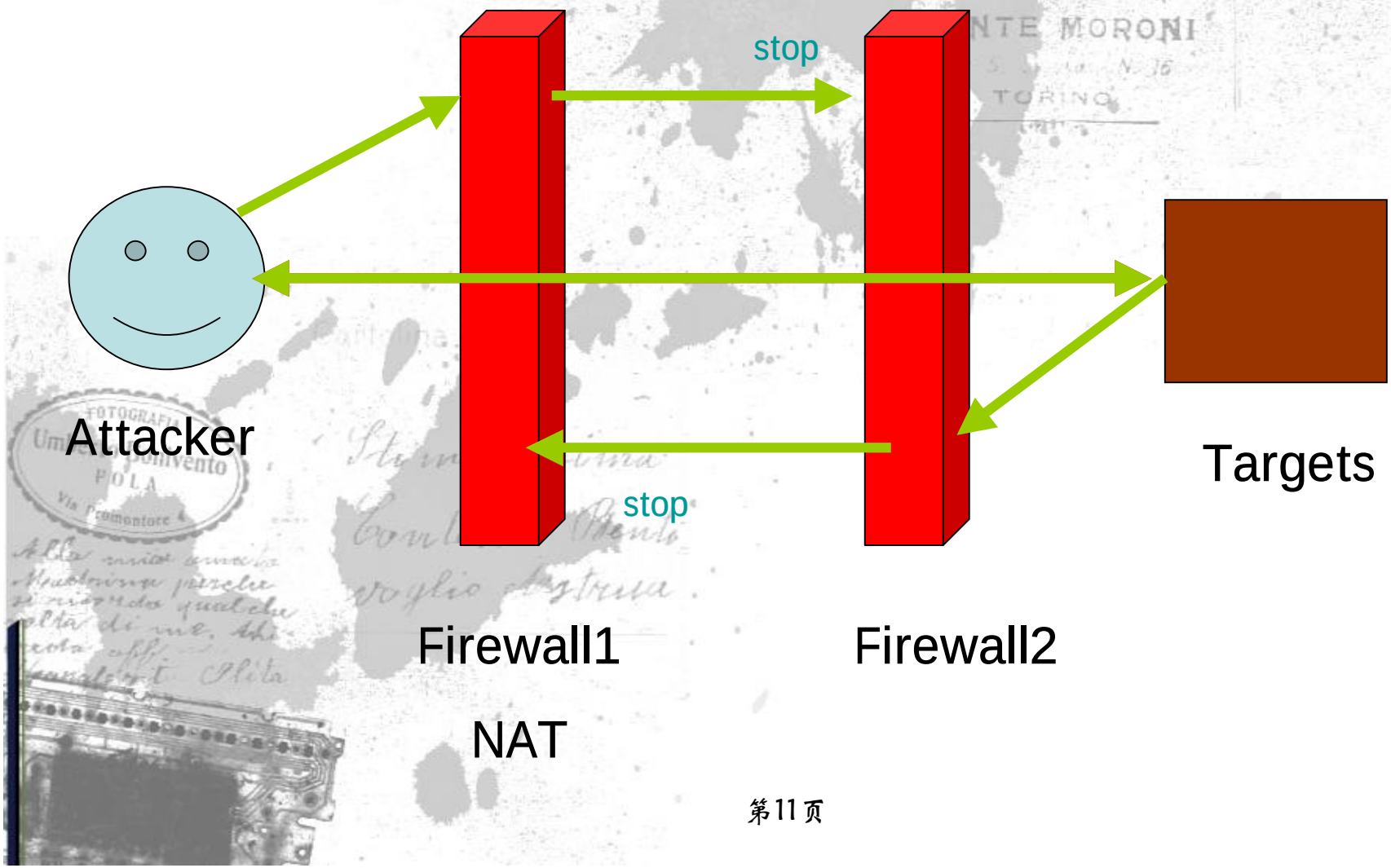
Shellcode attack path



Shellcode attack path:Firewall

- Connection contrack ,NAT
  - bind socket can’t work
- If we behind nat stuff
  - Reverse connection can’t wrok
- Our method
  - Find socket,reuse it!

Shellcode attack path:Firewall



Shellcode attack path:NIDS

- String greper
  - Can’t include quasi“/bin/bash”
- Nop seld inspect, instruction emulation
    - Too many cpu cycles
- Our method :polymorphic shellcode

Shellcode attack path:APP Filter

- Strcpy() forbid NULL
  - Shellcode can't include NULL
  - Can't use instruction included NULL
  - RET can't include NULL
- alphanumeric&UNICODED
- Buffer Size restriction
- Our method:multi-stage&transform proof shellcode

Shellcode attack path:stack protect

- BoWall
  - Inspect strcpy()
    - Need trickly set EIP
- Windows 2003
  - Stack cookie
    - Need trickly set EIP
- OverflowGuard
  - No-exec stack/heap
- Our method:SEH&ret-into-lib

Shellcode attack path:HIDS

- Application HIDS
  - ACL,honeytoken,system log
    - Can’t optional access file,can’t produce exception…
- Kernel HIDS
  - Windows token , Entercept
    - Can’t optional call API
    - Normal shellcode attack can’t work
- Our method: exploring process belonging: eg.preloaded API

Shellcode implement

- Firewall: find socket and reuse it.
  - Find accept() return value

```
00401171    LEA EAX,DWORD PTR SS:[EBP-4]
00401174    PUSH EAX                              ;pAddrLen
00401175    LEA EAX,DWORD PTR SS:[EBP-DC]
0040117B    PUSH EAX                              ; pSockAddr
0040117C    PUSH DWORD PTR SS:[EBP-29C]       ; Socket
00401182    CALL <JMP.&WS2_32.#1>                 ; accept
00401187    MOV DWORD PTR SS:[EBP-298],EAX  ;[EBP-298]=0x0012FD04
0040118D    CMP DWORD PTR SS:[EBP-298],0
00401194    JGE SHORT server2.004011A9
00401196    PUSH server2.004090C0                 ;  ASCII "Error on accept!"
0040119B    CALL server2.00402E9E
```

