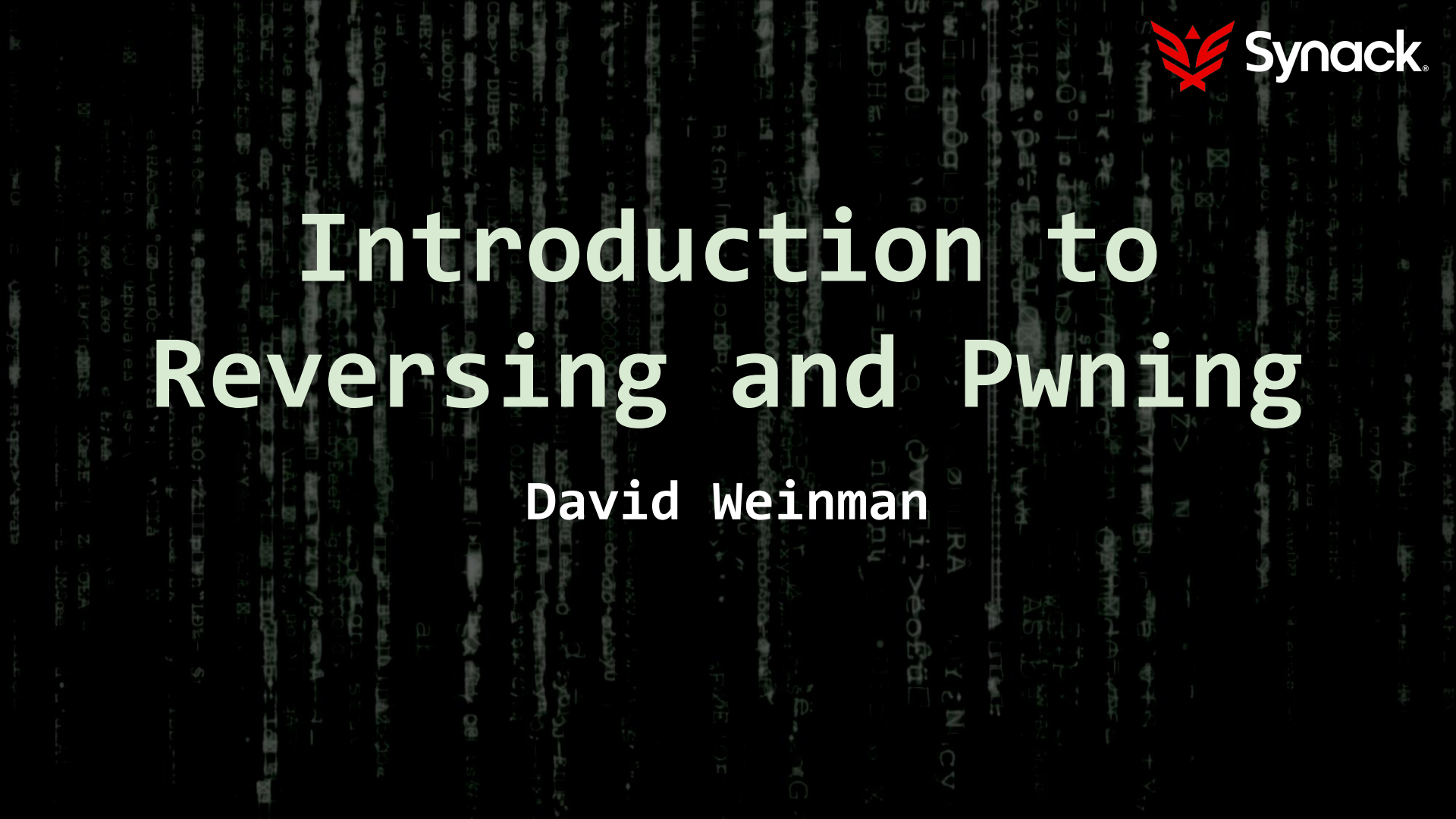
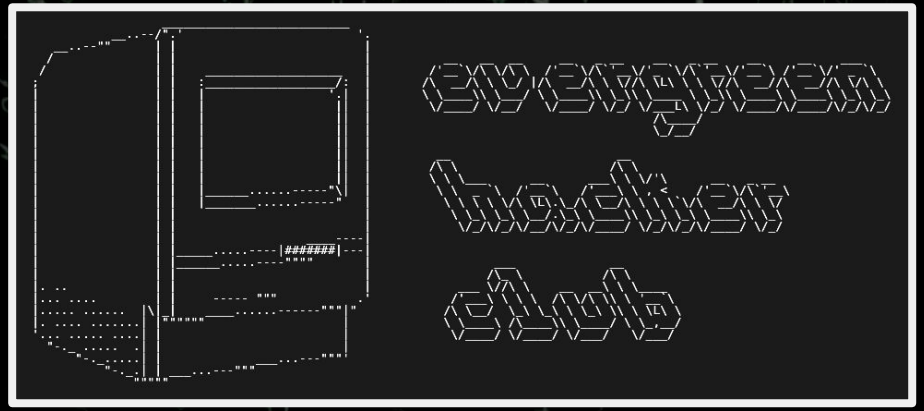

