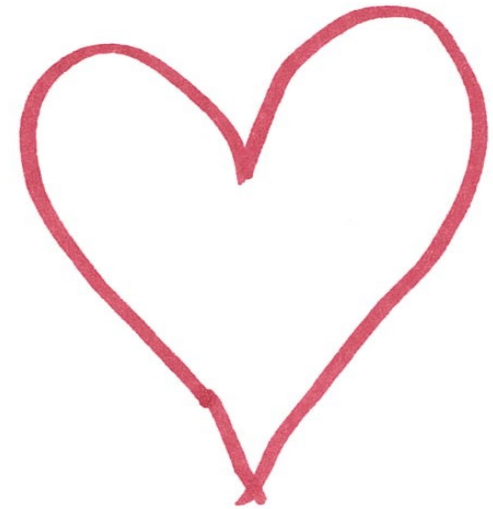
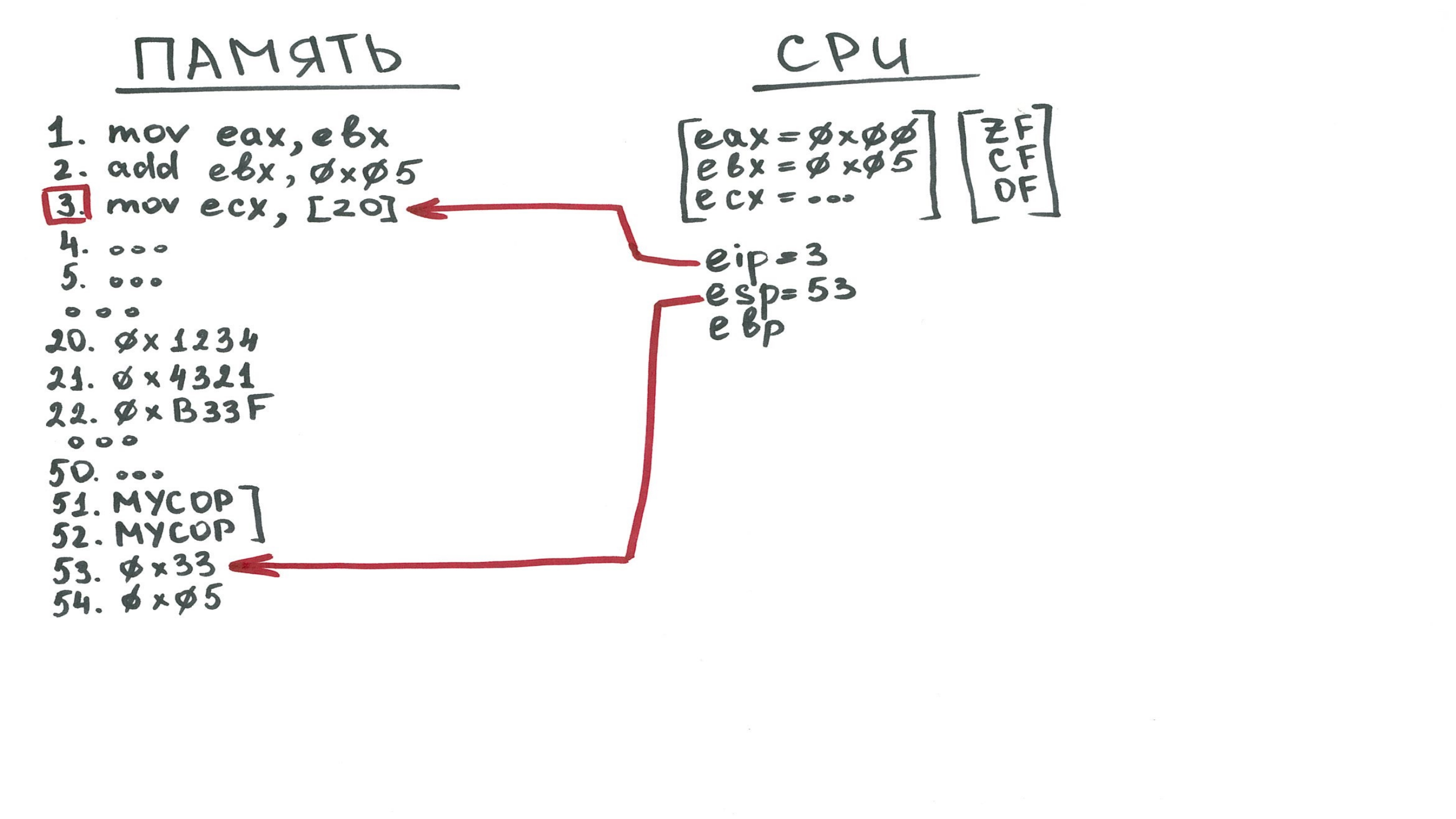
