高级shellcode设计技巧

plan9@jump.net.cn

What will we learn

- 穿越防火墙
- 躲避NIDS
- 防止应用过滤
- 绕过栈溢出保护措施
- 躲避HIDS

Shellcode攻击

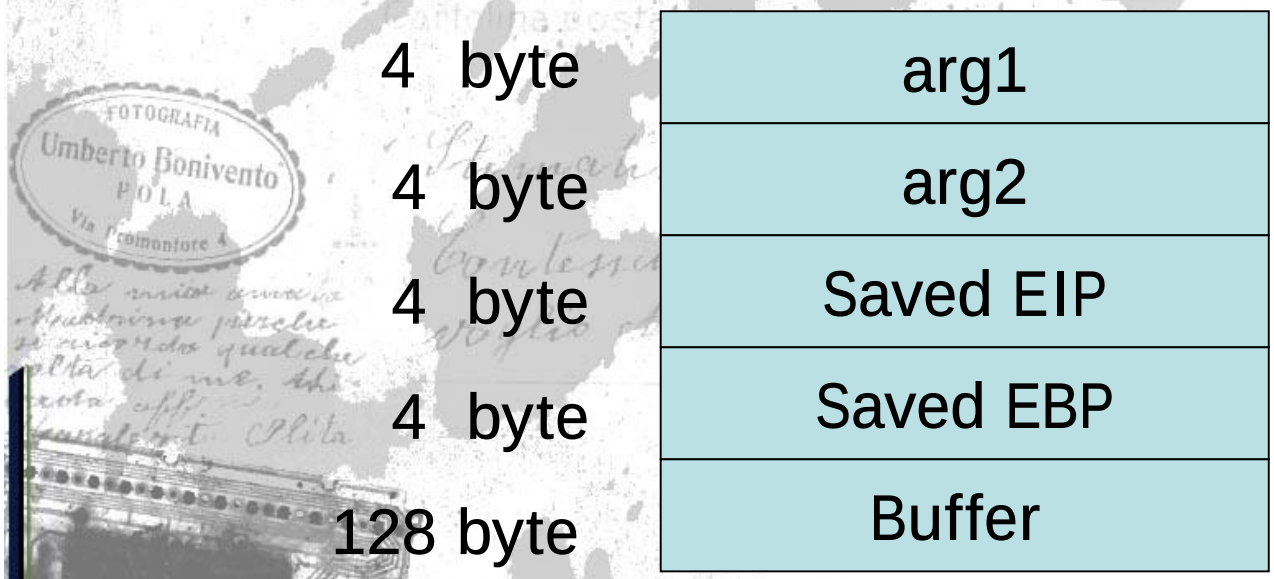
- 劫持进程执行流...
- ...控制的内存区
  - 在进程上下文和特权上下文中调用系统调用
    - 多数情况执行给出交互shell的代码
  - 执行自己构造的功能调用

Shellcode攻击(con.)

- **stack溢出**

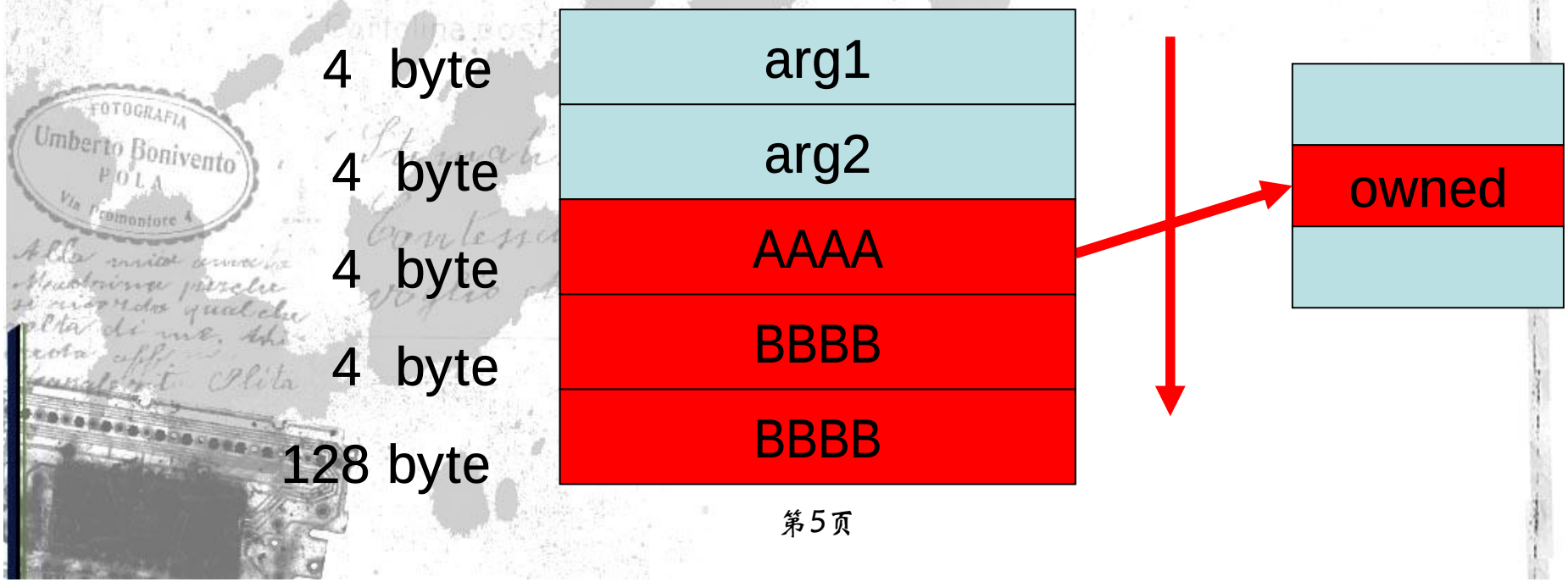
```
void fun(char *arg2,long arg1){
    char buffer[128];
    strcpy(buffer,arg2); /* w/o bounds check */
        return 0;}
```

- X86 栈帧



Shellcode攻击(con.)

- 如果arg=BBB…BAAAA=136 bytes
  - 执行strcpy()
- X86 栈帧



Shellcode攻击(con.)

- Inject vector à 弹头
  - 能够劫持执行流的脆弱点
    - 形式:环境变量/全局参数,文件,协议中参数,socket…一切由外部获取的数据!
    - 本质:Stack Overflow,Heap overflow,Integer overflow,Unsigned/signed,Format string,Calculate error
- Payload à 弹体
  - 完成期望功能的指令序列
    - Shell,virus,wrom,rootkit and anything you can think!