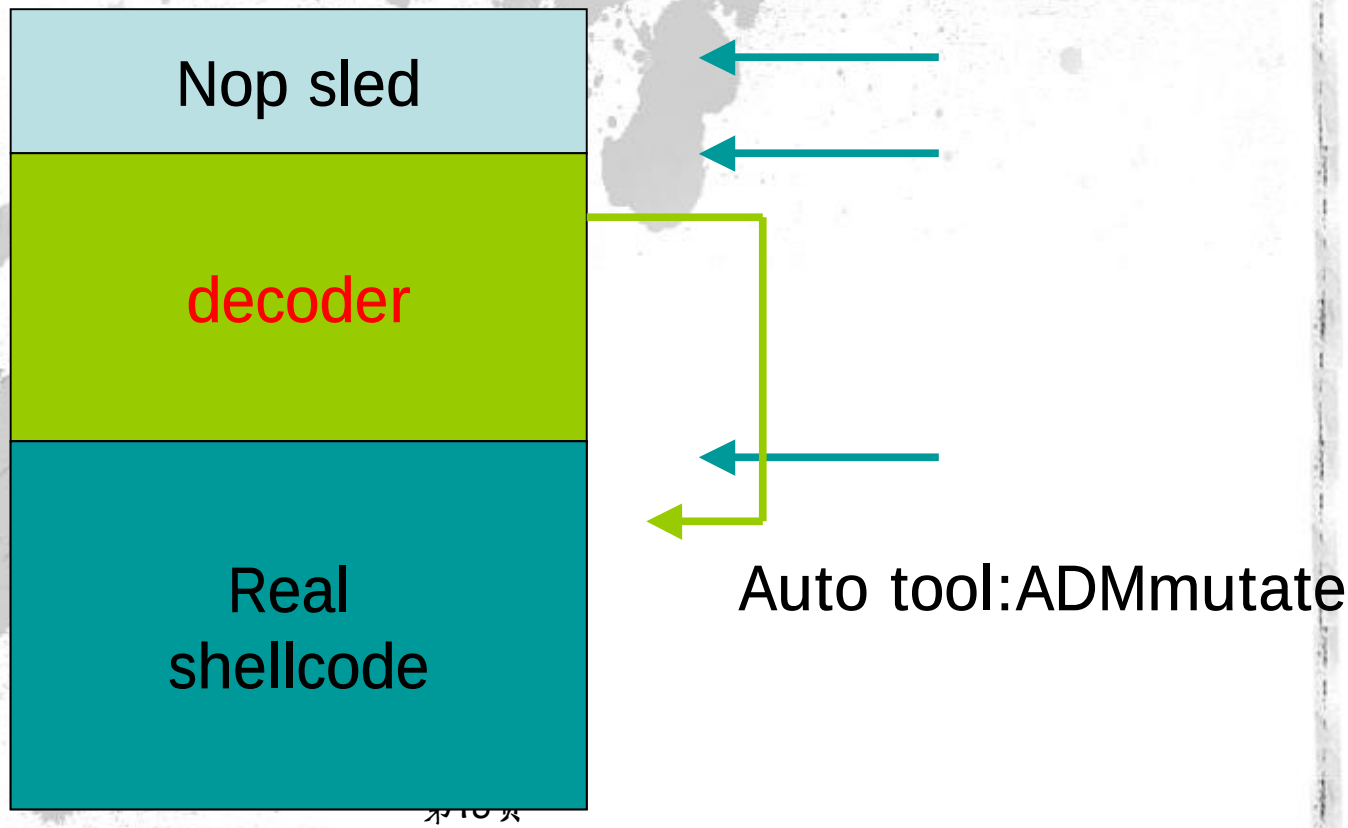
Shellcode implement(con.)

- Firewall:Find socket and reuseit!
  - Find accept() return value
    - EBP changed: Fix relative distance from top of frame

```
0x89,0xE5,                      MOV EBP,ESP
0x66,0x81,0xEC,0xF0,0x03        SUB SP,3F0
0x31,0xC0,                      XOR EAX,EAX
0x50,                           PUSH EAX          ;flags
0x6A,0x7F,                      PUSH 7F           ;size
0x8D,0x44,0x24,0x08,            LEA EAX,SS:[ESP+8] ;buffer
0x50,                           PUSH EAX
0x8D,0x45,0x20,                 LEA EAX, SS:[EBP+20] ;sock
0xFF,0x30,                      PUSH DWORD PTR DS:[EAX]
0xB8,0xFF,0xEC,0x30,0x40,       MOV EAX,4030ECFF
0xC1,0xE8,0x08,                 SHR EAX,8
0xFF,0xD0,                      CALL EAX
```

Shellcode implement(con.)

- NIDS: Polymorphism shellcode



Shellcode implement(con.)

- NIDS: polymorphic shellcode
  - Nop sled
    - Don't do harm and nop quasi instruction
    - Nop,inc eax…
  - decoder
    - multiple code paths
    - Push aaaa,pop eax=xor eax,eax,xor eax aaaa=mov eax 0,add eax aaaa=…
  - Real code :xored

Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:alphanumeric
    - X86 instruction format

    [opcode] [Mode R/M byte] [<SIB>] [<disp8>/<disp32>]
    - All field of instruction only be [0-9],[A-Z],[a-z]
    - inc ebx /* 0x43 = C */

Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - UNICODED

AABBCCXX=00AA00BB00CC00XX (XX<=0x7f)

    - Instruction can use

One byte instruction :inc eax /* 0x40 */

two byte instruction :add [esi],ch /*0x002E*/

three byte instruction :add byte ptr [ebp] ,ch /*0x006d00*/

five byte instruction :add eax ,4C007500h /*0x050075004C */

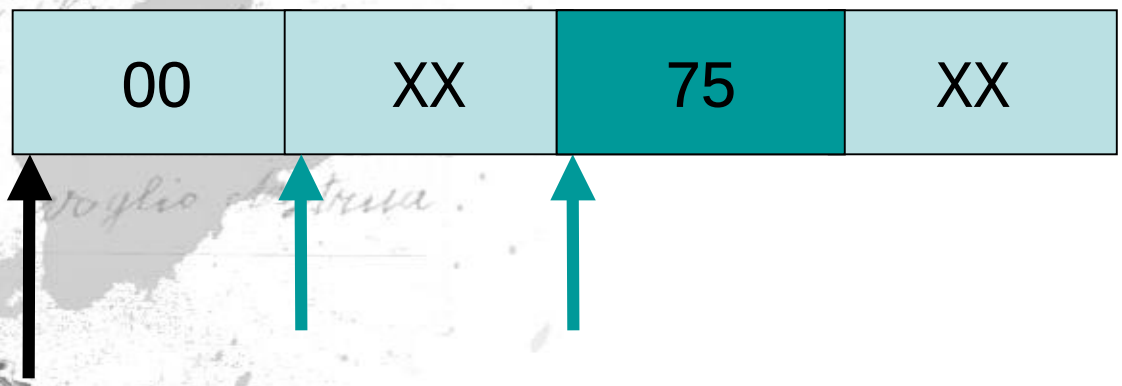
Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[1] shellcode
      - Named by chris@nextgenss.com

```
40          :inc eax
00 6D 00 :add byte ptr [ebp],ch
40          :inc eax
00 6D 00 :add byte ptr [ebp],ch
80 00 75 :add byte ptr [eax],75h
00 6D 00 :add byte ptr [ebp],ch
```

Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
- For who you don't know:UNICODE
  - Venetian[1] shellcode