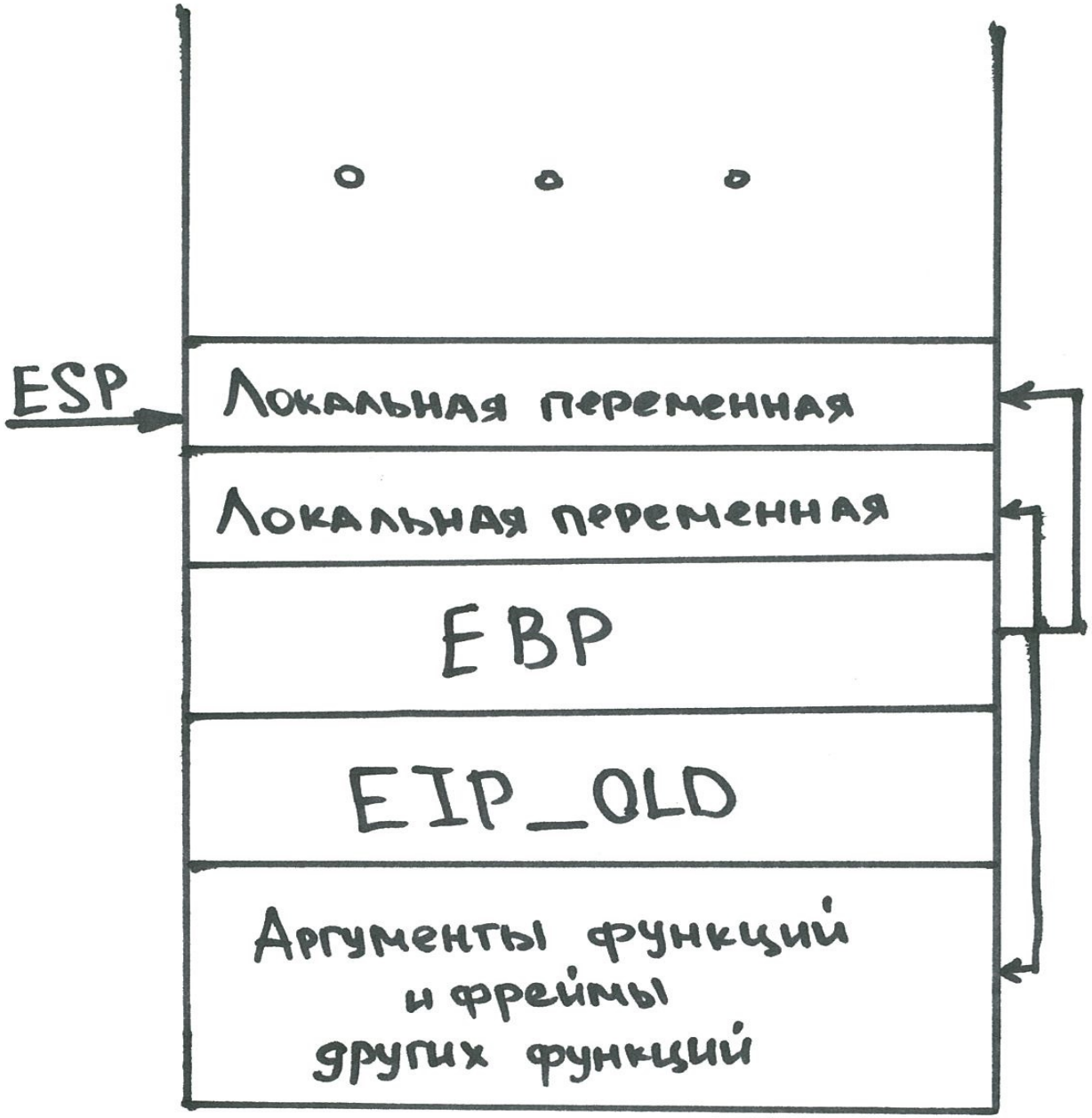
Memory corruption 101