Synack
Introduction to
Reversing and Pwning
David Weinman

whoami - @h3ll_d0g

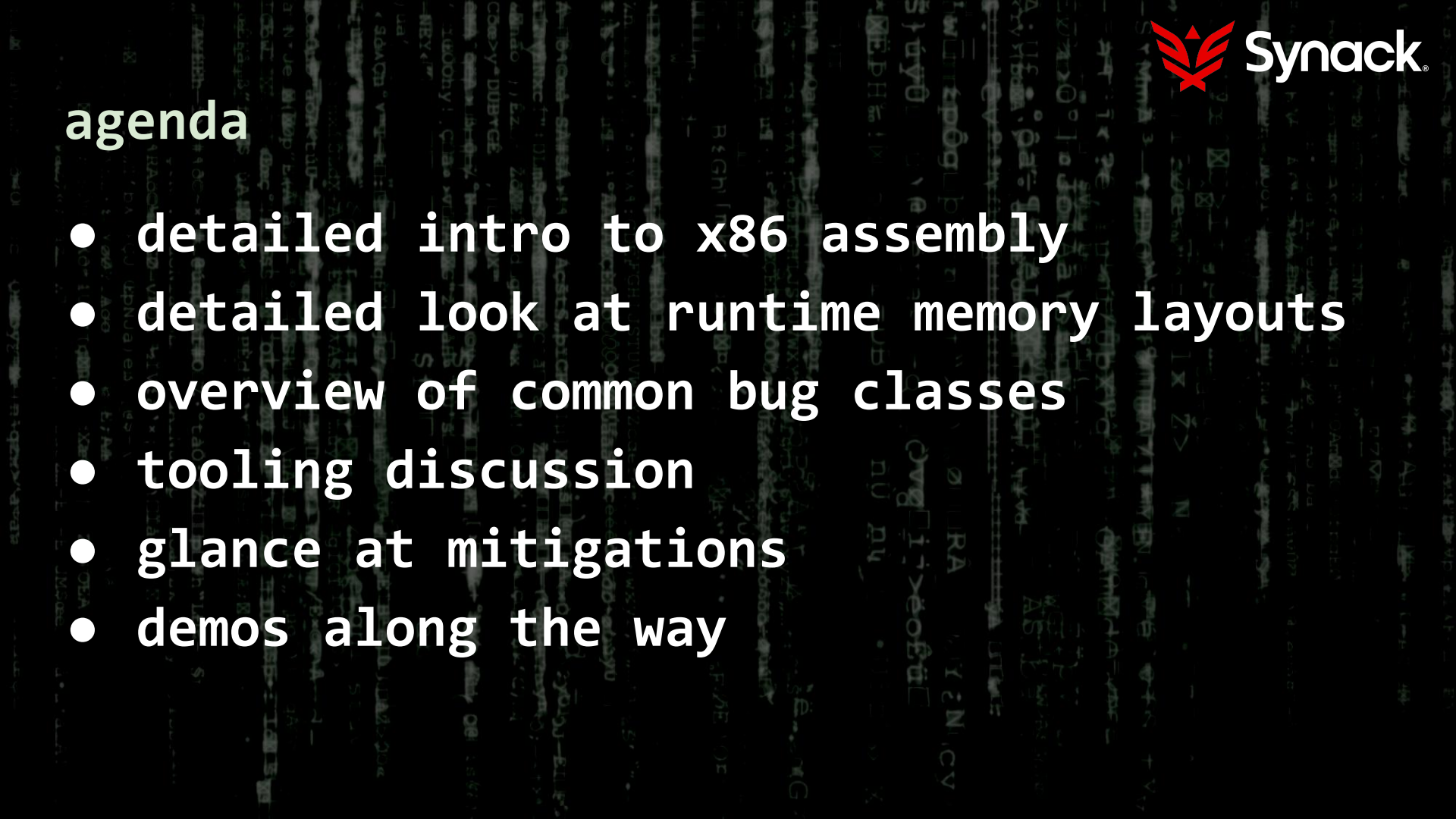
- security research engineer at synack, TESC alum
- member of the opentoall ctf team
- hippie millennial loves the pacific northwest
- snowboarder, skateboarder, metal head