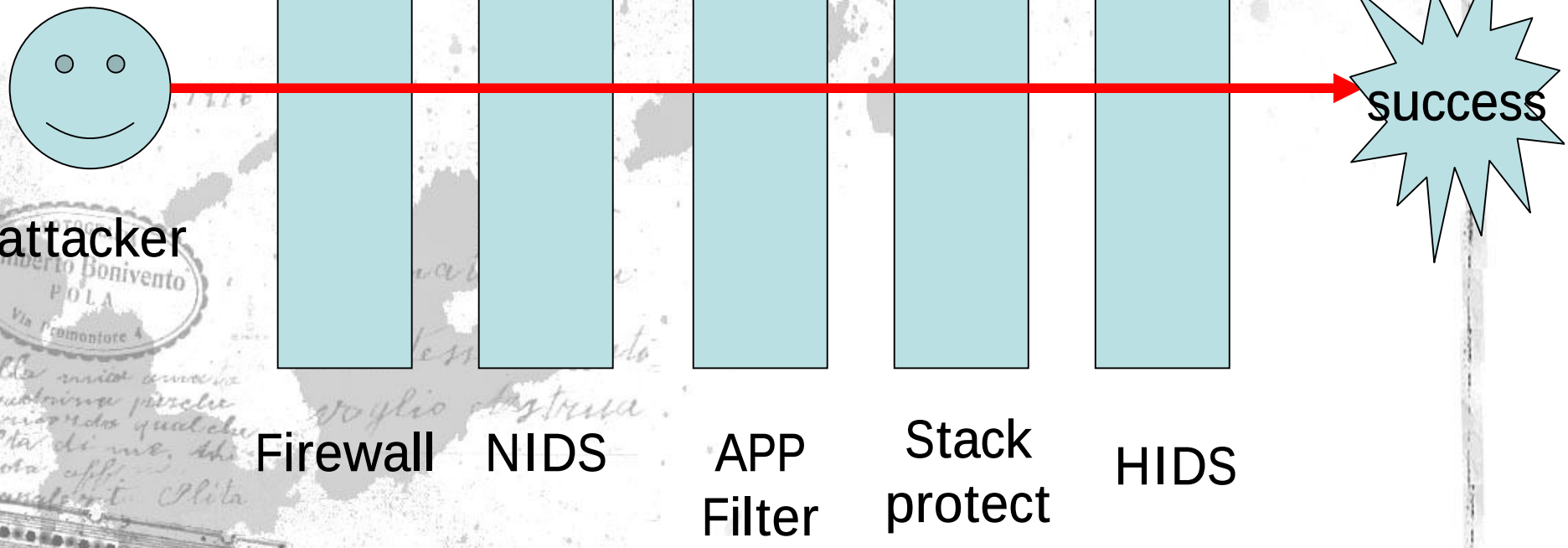
Why learn shellcode technique

- 流行的攻击向量
  - 表现:CERT布告,BUGTRAP
  - 本质:不安全的编码和设计,人为的错误
- 有效的控制方式
  - 能够在进程上下文中执行任意代码
- 但广泛部署的防护措施阻止达成目标
  - Firewall,NIDS,HIDS,OS

Shellcode 攻击思考

- 目标系统可能部署各种防护措施
  - 不能任意存取文件,调用系统调用,执行程序...
- 可使用的进程资源
  - 打开文件句柄,socket,内存中关键数据,可用特权,preloaded函数...
- 创建small shell
  - 建立通信隧道
  - Upload文件和执行命令
  - 为进一步攻击做准备

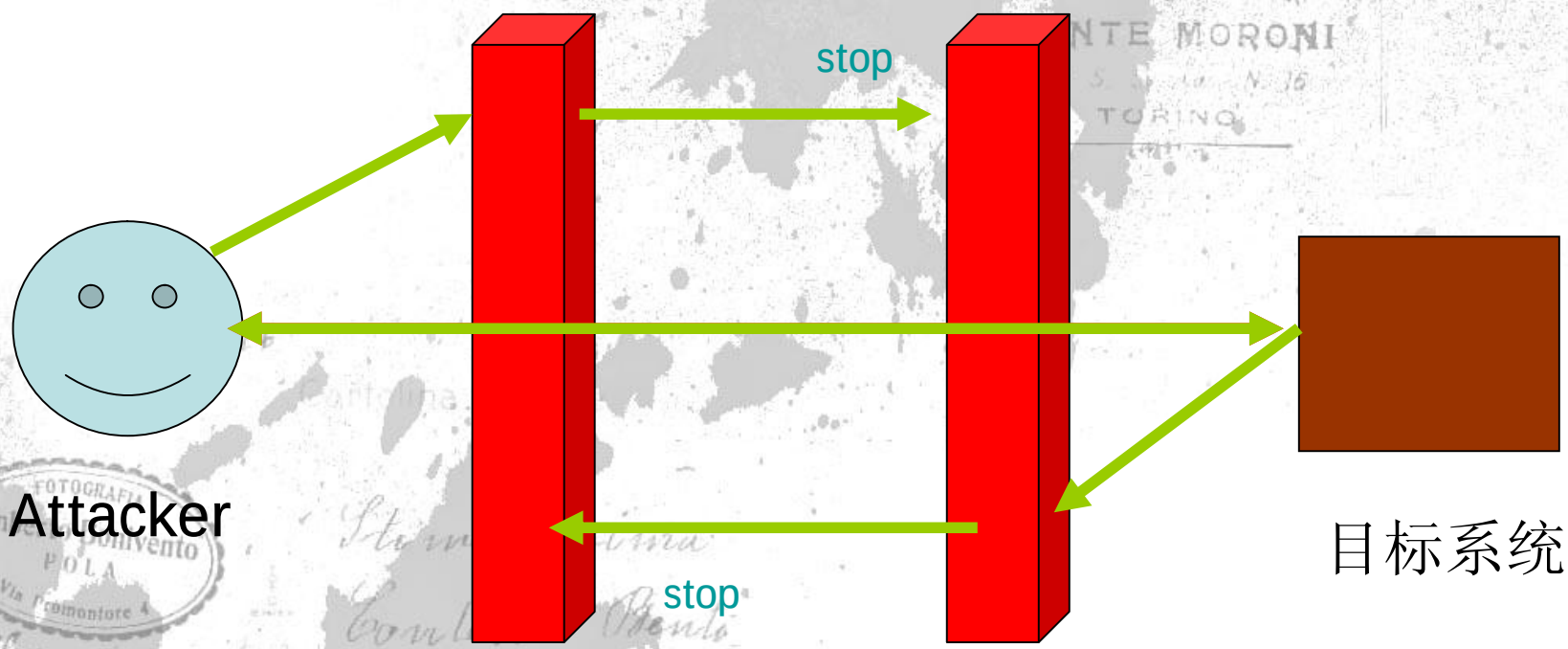
Shellcode 攻击路径



Shellcode 攻击路径:Firewall

- 连接跟踪,NAT
  - 不能使用bind socket方法
- 如果我们在NAT设备后
  - 不能使用反向连接的方法
- 解决方法
  - Find socket,reuse it!

Shellcode攻击路径:Firewall



Shellcode攻击路径:NIDS

- 字符串匹配
  - 不能包含“shell”类似的字符
- Nop区检测,指令模拟
  - 消耗大量CPU时间,不能执行充分的检查
- 解决方法:多态shellcode

Shellcode攻击路径:APP Filter

- Strcpy()禁止NULL
  - 字符串不能包含NULL
  - 不能使用包含NULL的指令
  - 返回地址不能包含NULL字节
- 纯字符过滤与UNICODE变换
- 缓冲区大小的限制
- 解决方法:multi-stage&防范变换的shellcode

Shellcode攻击路径:stack protect

- BoWall
  - 拦截strcpy()
    - 需技巧性覆盖EIP
- Windows 2003
  - Stack cookie
    - 需技巧性覆盖EIP
- OverflowGuard
  - No-exec stack/heap
- 解决方法:SEH&ret-into-lib

Shellcode攻击路径:HIDS

- 应用层HIDS
  - ACL,honeytoken,系统日志
    - 不能任意存取文件,不能触发异常
- 核心层HIDS
  - windows特权，**Entercept**
    - 不能任意调用系统调用/库函数
    - 缓冲区溢出检查
- 解决方法:利用进程中可用资源,如preloaded函数

Shellcode 设计实现

- Firewall: find socket and reuse it.
  - Find accept()返回值

```
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
```

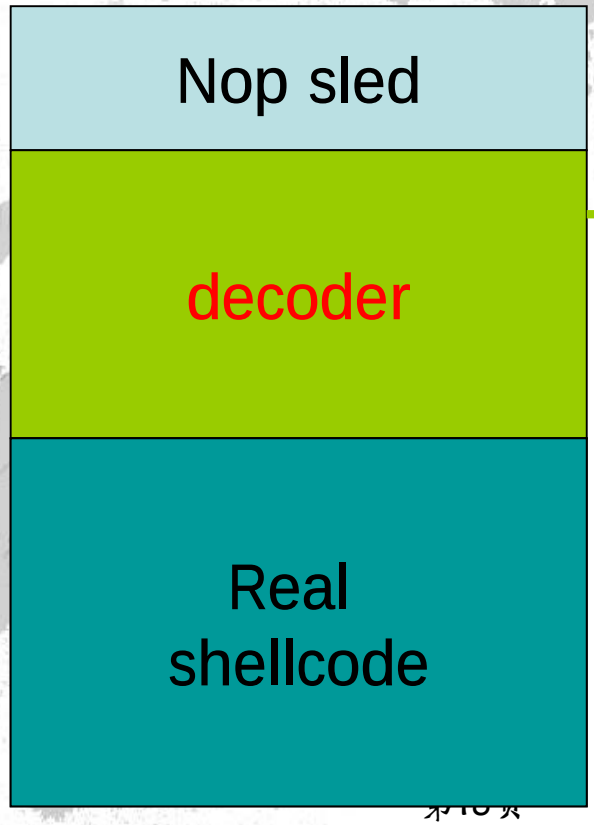
Shellcode 设计实现(con.)