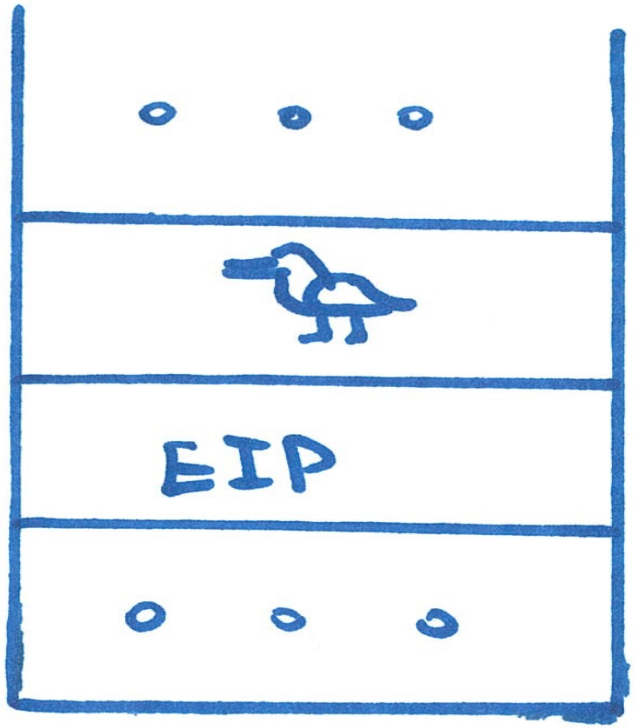

ПАМЯТЬ
1. mov eax, ebx
2. add ebx, 0x05
3. mov ecx, [20]
4. ...
5. ...
...
20. 0x1234
21. 0x4321
22. 0xB33F
...
50. ...
51. МУСОР
52. МУСОР
53. 0x33
54. 0x05
CPU
eax = 0x00
ebx = 0x05
ecx = ...
ZF
CF
OF
eip = 3
esp = 53
ebp



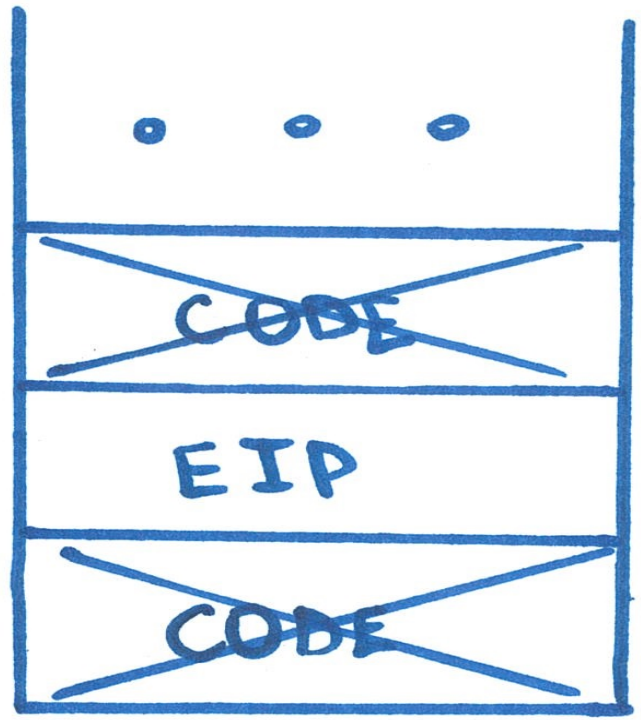

```
int main( ){
    char source[ ]="AAAA...A";
    char dest [8];
    strcpy(dest, source);
    return 0;
}
```