i’m still learning

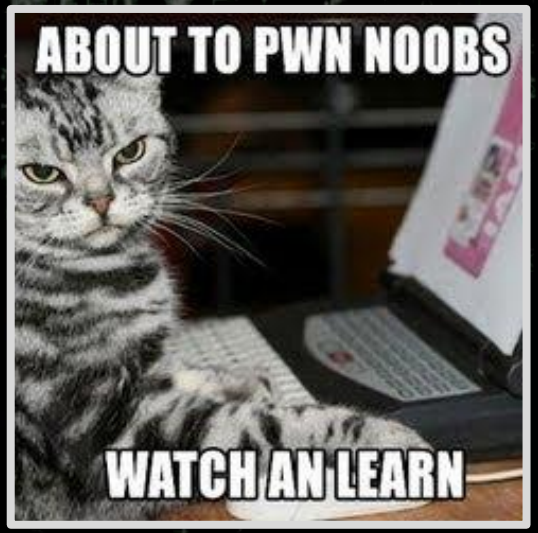
why pwn or reverse?

- $
  - amount? not all bugs are equal, not all bounties focus on the same kinds of targets
    - synack has a mobile app bounty - can use related skills
    - android bugs <= $4k (google)
    - in 2015, average android bug payout was $2.2k (google)
- fun! pwn challenges can be super rewarding to solve