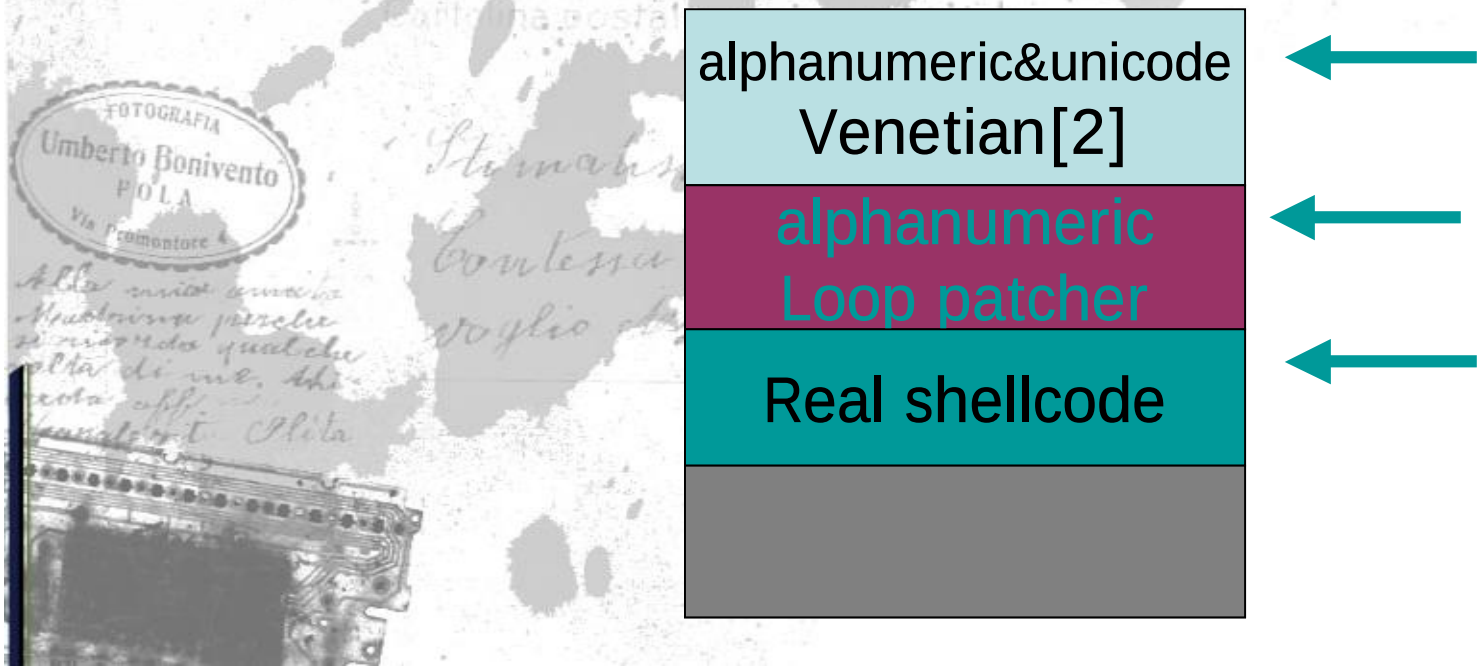


Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[1] problem
      - Buffer Size restrictions
        - » 14 byte get two byte shellcode
      - W/O consider alphanumeric

Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[2] shellcode



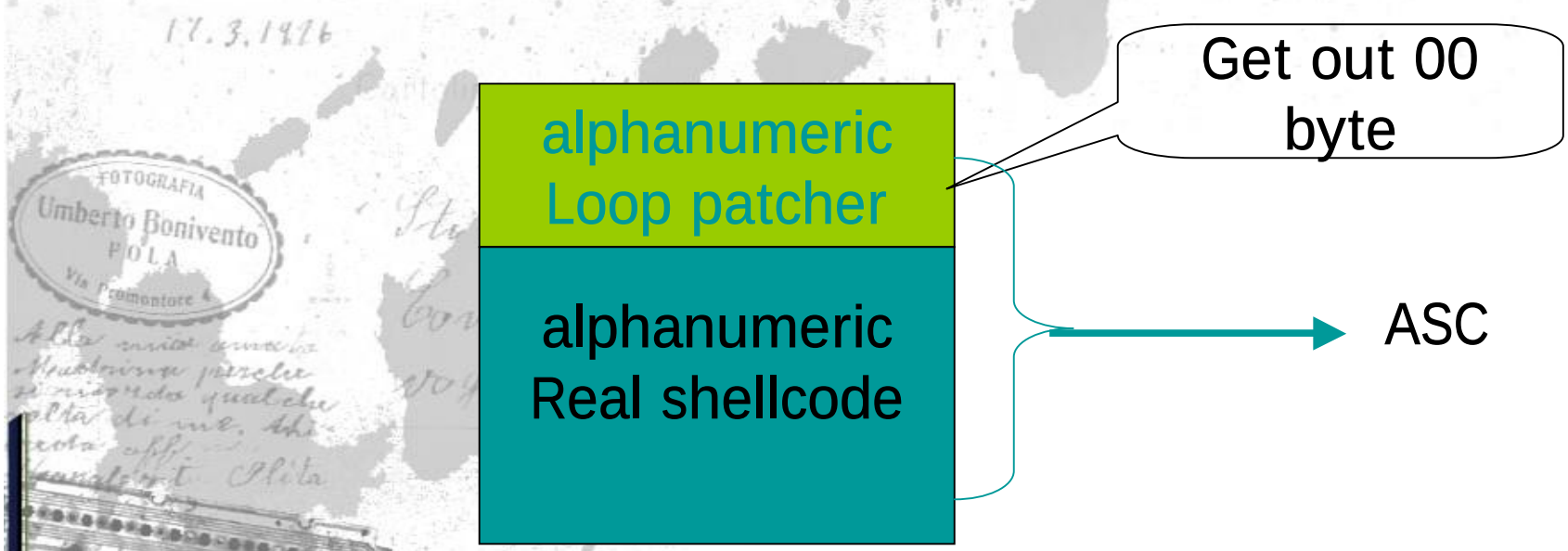
Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[2] shellcode

```
41              :inc   ecx                    /*A*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
41              :inc   ecx                    /*A*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
51              :push ecx                     /*Q*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
50              :pop eax                      /*P*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
68 00 75 00 68  :push 68007500h               /*huh*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
5A              :pop edx                      /*Z*/
00 6D 00        :add byte ptr [ebp],ch        /*m*/
41              :inc   ecx                    /*A*/
00 70 00        : add byte [eax],dh           /*p*/
```