КОД, КОТОРЫЙ
НАМ ЗАХОТЕЛОСЬ
EBX?
EDX?
EIP_OLD
JMP EXX
ESP?
ИЛИ ЗДЕСЬ

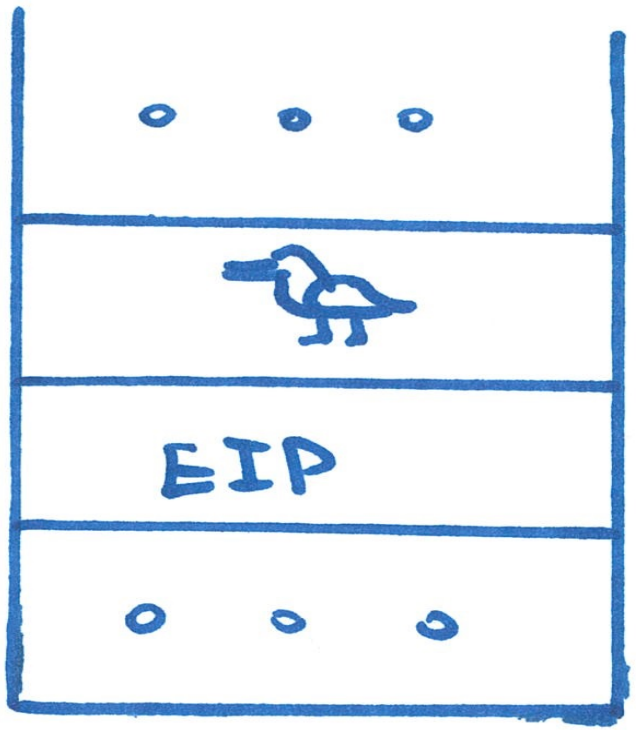
CANARY



NX bit

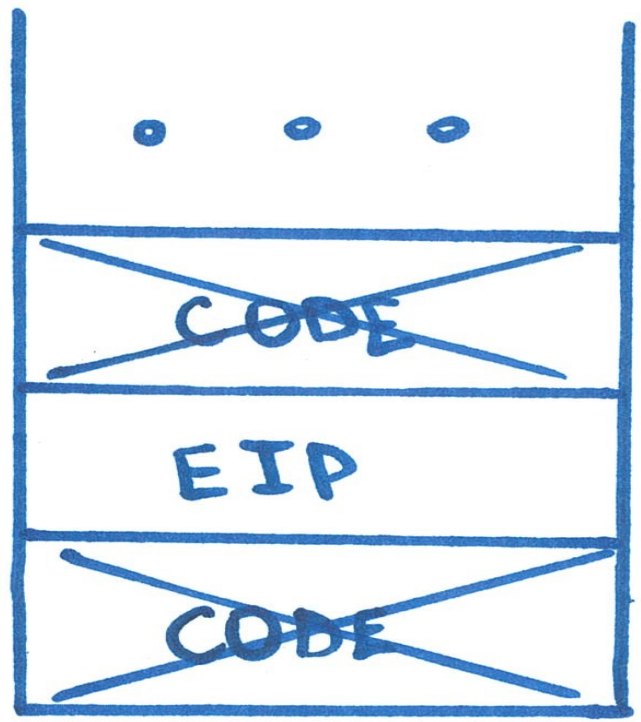


CANARY



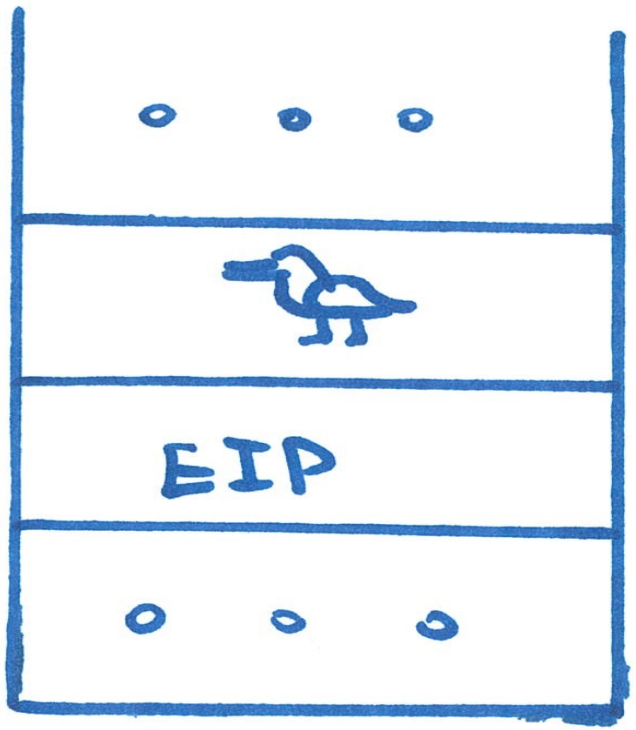
SEH

NX bit

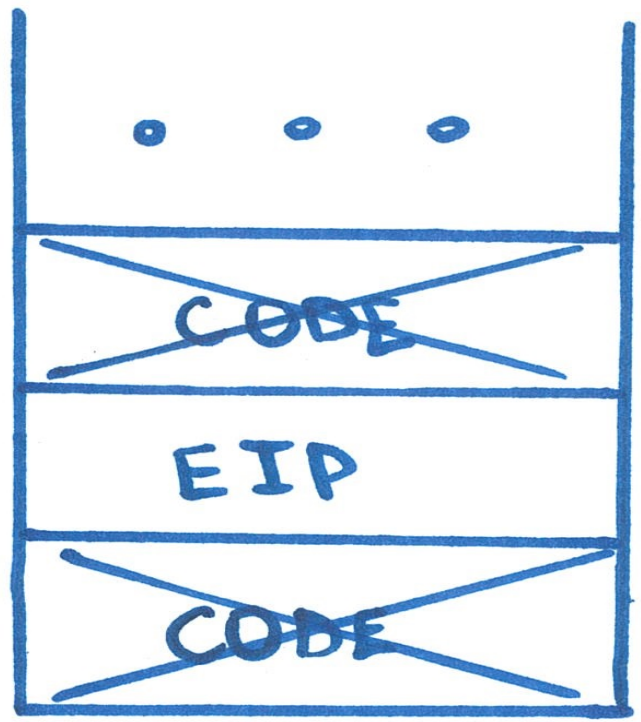


ROP

CANARY

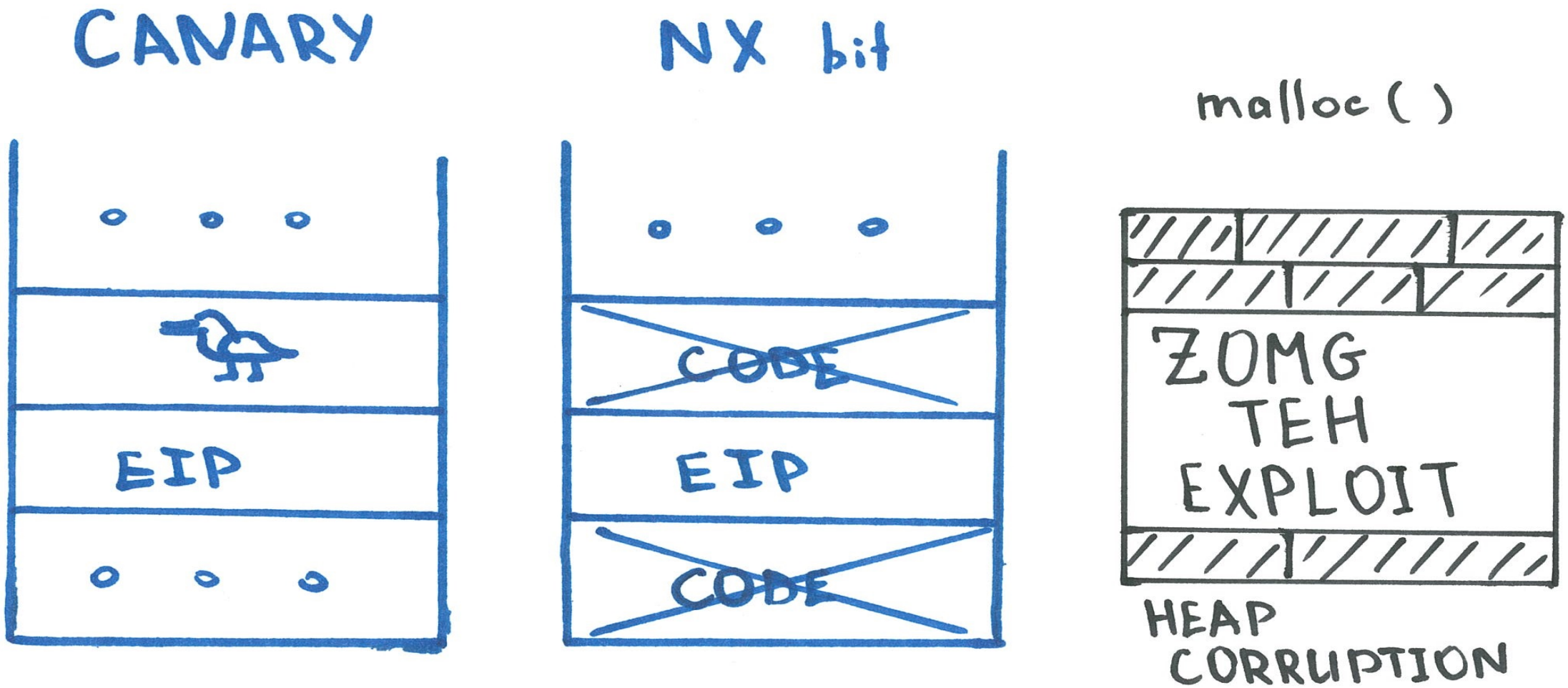


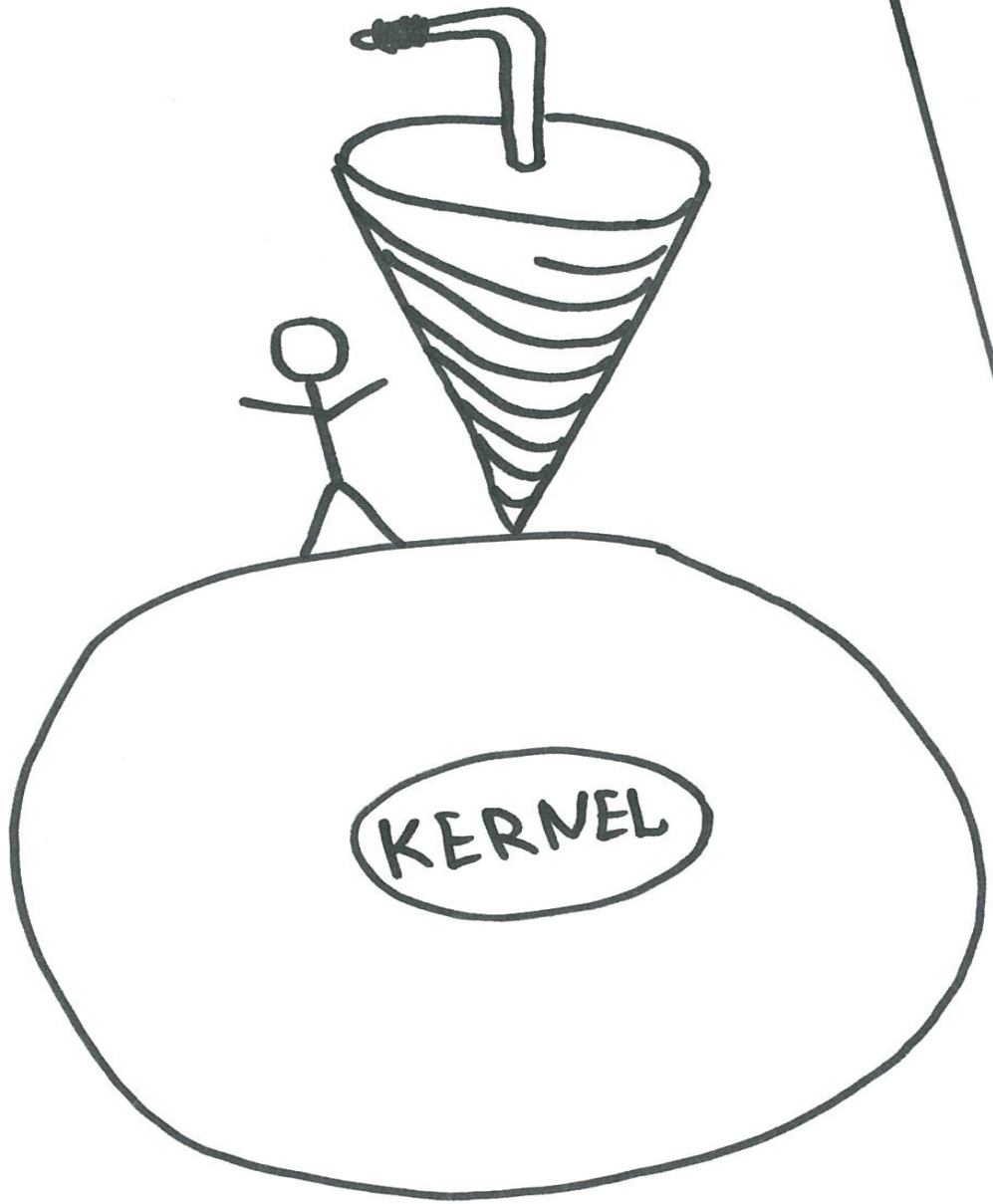
NX bit



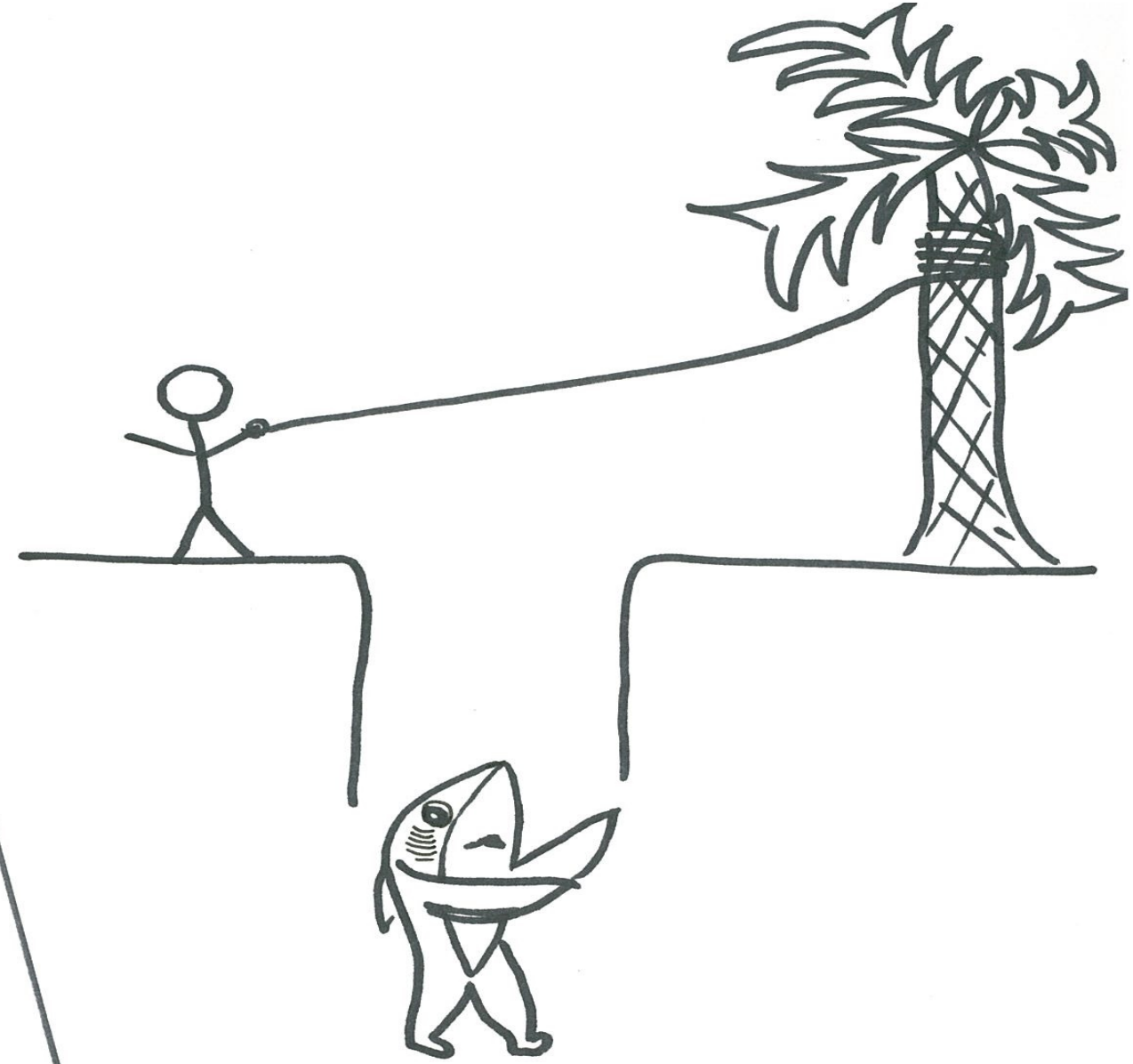
SEH ⟹ SafeSEH

ROP ⟹ ASLR


CANARY
NX bit
malloc()
EIP
CODE
EIP
CODE
ZOMG