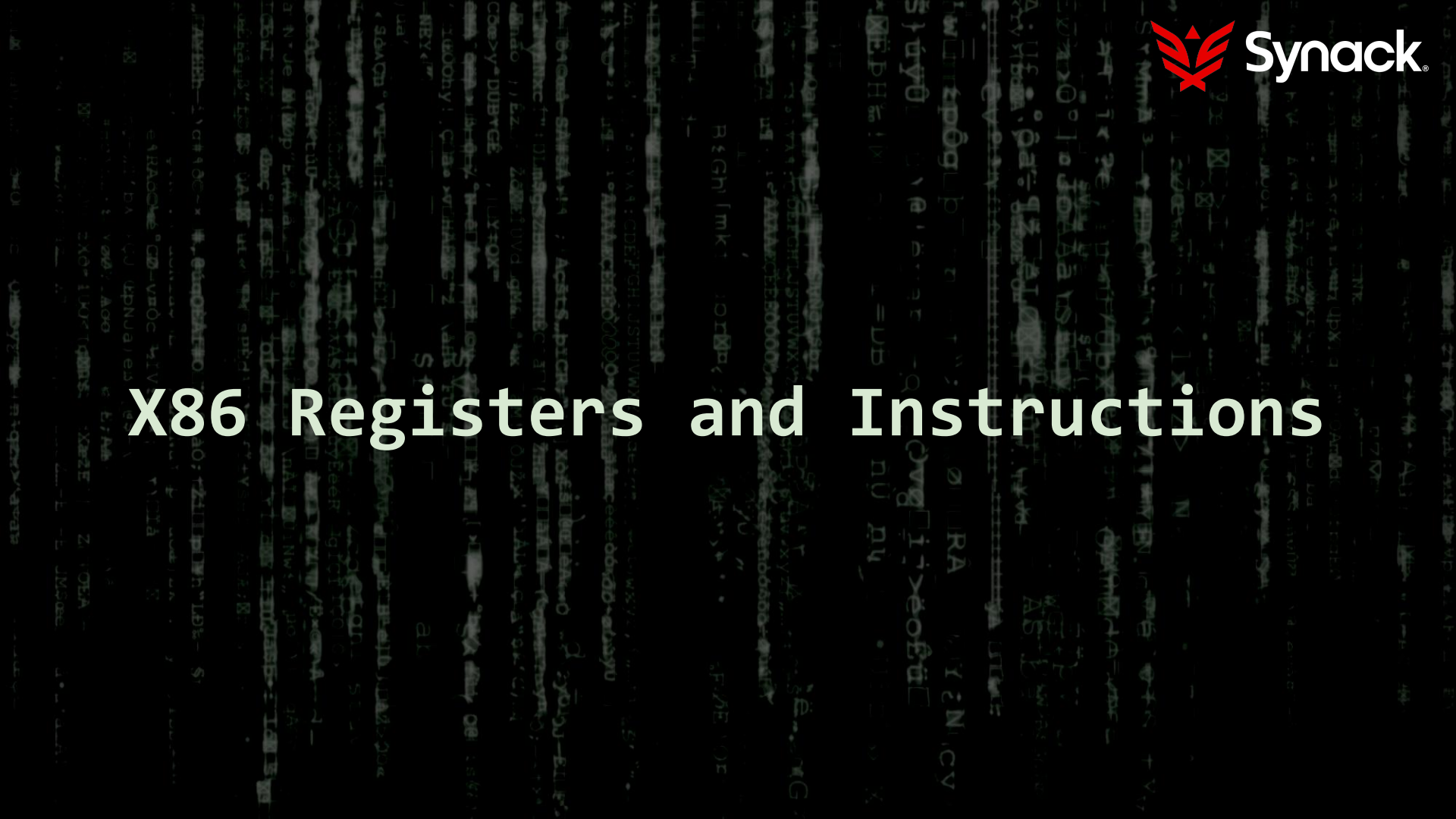

Synack
agenda
• detailed intro to x86 assembly
• detailed look at runtime memory layouts
• overview of common bug classes
• tooling discussion
• glance at mitigations
• demos along the way

takeaway

- how a program is compiled and run
- common bug classes/exploit mitigations
- ability to decompile c programs
- techniques for binary analysis
- exposure to a pwn challenge

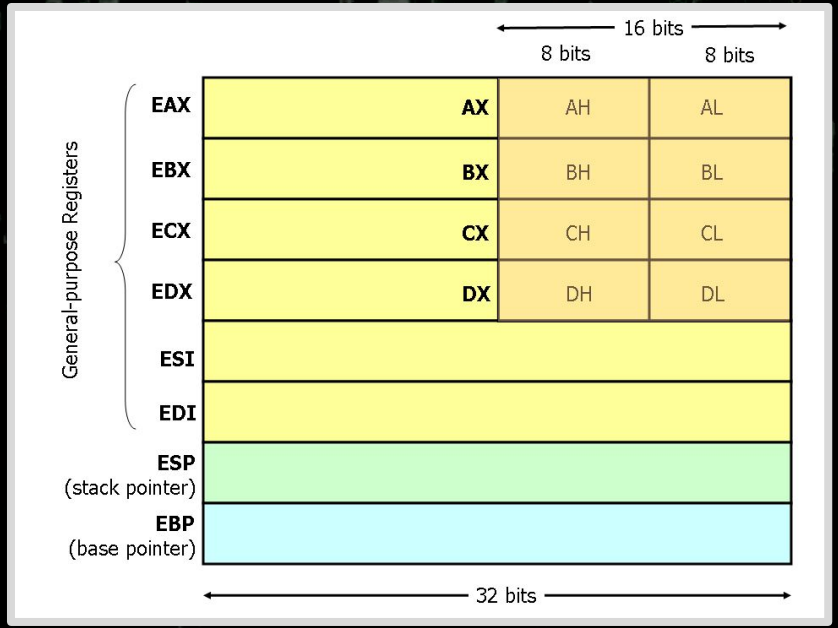



Synack
Demo time


Synack
X86 Registers and Instructions

X86 / peek inside a CPU

- x86 arch register: 32 bit memory chunk
- can refer to 16 bit/8 bit subsets



X86 / assembly language

- architecture specific - converted to machine code by assembler
- there are two syntax flavors of x86 assembly, intel and at&t
- intel syntax: operand destination, source

```
mov eax, 5
```

```
int square(int num) {
    return num * num;
}
```

```
square(int):
        push    ebp
        mov     ebp, esp
        mov     eax, DWORD PTR [ebp+8]
        imul    eax, DWORD PTR [ebp+8]
        pop     ebp
        ret
```