- Firewall:Find socket and reuseit!
  - Find accept()返回值
    - EBP改变:使用socket固定相对偏移

```
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 设计实现(con.)

- NIDS: 多态shellcode



自动工具:ADMmutate

Shellcode 设计实现(con.)

- NIDS: 多态shellcode
  - Nop sled
    - 无负作用单字节指令
    - Nop,inc eax…
  - decoder
    - 相同操作功能的代码替换
    - Push aaaa,pop eax=xor eax,eax,xor eax aaaa=mov eax 0,add eax aaaa=…
  - Real code :xored

Shellcode 设计实现(con.)

- APP Filter: 全字符&Unicode proof shellcode
  - For who you don't know:全字符
    - X86指令格式

    [opcode] [Mode R/M byte] [<SIB>] [<disp8>/<disp32>]
    - 指令各部分只能是[0-9],[A-Z],[a-z]
    - inc ebx /* 0x43 = C */

Shellcode 设计实现(con.)

- APP Filter: 全字符&Unicode proof shellcode
  - For who you don't know:UNICODE
    - UNICODE变换

AABBCCXX=00AA00BB00CC00XX (XX<=0x7f)

- 可使用指令

单字节指令:inc eax /* 0x40 */

双字节指令:add [esi],ch /*0x002E*/

三字节指令:add byte ptr [ebp] ,ch /*0x006d00*/

五字节指令:add eax ,4C007500h /*0x050075004C */

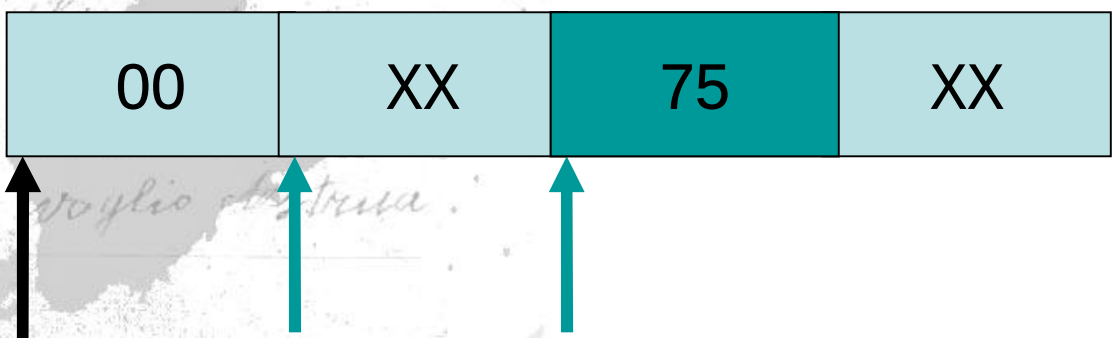
Shellcode 设计实现(con.)

- APP Filter: 全字符&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 设计实现(con.)

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

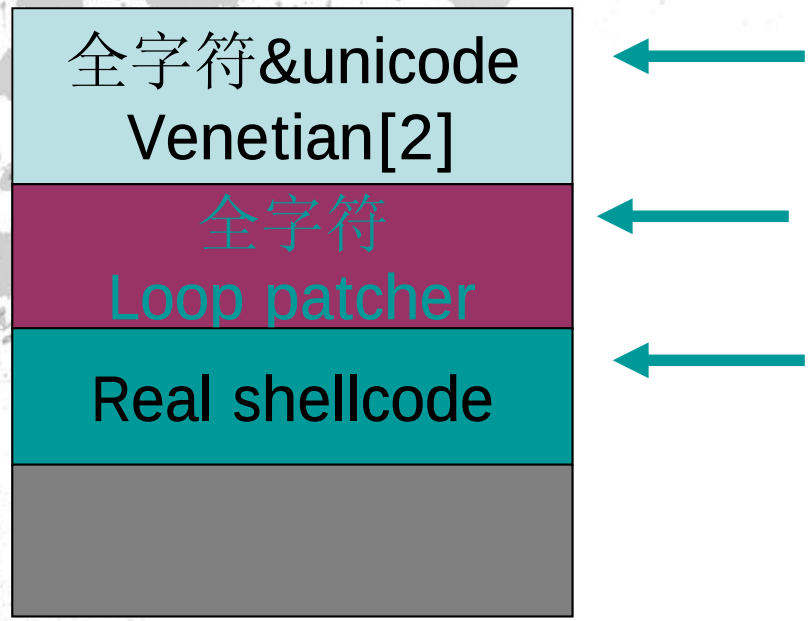


Shellcode 设计实现(con.)

- APP Filter: 全字符&Unicode proof shellcode
  - For who you don't know:UNICODE
    - Venetian[1]存在的问题
      - 缓冲区大小的限制问题
        - » 14字节设置两字节指令
      - 没处理全字符过滤

Shellcode 设计实现(con.)

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



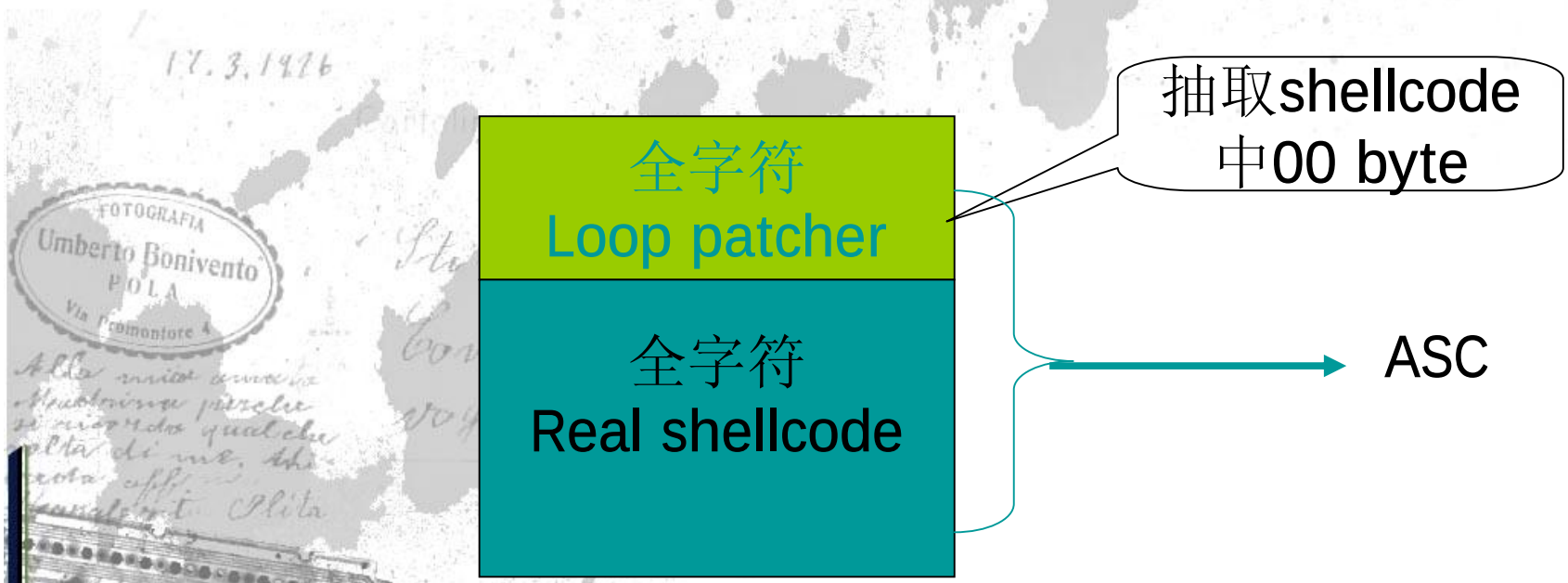
Shellcode 设计实现(con.)

- APP Filter: 全字符&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*/
```