TEH
EXPLOIT
HEAP
CORRUPTION
SEH
ROP
SafeSEH
ASLR



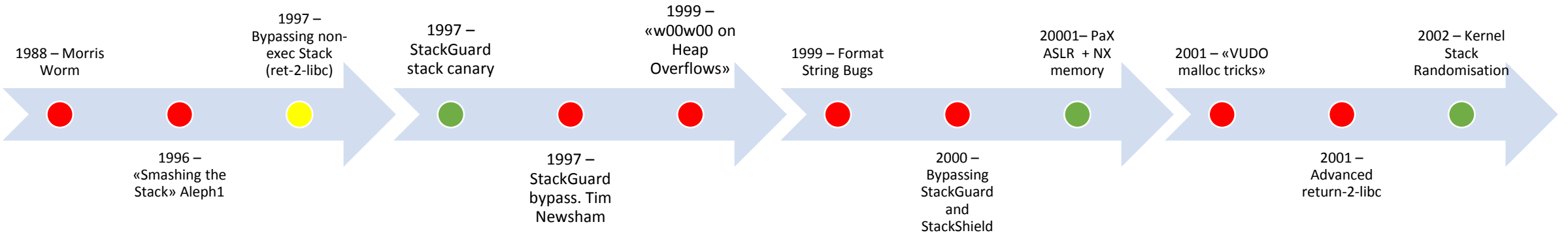
OPEN BSD

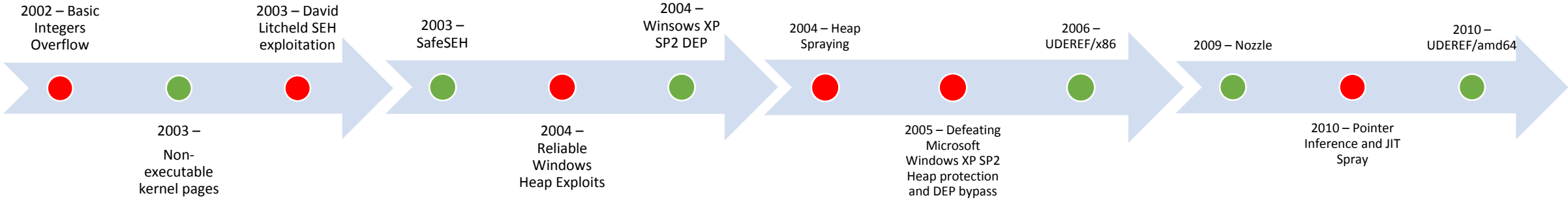


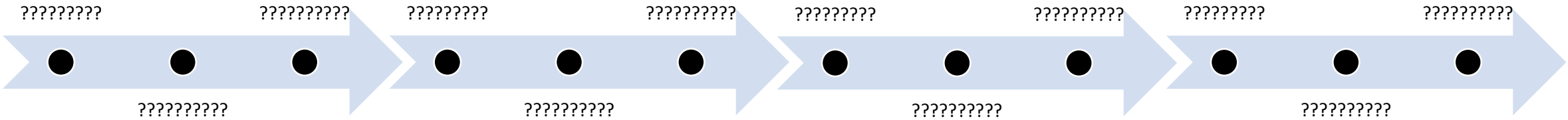
Windows 10

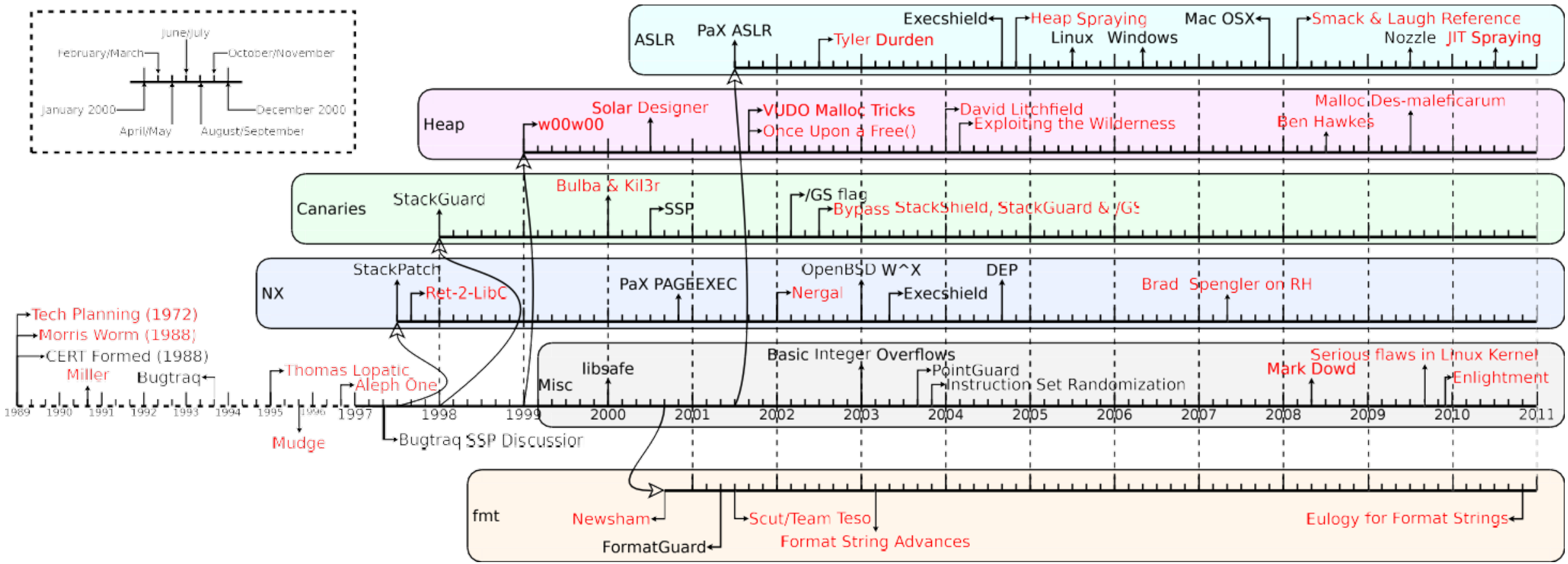
Style points

11/10


1988 – Morris Worm
1996 – «Smashing the Stack» Aleph1
1997 – Bypassing non-exec Stack (ret-2-libc)
1997 – StackGuard stack canary
1997 – StackGuard bypass. Tim Newsham
1999 – «w00w00 on Heap Overflows»
1999 – Format String Bugs