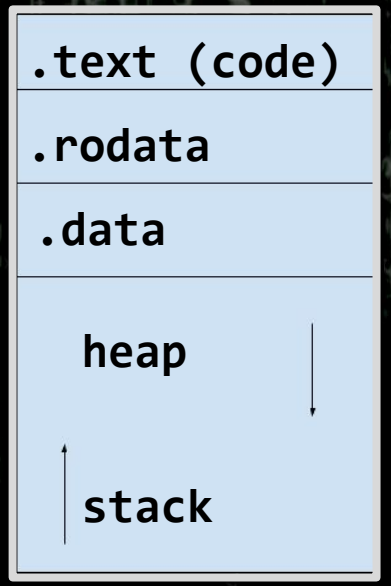

Synack
Runtime

runtime / heap vs stack

- heap is for allocated data usually of variable size, accessible to threads and shared libraries (malloc/free)
- stack is for local variables/arguments, environment variables and function call metadata

low addresses

elf memory layout ➤

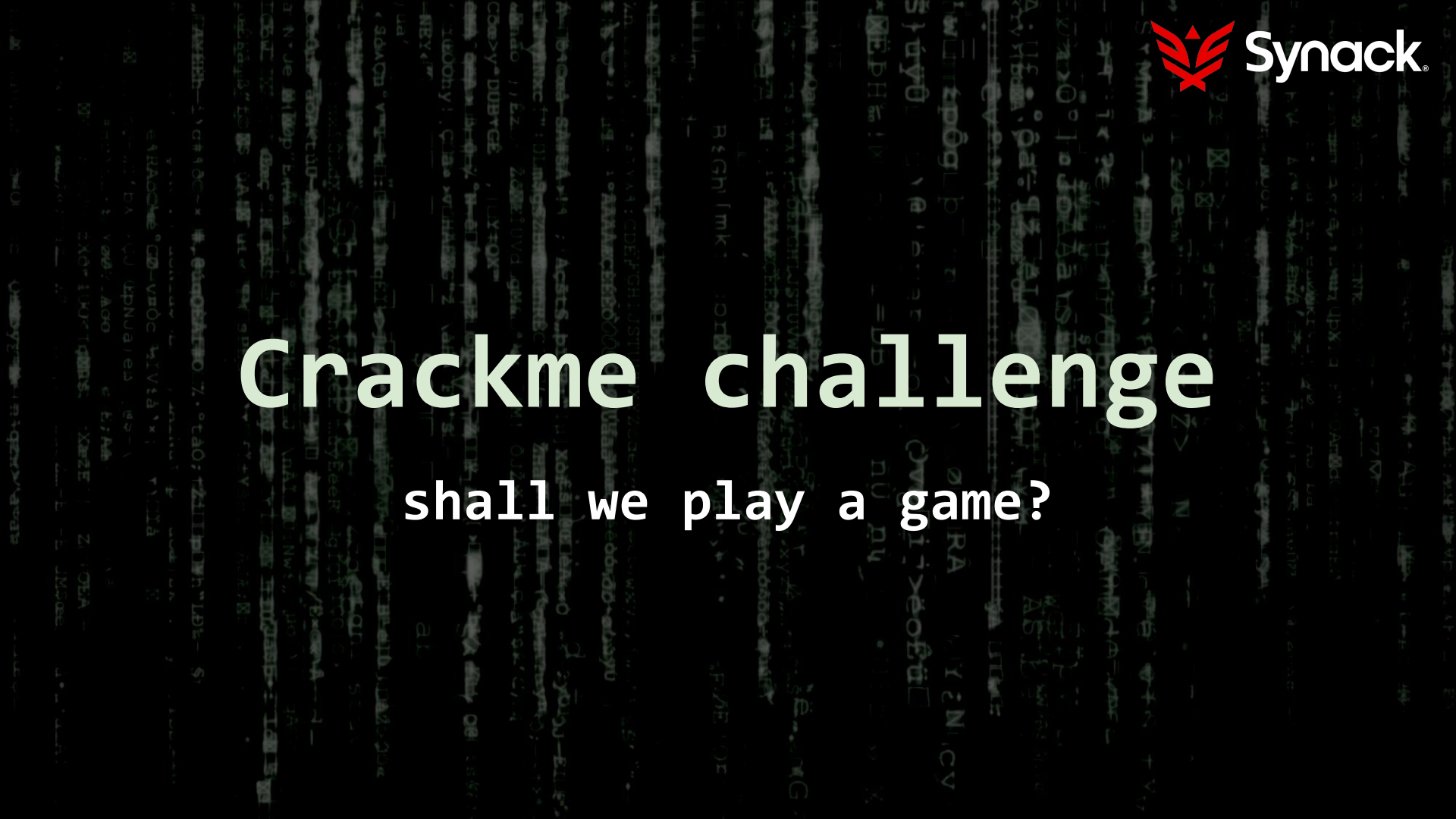


high addresses

runtime / stack frame layout

- caller pushes args and eip onto the stack
- callee pushes ebp and local vars onto the stack

```
void function(int arg1, int arg2) {
  int var1;
  int var2;
}
void main() {
  function(1, 2);
}
```


Synack
Crackme challenge
shall we play a game?


Synack
Bugs

bug classes / stack buffer overflow

```
int main() {
    int  i;
    char buf[10];
    i = 0xd3ad;
    /* out of bounds local variable data can
       overwrite other vars/function metadata */
    fgets(buf, 0x10, stdin);
    printf("k you wrote %s\nnow ... ", buf);
    fflush(stdout);
    if (0 == (0x1337 ^ i)) {
        runme();
    } else {
        puts("eat my shorts!\n");
    }
}
```

low addrs

high addrs

bug classes / format string

```
void main() {
    int i = 0;
    char buf[64];
    unsigned int iptr = (unsigned int) &i;
    disable_buffering(stdout);
    /* printf takes a variable number of arguments
       how is the variable known at runtime? */
    printf("format string playground, enter buf: ");
    fgets(buf, 64, stdin);
    printf("check out the buf you entered: ");
    printf(buf);