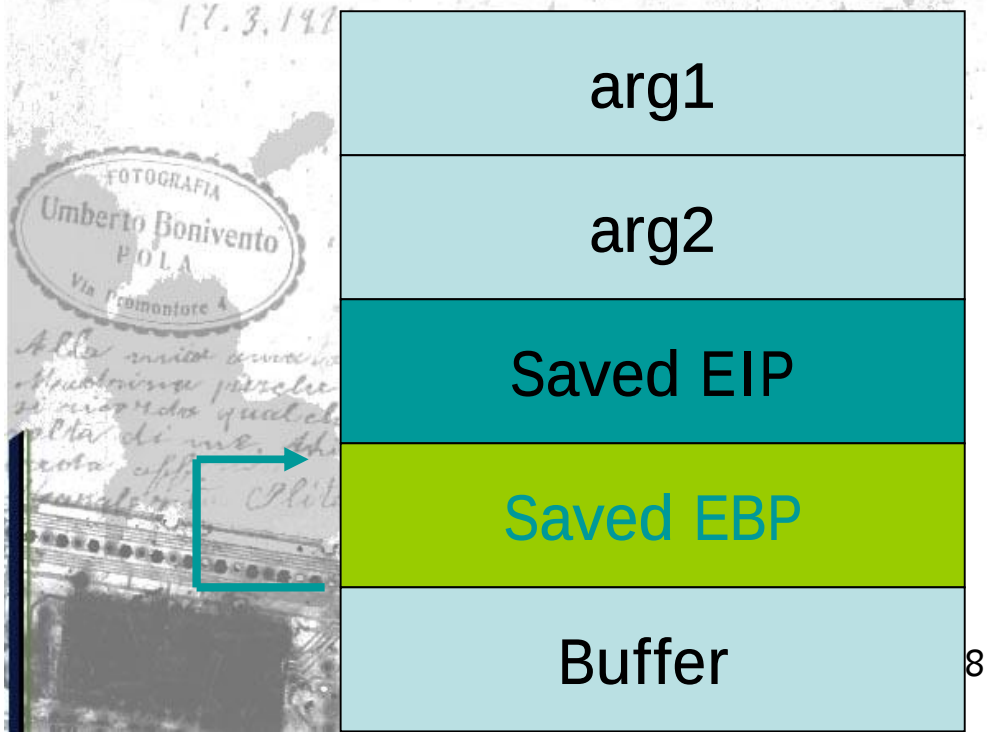
Shellcode 设计实现(con.)

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



Shellcode 设计实现(con.)

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



Saved Ret=[EBP+4]

↓

Strcpy()

↓

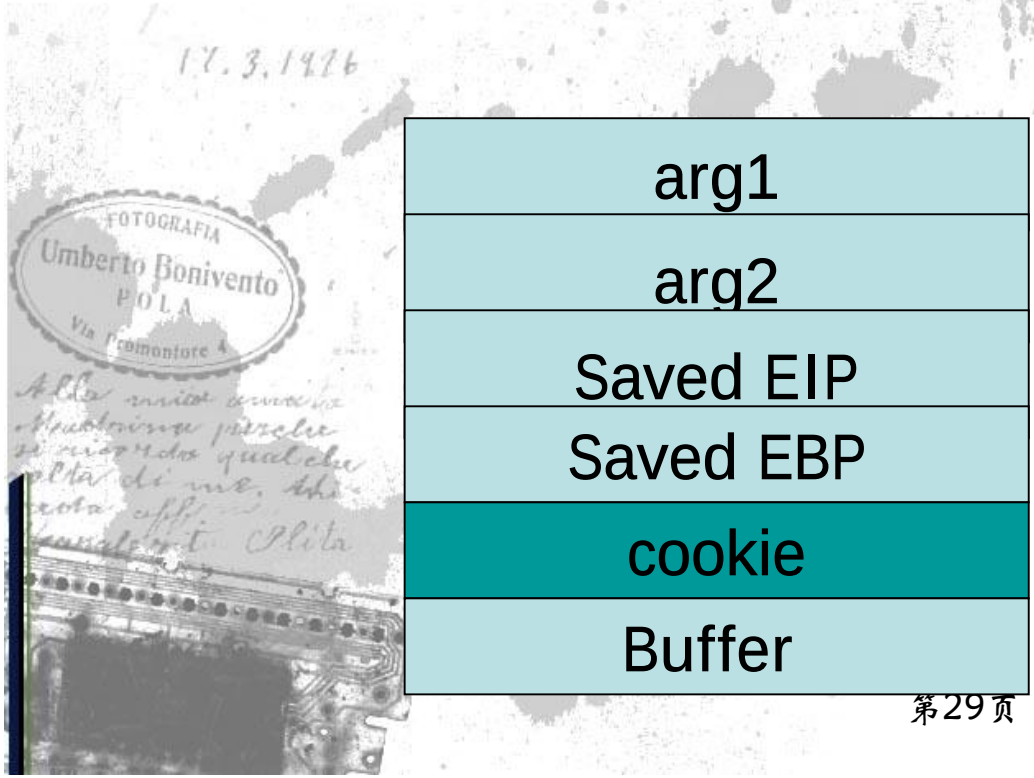
Get ret

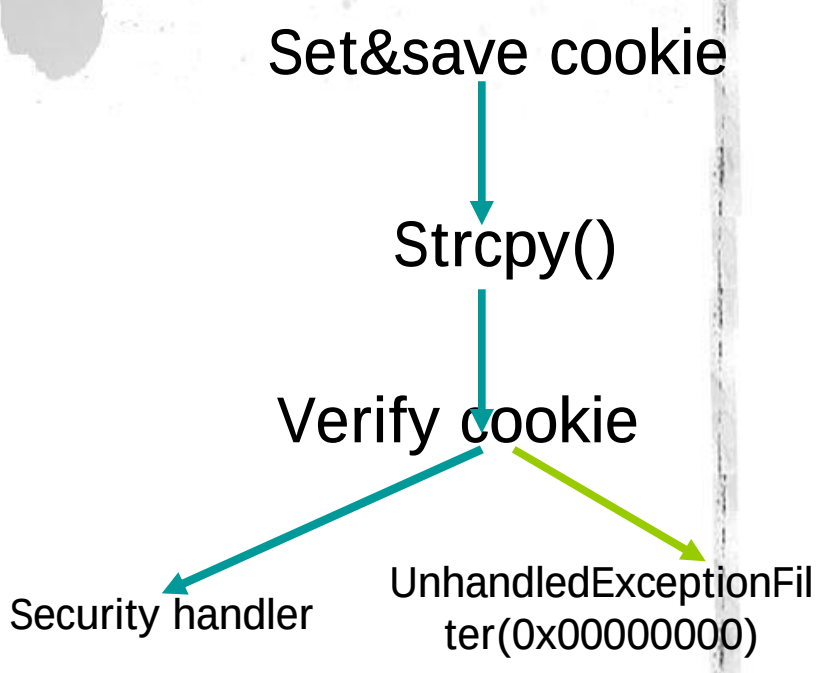
↓

Verify

Shellcode 设计实现(con.)