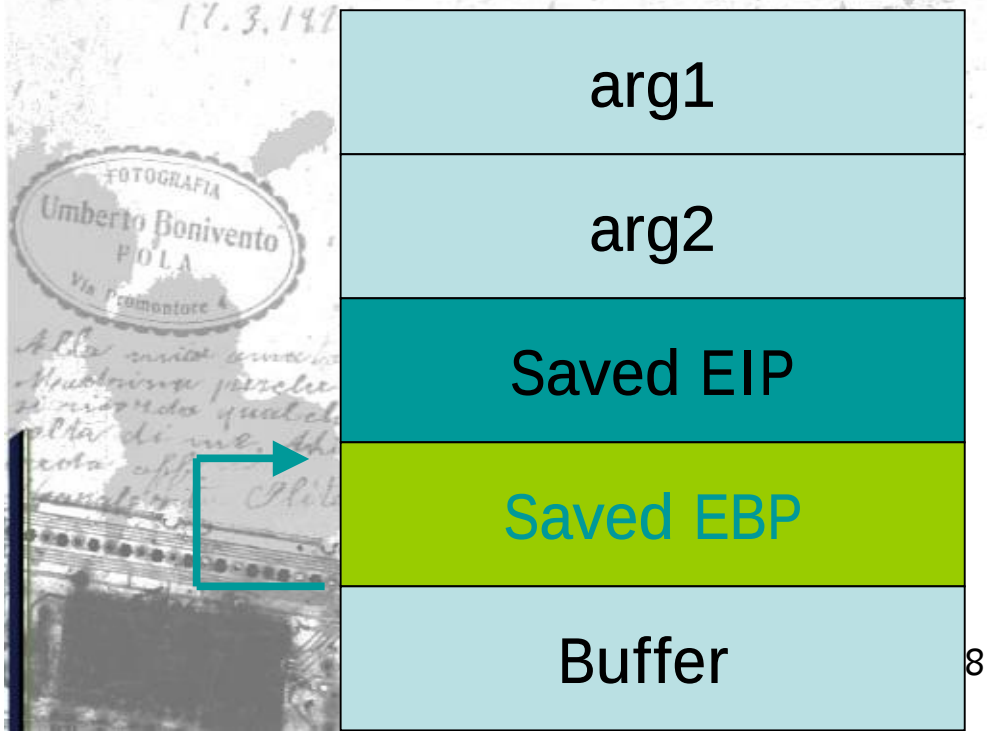
Shellcode implement(con.)

- APP Filter: alphanumeric&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[2] shellcode



Shellcode implement(con.)

- Stack protect:SEH&ret-into-lib
  - For who you don't know:BoWall
    - Strcpy() inspect : SEH



Saved Ret=[EBP+4]

↓

Strcpy()

↓

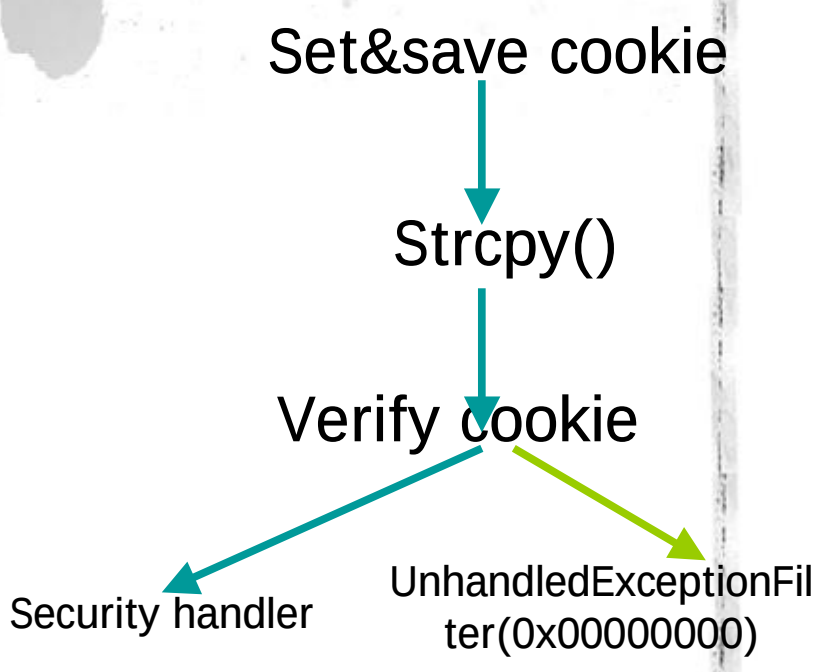
Get ret

↓

Verify

Shellcode implement(con.)

- Stack protect:SEH&ret-into-lib
  - For who you don't know:windows 2003
    - Stack cookie: SEH

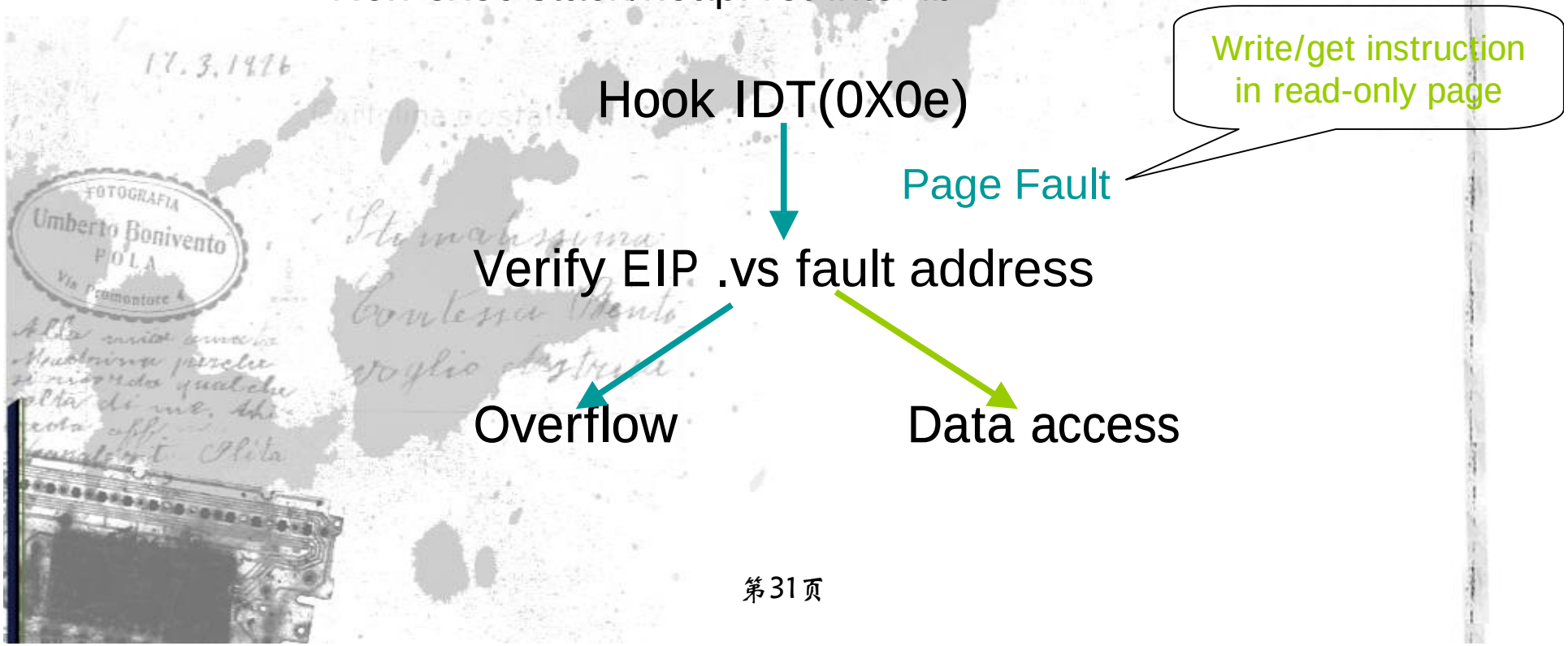


Shellcode implement(con.)