    if (i) {
        printf("congratz u win!\n");
        system("/bin/bash");
    }
}
```

```
> ./formatstring
format string playground, enter buf: %08x.%08x.%08x
check out the buf you entered: 00000040.f774f5a0.080482ba
```

bug classes / format string

```
› ./formatstring
format string playground, enter buf: %08x.%08x.%08x
check out the buf you entered: 00000040.f774f5a0.080482ba
```

```
format string playground, enter buf: %3$08x
check out the buf you entered: 080482ba
```

bug classes / wild copy

- the size of a copy is under limited attacker control
- the copy is large enough to cause a fault prior to completing

```
struct message {
    char buf[24];
    int len;
    void (* result)();
};

void lose() {
    puts("loser\n");
}

void win() {
    system("/bin/bash");
}
```

```
void main() {
    struct message *msg;
    disable_buffering(stdout);
    msg = malloc(sizeof(struct message));
    msg->len = 24;
    msg->result = lose;
    while (msg->len >= 24) {
        printf("What is your length? ");
        scanf("%d", &msg->len);
        getc(stdin); // eat up newline
    }
    printf("OK, what is your buf? ");
    read(0, msg->buf, msg->len);
    msg->result();
    free(msg);
}
```

bug classes / use after free

- malloc chunk is freed but pointer to the chunk is reused
- if sizes are similar malloc may reuse chunks after frees

```
mallocing a chunk
malloced chunk at: 0x09269410
freeing chunk
mallocing chunk again
malloced chunk at: 0x09269410
```