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



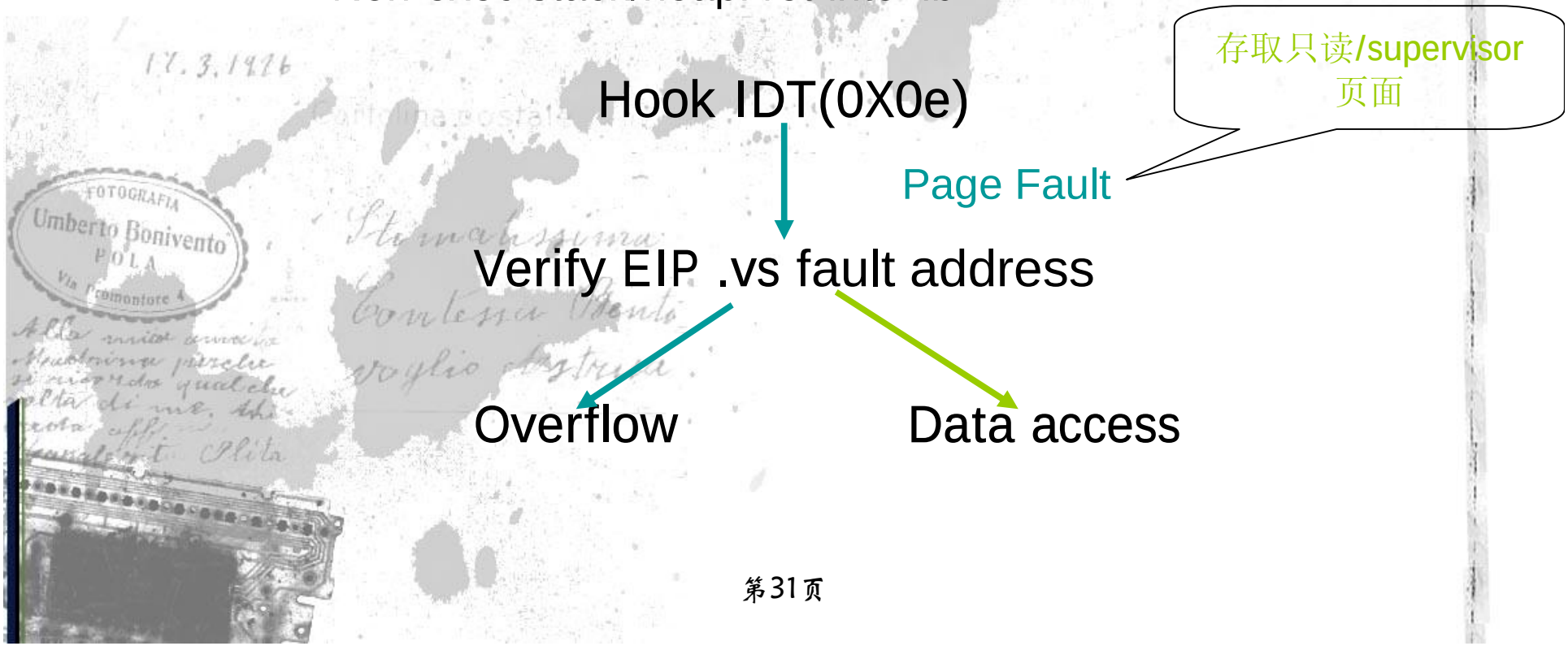


Shellcode 设计实现(con.)

- Stack protect:SEH&ret-into-lib
  - For who you don't know:windows 2003
    - SEH
      - 清零所有关键寄存器
        - » 不能使用:call eax,jump eax…
      - **Exception handler**在load configuration保存副本
        - » 简单覆盖**Exception handler**不再有效
        - » 可利用已注册 **Exception handler**
      - load configruation没匹配项,**Exception handler**在模块地址也将得到执行
        - » 以模块地址范围外的指令块覆盖 **Exception handler**
      - **Exception handler**不能指向**stack**地址,可指向**heap**
        - » 以heap地址内指令块覆盖 **Exception handler**

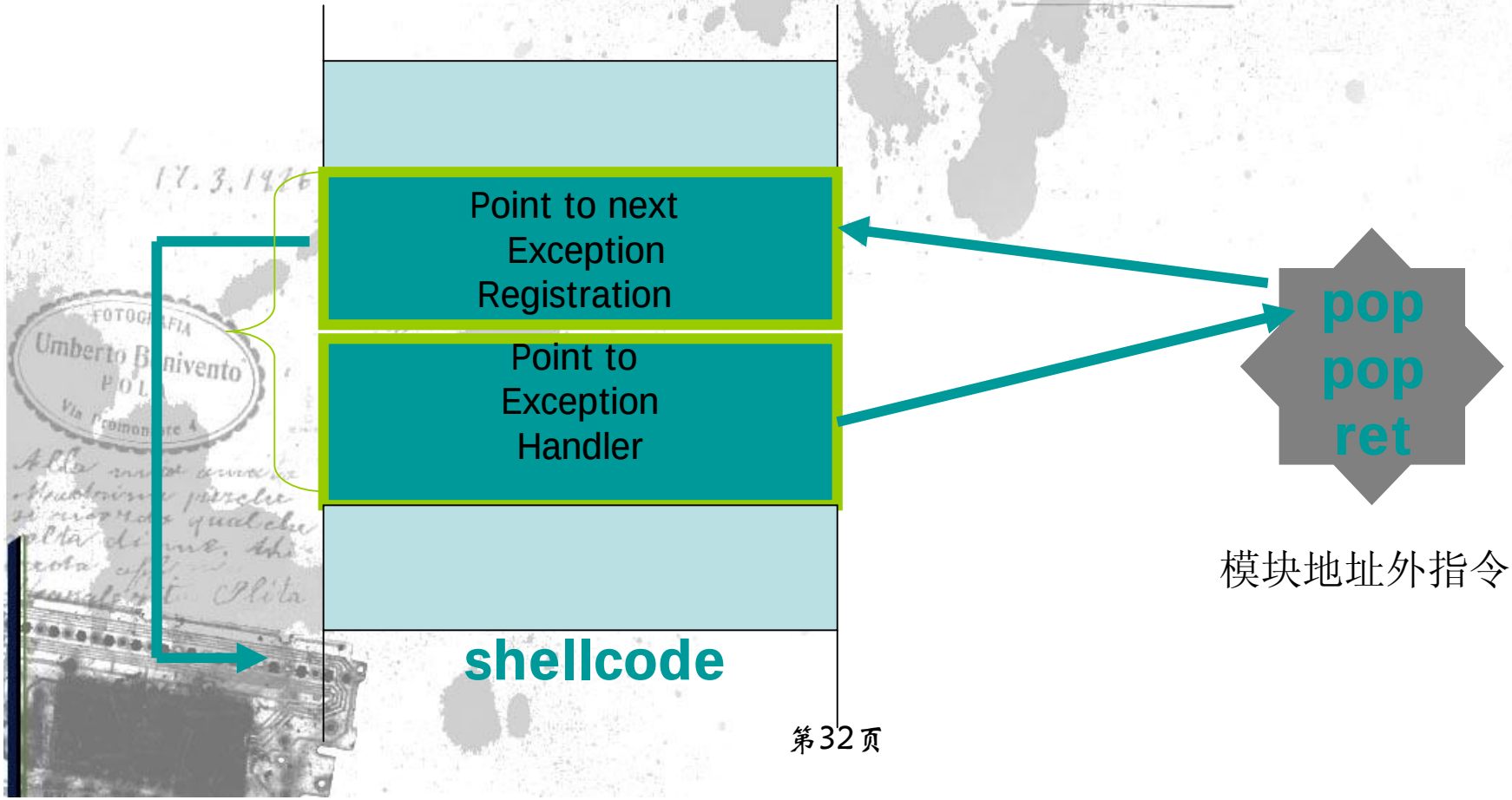
Shellcode 设计实现(con.)