- Stack protect:SEH&ret-into-lib
  - For who you don't know:windows 2003
    - SEH
      - Clear most important register to zero
        - » call ebx,jump eax can't work...
      - load configuration has a copy of Exception handler
        - » optional set **Exception handler can't work**
        - » **Can exploring registered Exception handler**
      - Exception handler out of module address can work
      - Exception handler can't point to stack,but heap ok
        - » Set **Exception handler point to heap**

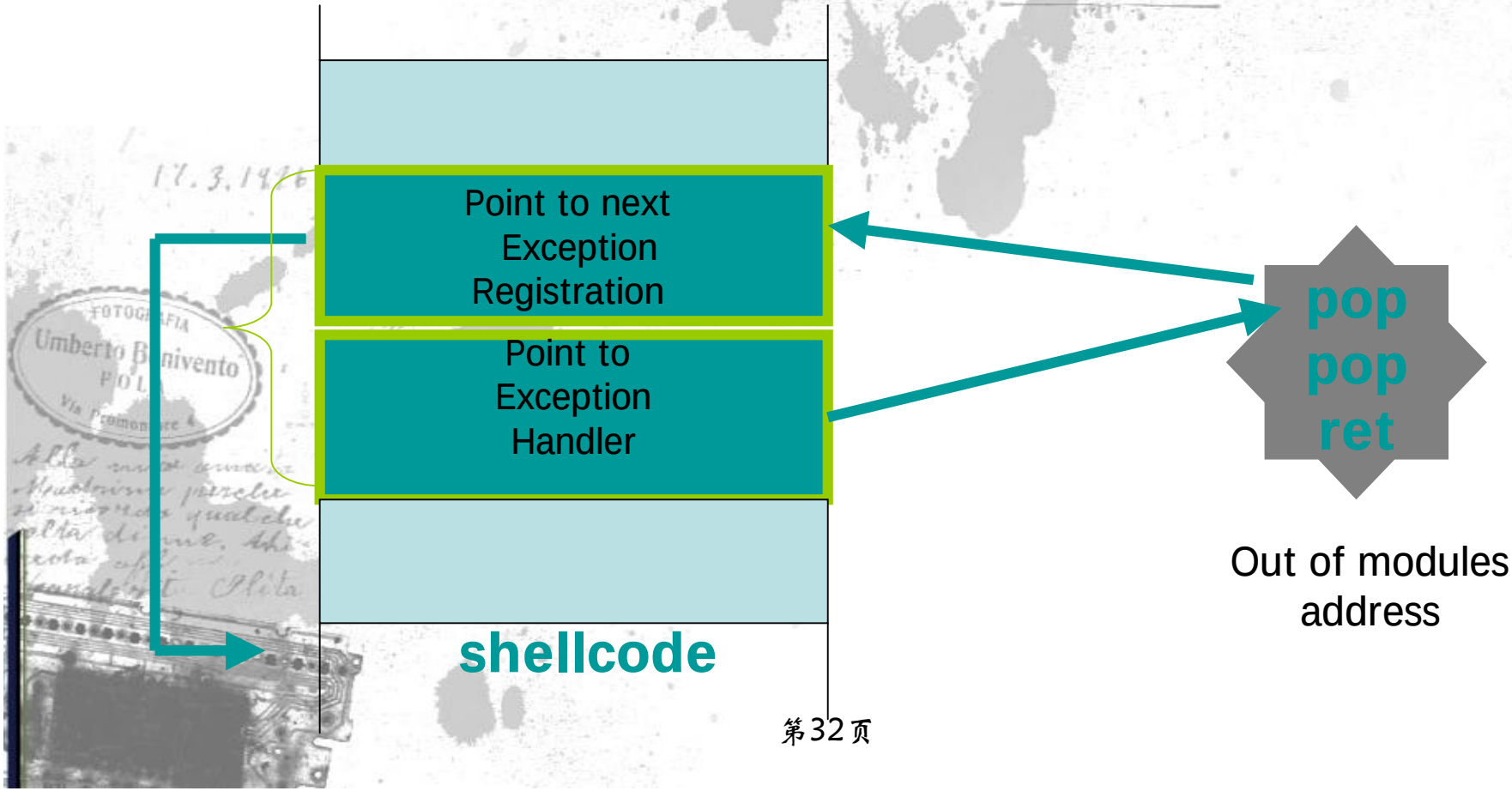
Shellcode implement(con.)

- Stack protect:SEH&ret-into-lib
  - For who you don't know:OverflowGuard
    - Non-exec stack/heap: ret-into-lib



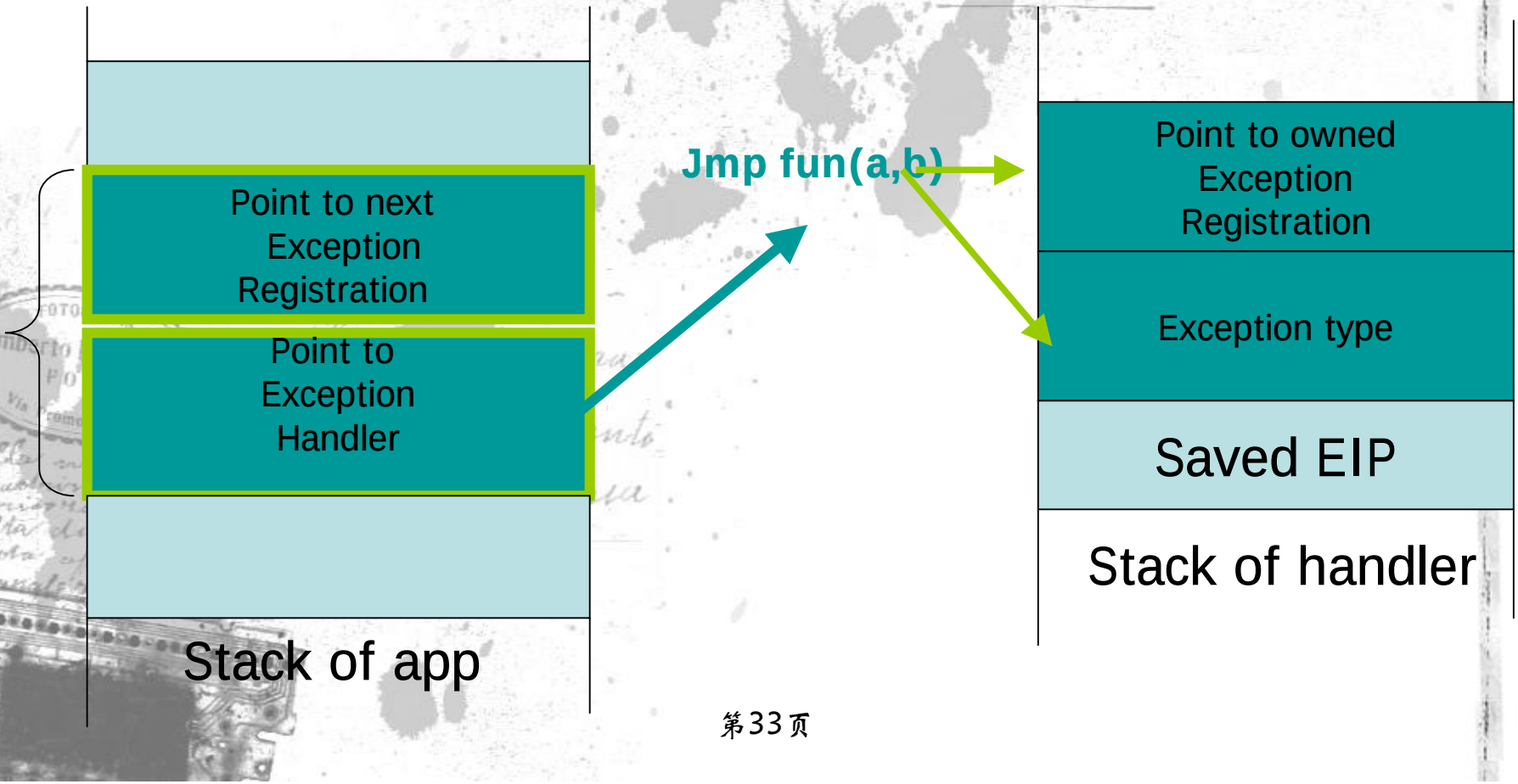
Shellcode implement(con.)