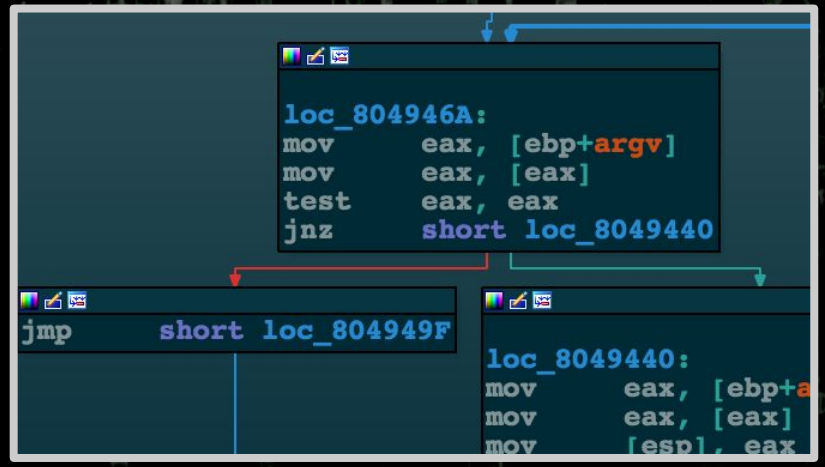
```c
#define SZ1 20
#define SZ2 15
void main() {
    unsigned int *ptr;
    printf("mallocing a chunk\n");
    ptr = malloc(SZ1);
    printf("malloced chunk at: 0x%08x\n", (unsigned int)ptr);
    printf("freeing chunk\n");
    free(ptr);
    printf("mallocing chunk again\n");
    ptr = malloc(SZ2);
    printf("malloced chunk at: 0x%08x\n", (unsigned int)ptr);
    free(ptr);
}
```


Synack
Tools of the trade

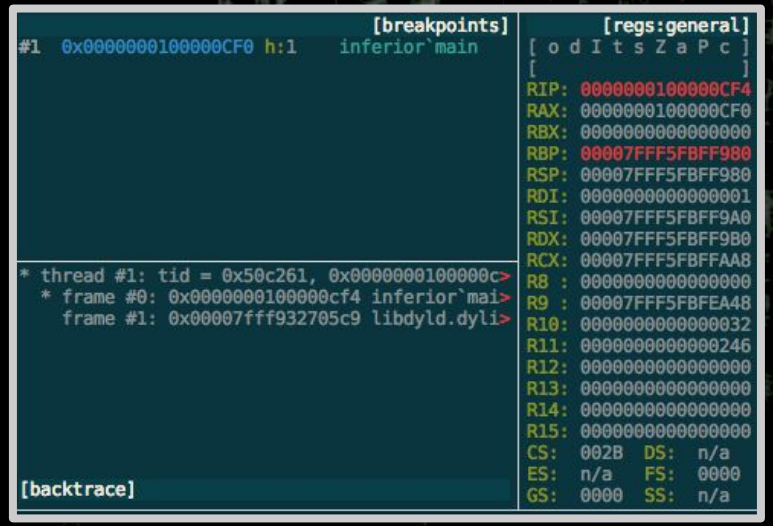
pwn tools / static & dynamic

- static analysis - looking at the binary, its associated shared libraries, anything to do with reversing the binary on disk
- dynamic analysis - debugging the process, memory dumping, anything to do with reversing the binary at runtime