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



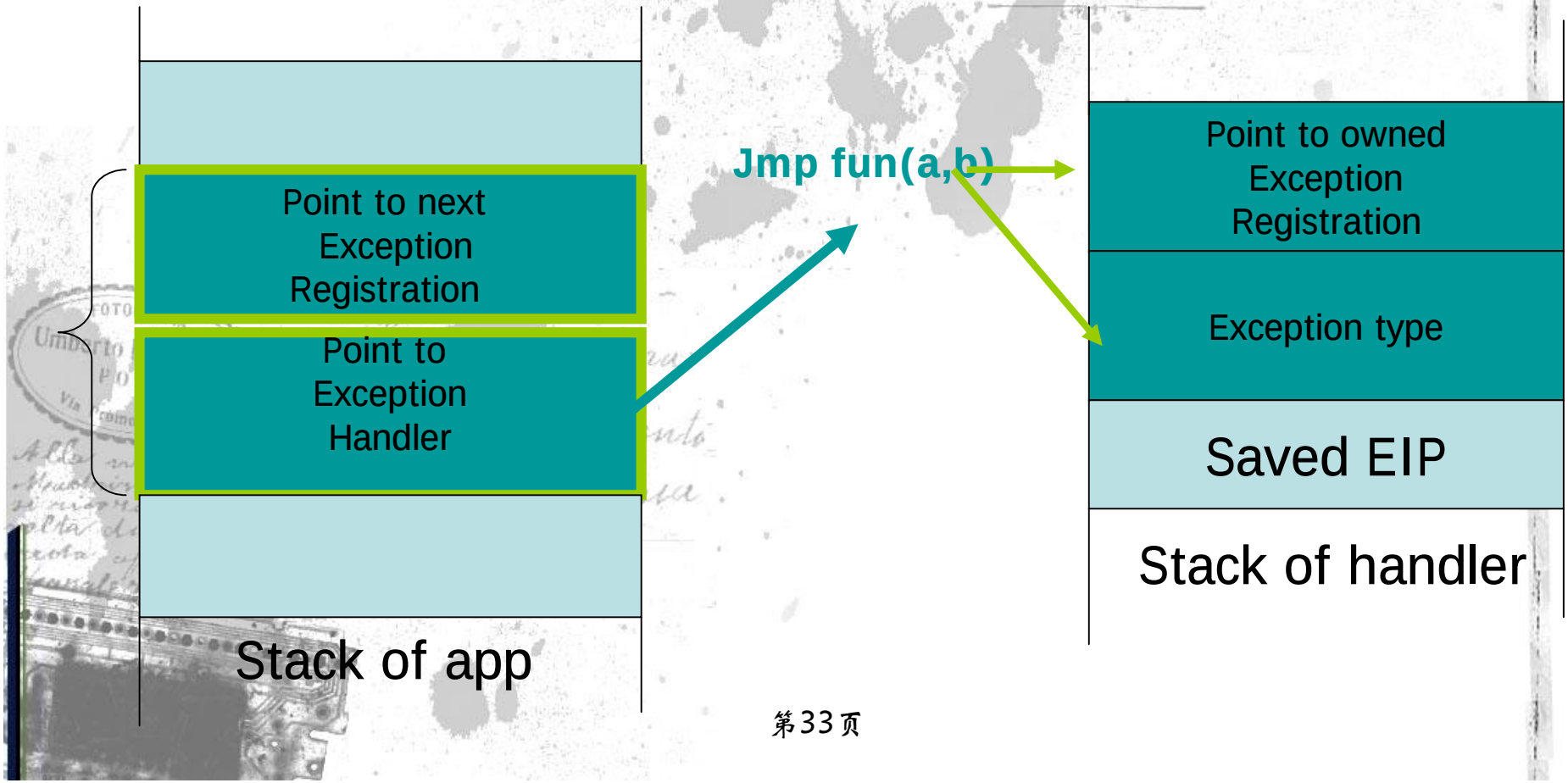
Shellcode 设计实现(con.)

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



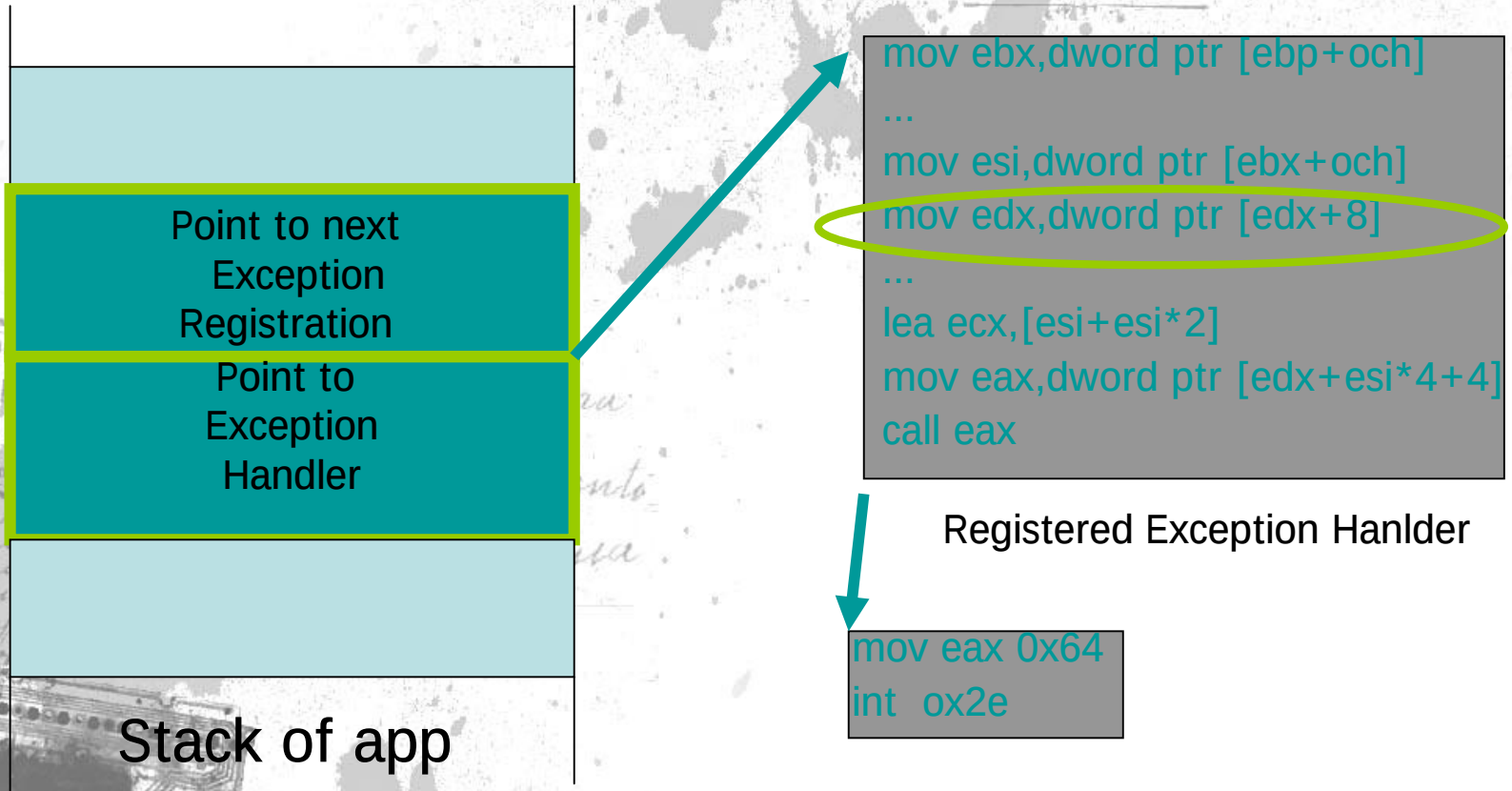
Shellcode 设计实现(con.)