- Stack protect:Win2003&BoWall w/o non-exec



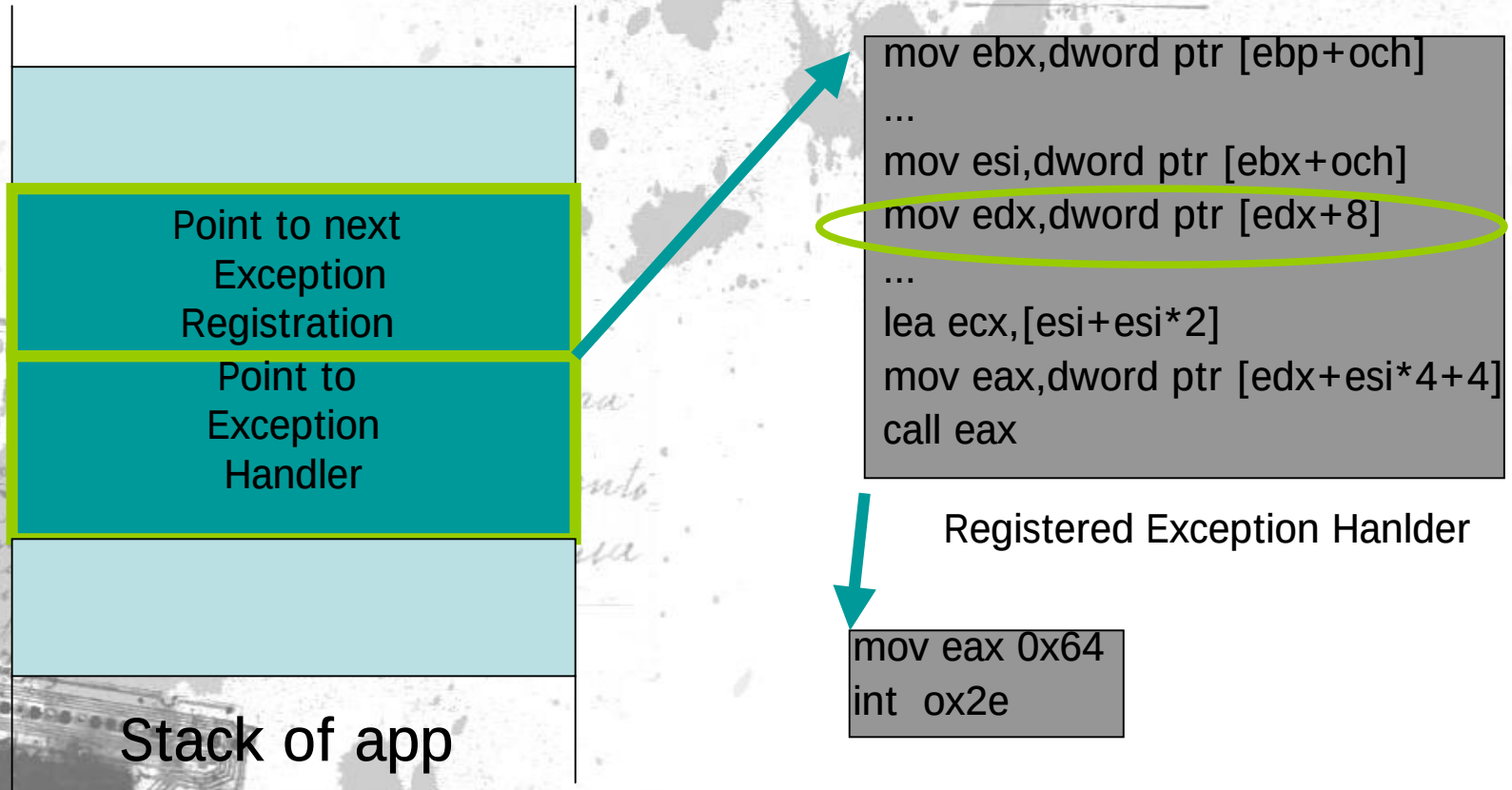
Shellcode implement(con.)

- Stack protect:Win2003&BoWall w/ non-exec



Shellcode implement(con.)