ida ➤



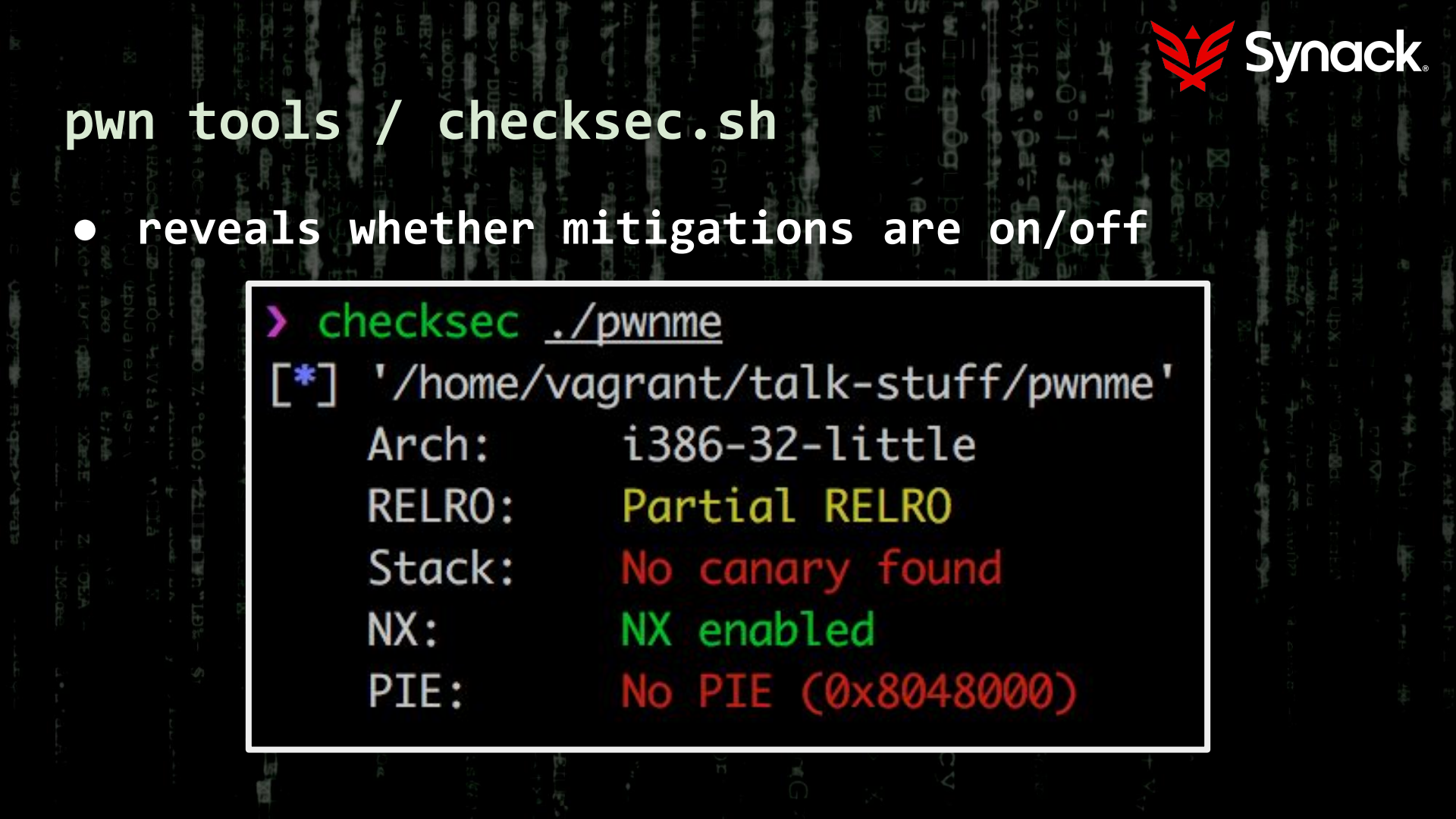
gdb-voltron ➤



pwn tools / binary hacking apis

- pwntools (github.com/Gallopsled/pwntools)

```
[+] Starting local process '/tmp/pwn-a
>>> io.sendline("id")
>>> io.sendline("exit")
>>> print io.recvall()
[x] Recieving all data
[x] Recieving all data: 0B
[x] Recieving all data: 60B
[+] Recieving all data: Done (60B)
[*] Process '/tmp/pwn-asm-LZj2YO/step3
uid=1000(zerocool) gid=1000(zerocool)
```


Synack
pwn tools / checksec.sh
• reveals whether mitigations are on/off
› checksec ./pwnme
[*] '/home/vagrant/talk-stuff/pwnme'
Arch: i386-32-little
RELRO: Partial RELRO
Stack: No canary found
NX: NX enabled
PIE: No PIE (0x8048000)


Synack
Wat do

wat do / shall we play one last game?

- # prereqs - virtualbox & vagrant
- git clone https://github.com/clampz/pwnvm
- vagrant up --provider=virtualbox
- vagrant ssh
- cd challs/intro-examples
- # pwn bins ~/challs/

wat do / bypassing mitigations


Synack
Pwn challenge

pwn3d

wat do / resources to learn more

- youtube.com/user/GynvaelEN/videos
- liveoverflow.com
- github.com/RPISEC/MBE
- challenges.re
- pwnable.tw
- a bug hunter’s diary

gr33tz

- synack for sponsoring this talk
- all students involved with RPISEC MBE, especially d00m
- OpenToAll grazfather & uafio for feedback and helping me
- members of cRUcible for inspiration, support, CTFs
- mike_pizza & rweiss for sparking my interest in pwning
- Miguel Gordo Garcia, Robert Musser & inkrypto for testing challs and slides
- Richo Butts for listening to my talk so much i lost count
- BSides crew for arranging everything


Synack®
Questions?