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



Shellcode 设计实现(con.)

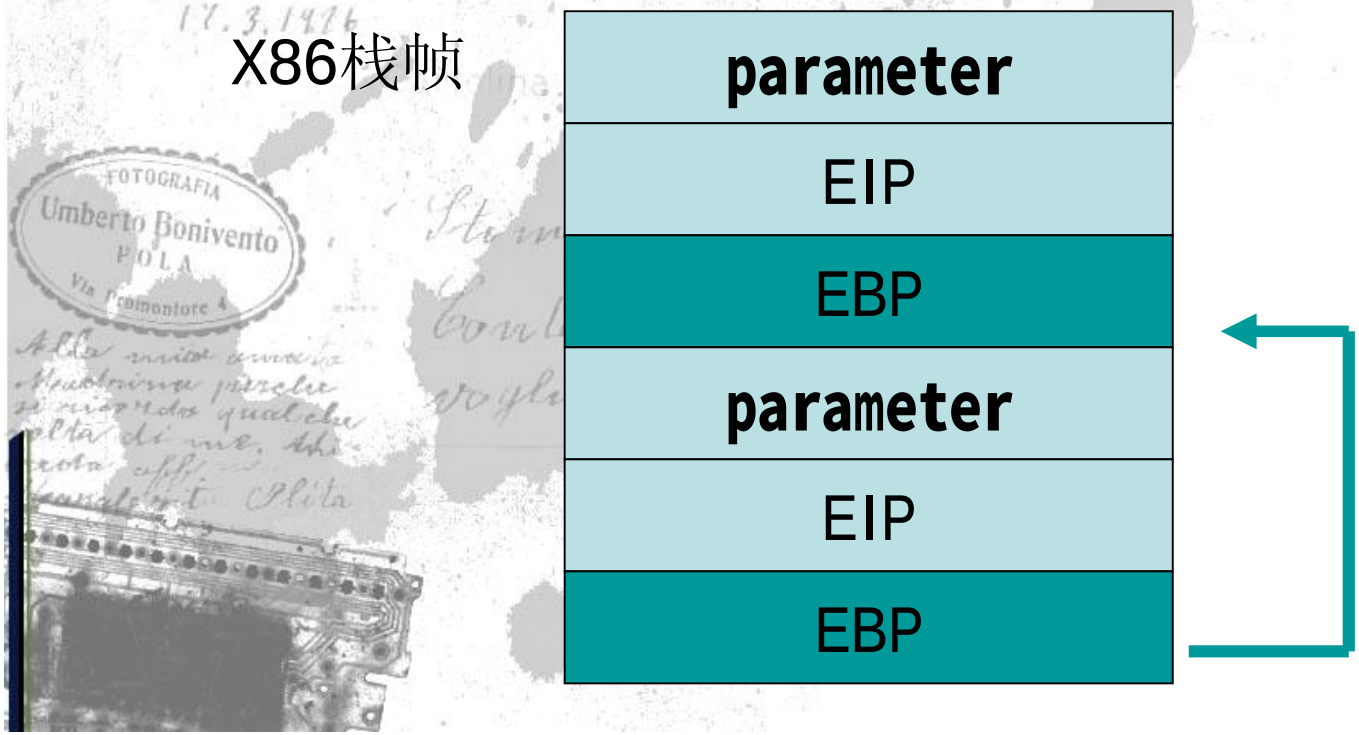
- Stack protect:Win2003&BoWall w/ non-exec



Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - For who you don't know: Entercept
    - Hooked API: fake frame

X86栈帧



Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - 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三要素:

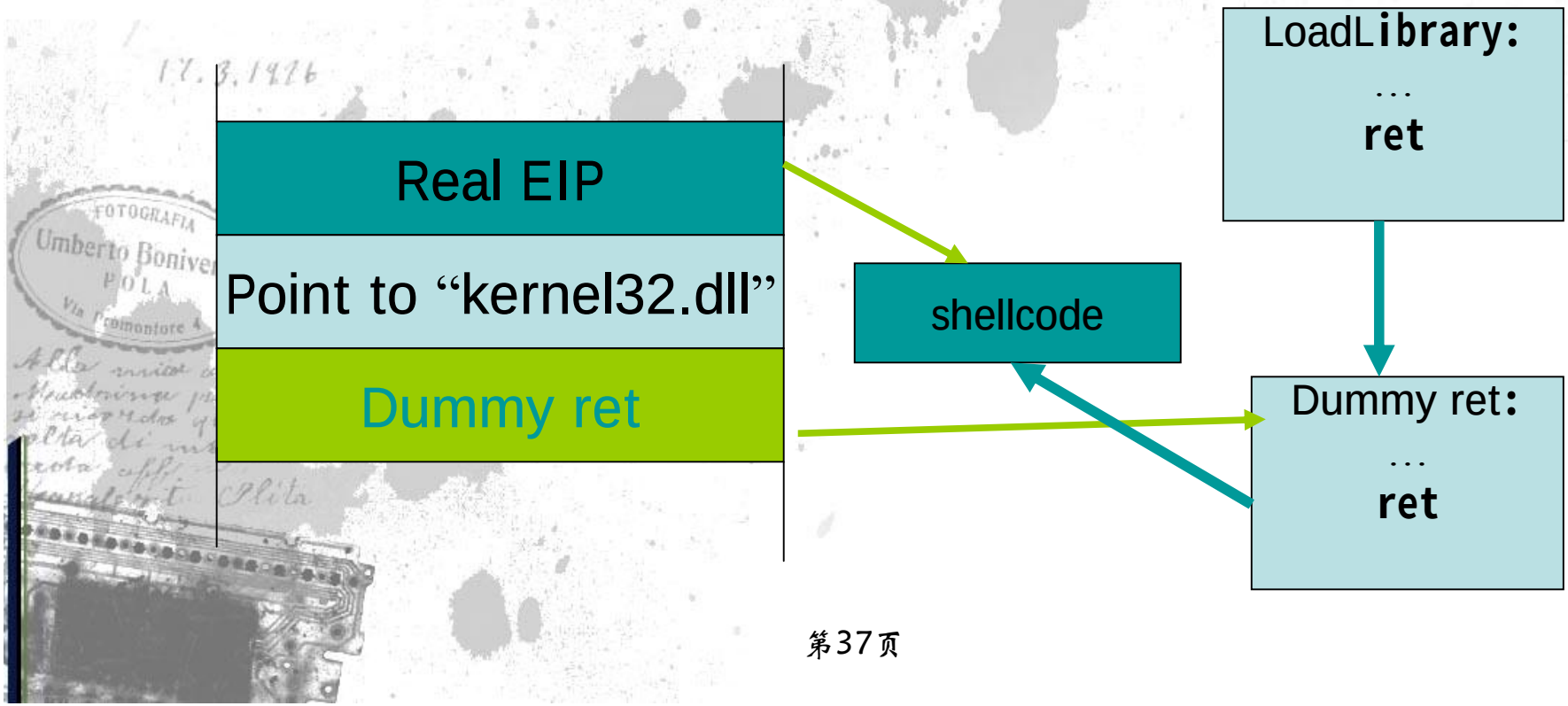
只读页面许可

属于内存映射

在jmp/call后

Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - For who you don't know: Entercept
    - Hooked API: fake frame



Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - For who you don't know: IAT
    - 分析IAT

PE offset

Optional Header

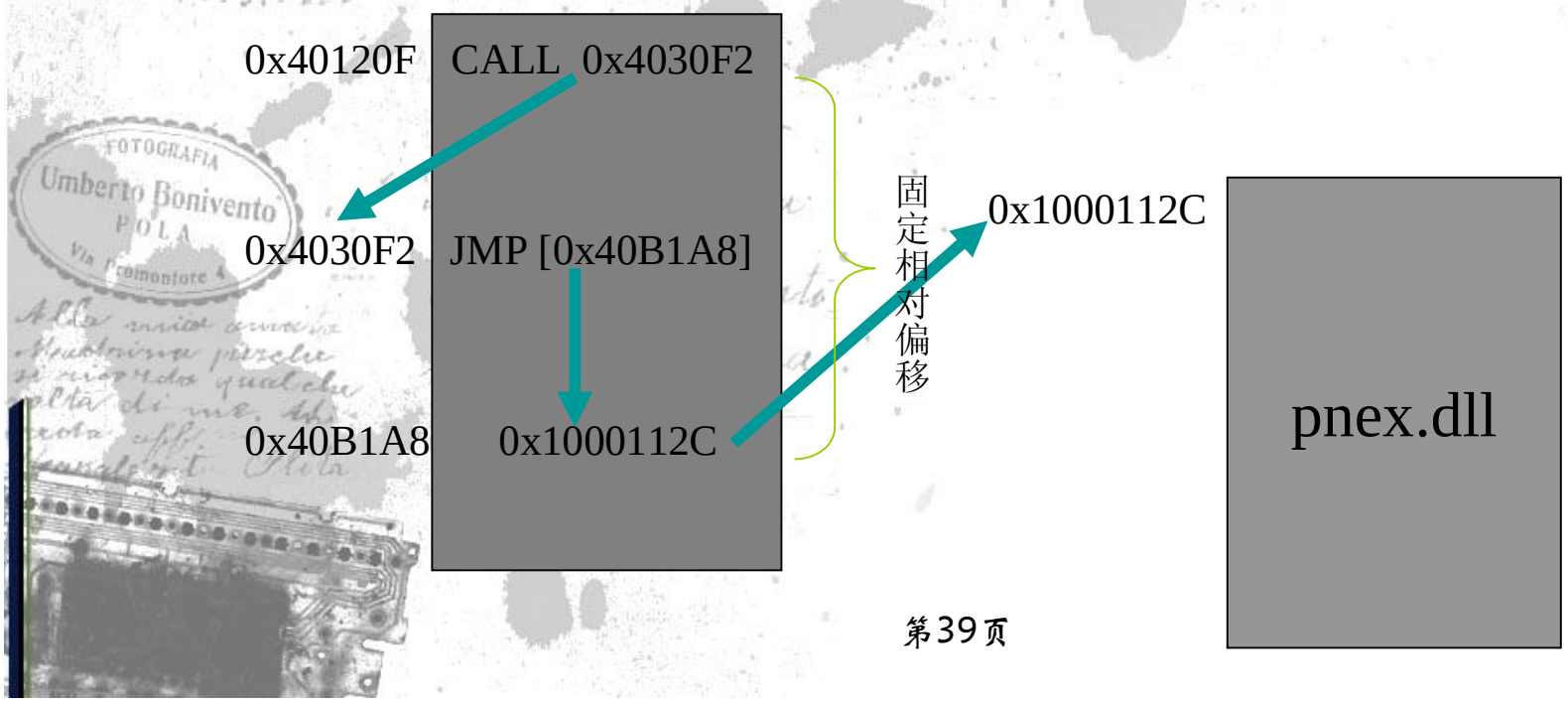
Ascii name

Address

shellcode中使用函数Ascii name的CRC

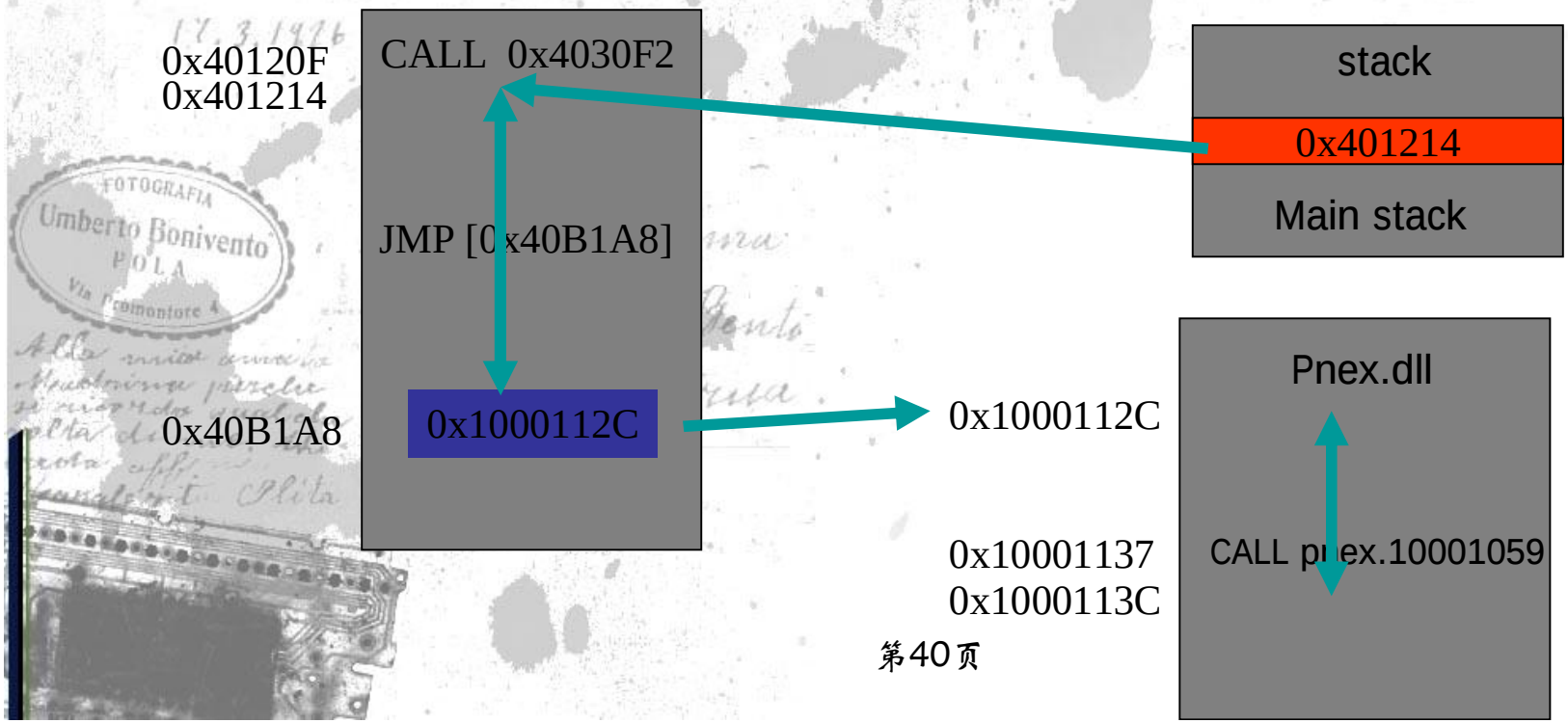
Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - For who you don't know: 计算返回地址
    - 分析IAT



Shellcode 设计实现(con.)

- HIDS:fake frame&IAT&计算返回地址
  - For who you don't know: 计算返回地址
    - 分析IAT



Demo

- 实现一部分讨论特征例子
- 脆弱服务器&sploits

Thank You

Question ?

X'CON 2004