2000 – Bypassing StackGuard and StackShield
20001– PaX ASLR + NX memory
2001 – «VUDO malloc tricks»
2001 – Advanced return-2-libc
2002 – Kernel Stack Randomisation


2002 – Basic Integers Overflow
2003 – Non-executable kernel pages
2003 – David Litcheld SEH exploitation
2003 – SafeSEH
2004 – Reliable Windows Heap Exploits
2004 – Winsows XP SP2 DEP
2004 – Heap Spraying
2005 – Defeating Microsoft Windows XP SP2 Heap protection and DEP bypass
2006 – UDEREF/x86
2009 – Nozzle
2010 – Pointer Inference and JIT Spray
2010 – UDEREF/amd64


??????????
??????????
??????????
??????????
??????????
??????????
??????????
??????????
??????????
??????????
??????????
??????????



«Memory Errors: The Past, the Present, and the Future»: Victor van der Veen , Nitish dutt-Sharma , Lorenzo Cavallaro , and Herbert Bos (http://www.isg.rhul.ac.uk/sullivan/pubs/raid-2012.pdf)

“As a 'new school' binary vulnerability researcher, I've found it somewhat challenging to learn the subject in the times when it's become highly commercialized, which pushed the detailed technical security advisories and technical analyses of regular vulnerabilities out of the public access.“

Алиса Шевченко
(http://phrack.org/issues/69/10.html)

Demo Time

Эксплуатация бинарных уязвимостей

- Идентификация уязвимости
  - Статический анализ (исходный код, ассемблер)
  - Динамический анализ (fuzzing)
  - Символьное исполнение (symbolic execution)
- Подготовка PoC эксплойта и поиск возможности выполнить произвольный код
- Создание боевого эксплойта