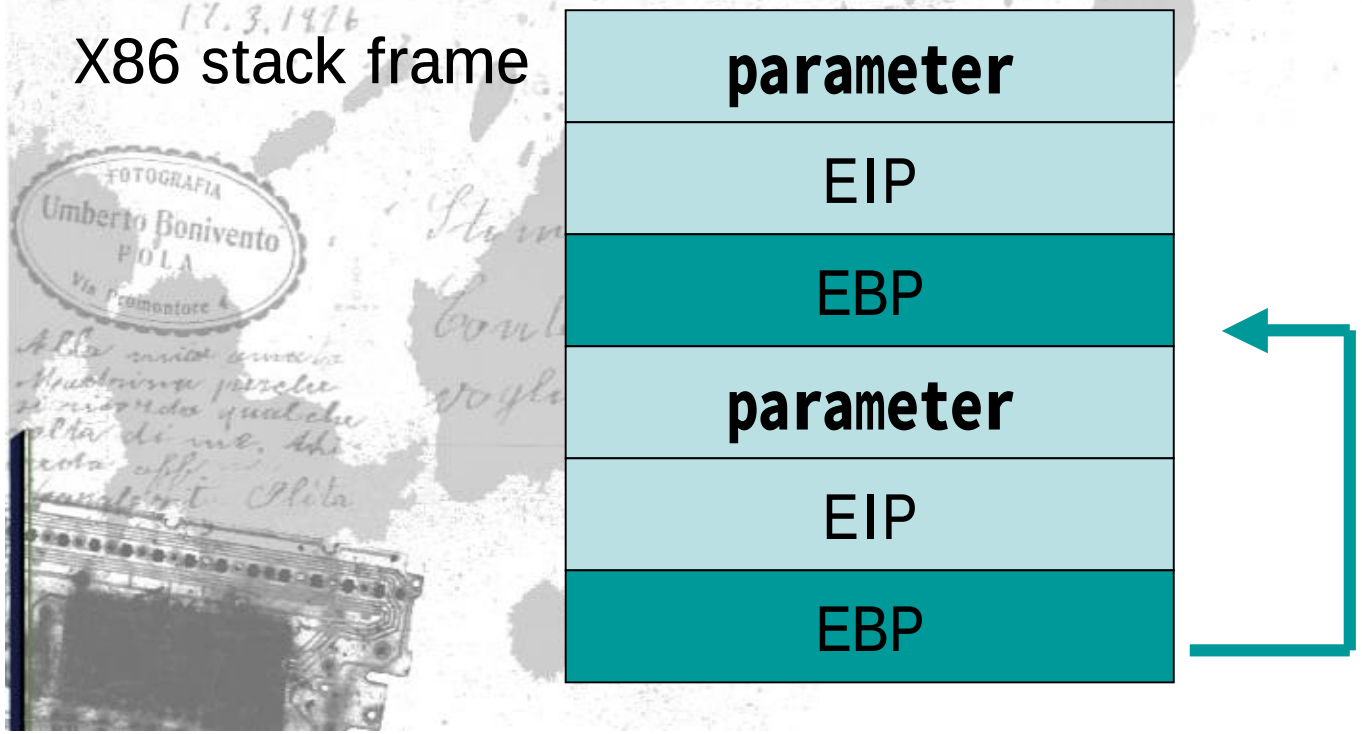
- Stack protect:Win2003&BoWall w/ non-exec



Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: Entercept
    - Hooked API: fake frame

X86 stack frame



Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: Entercept
    - Hooked API: fake frame

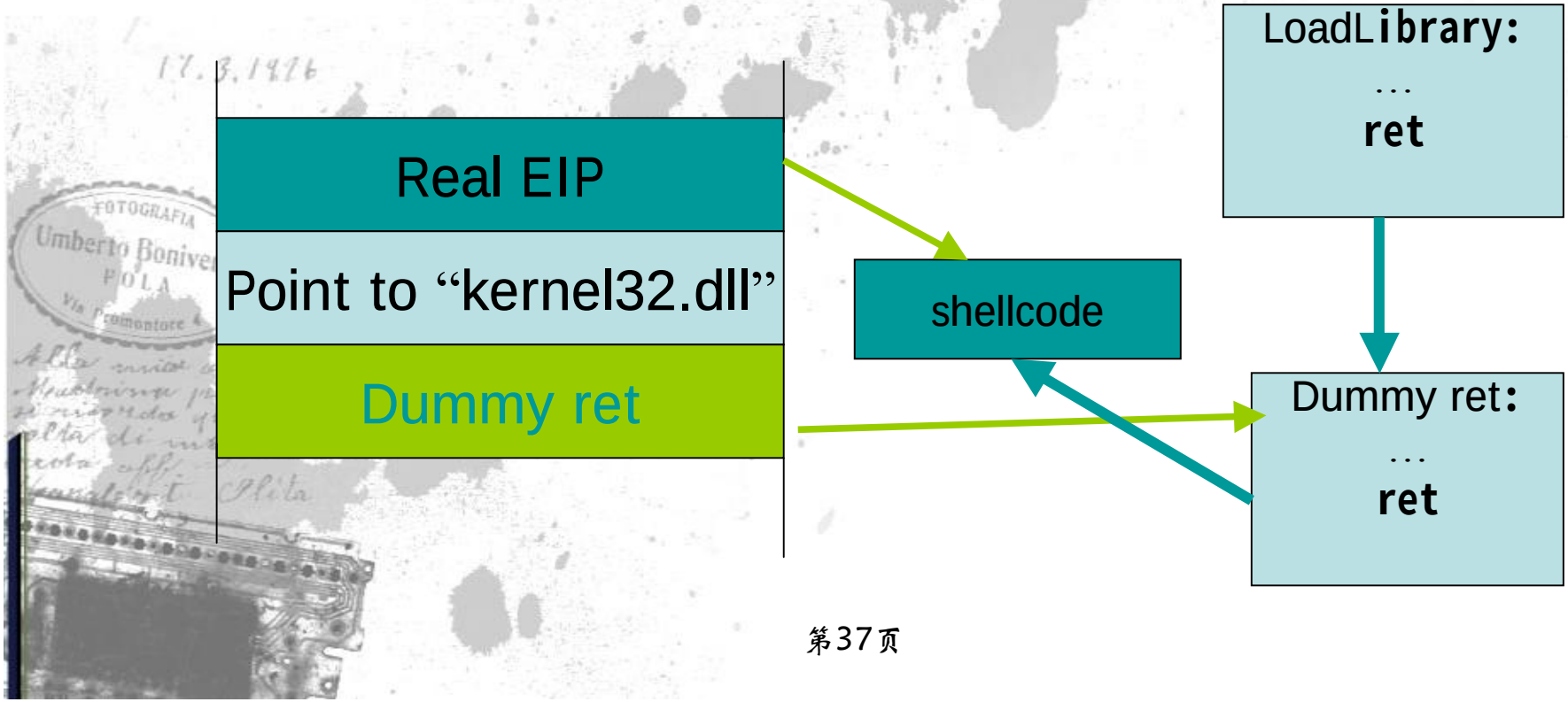
Stack backtrace

```
while (is_valid_frame_pointer( ebp )) {
        ret_addr = get_ret_addr( ebp )
if (check_code_page(ret_addr) == BUFFER_OVERFLOW)
        return BUFFER_OVERFLOW;
if (does_not_follow_call_or_jmp_opcode(ret_addr))
        return BUFFER_OVERFLOW;
        ebp = get_next_frame( ebp ); }
```

ret verify:
read-only memory mapped following jmp/call

Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: Entercept
    - Hooked API: fake frame



Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: IAT
    - Analyze IAT

PE offset

use Ascii name's CRC in shellcode

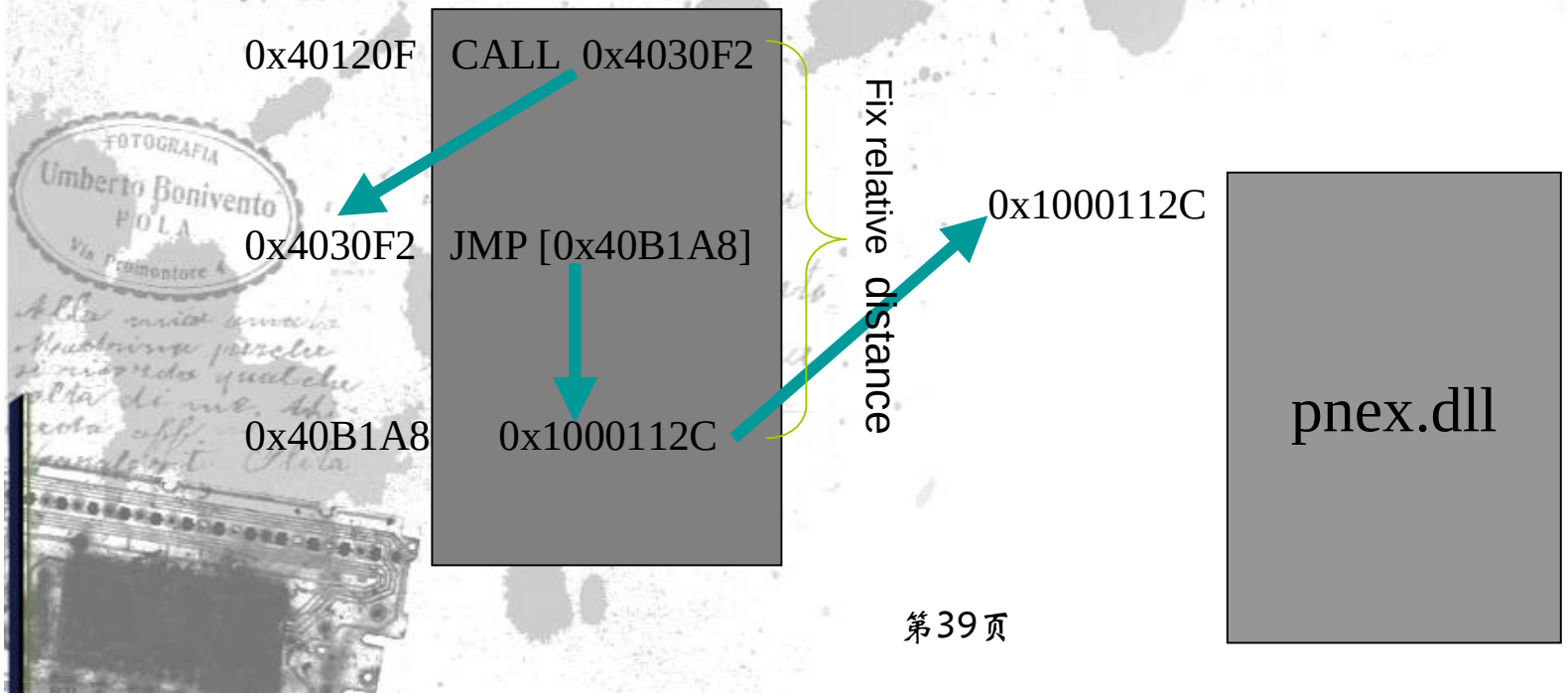
Optional Header

Ascii name

Address

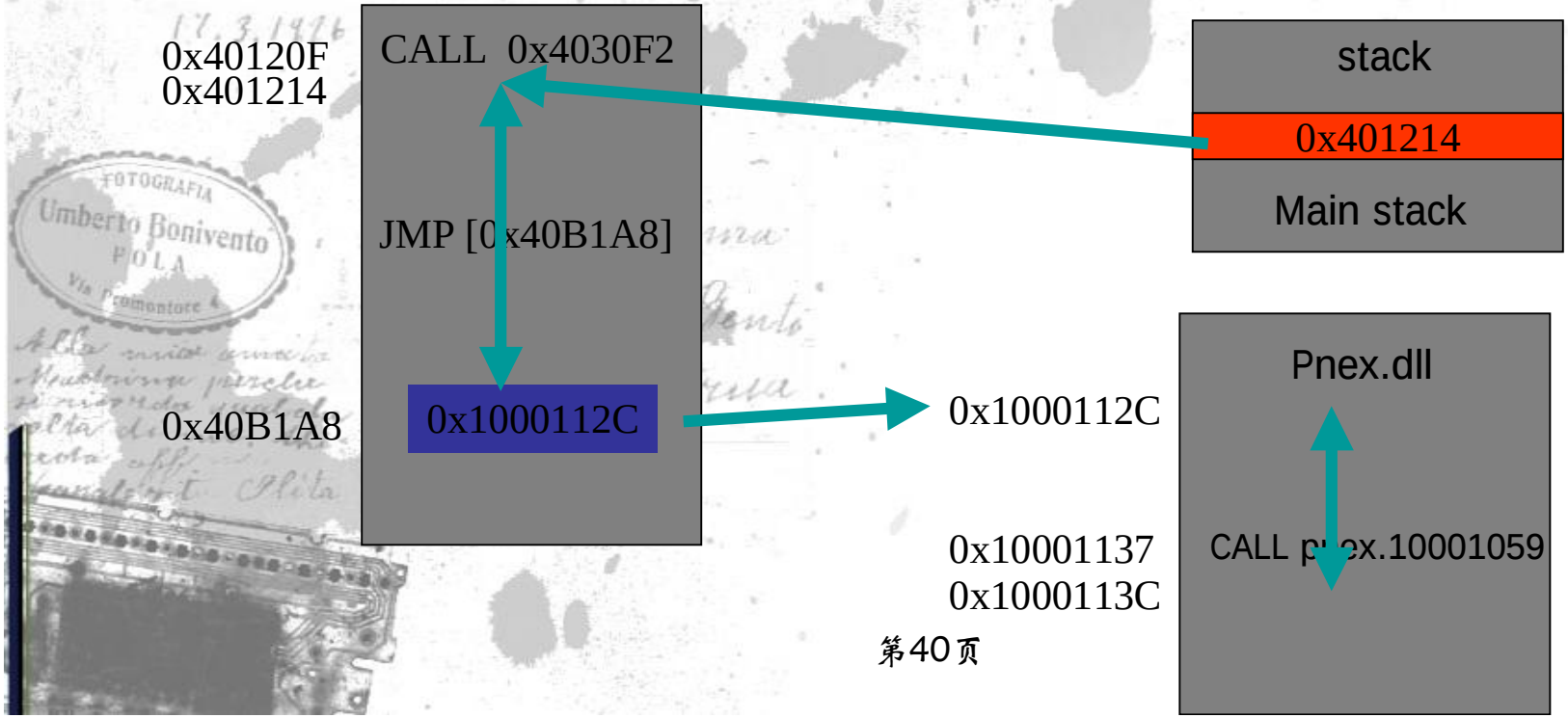
Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: Calculate RET
    - Analyze IAT



Shellcode implement(con.)

- HIDS:fake frame&IAT& Calculate RET
  - For who you don't know: Calculate RET
    - Analyze IAT



Demo

- Implement some discussed technique
- vulnerable server &sploits

XFOCUS TEAM
from internet,for internet

Thank You

Question ?

X'CON 2004