AVG®

RECON 2014

SHALL WE PLAY A GAME?

Lessons learned while playing CoreWars8086

Shapira Elad ('Zest') | Security Researcher | 29-6-2014

#Whois Elad Shapira ('Zest')

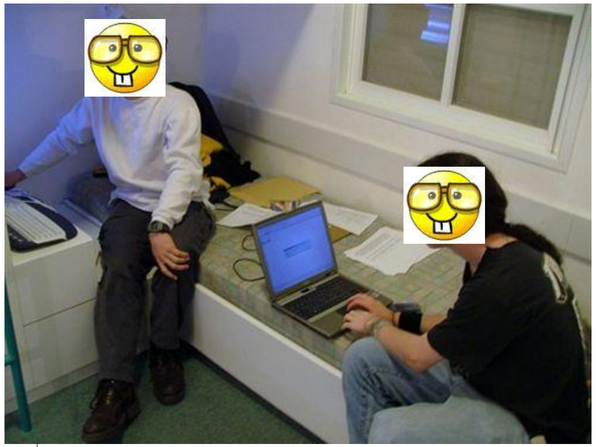
- Reverser from the Holy Land.
- Mobile Security Researcher @AVG.
- Highly passionate for RE, Assembly and Low-Level.
- Speaker (ClubHack, Ground Zero Summit..).
- Co-Organizer of CoreWars8086 competition (IL).

AVG

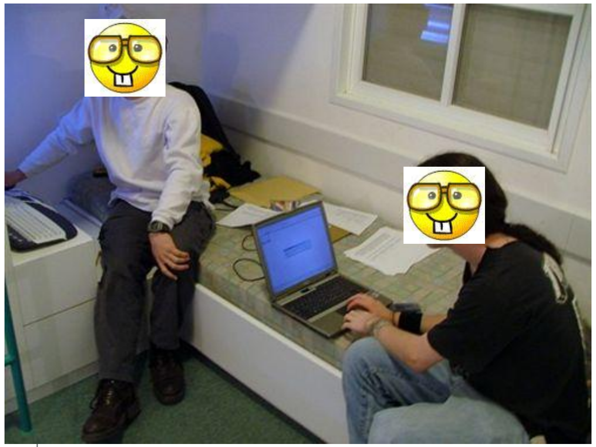
Popular Israeli TV show: "Beauty & the Geek"

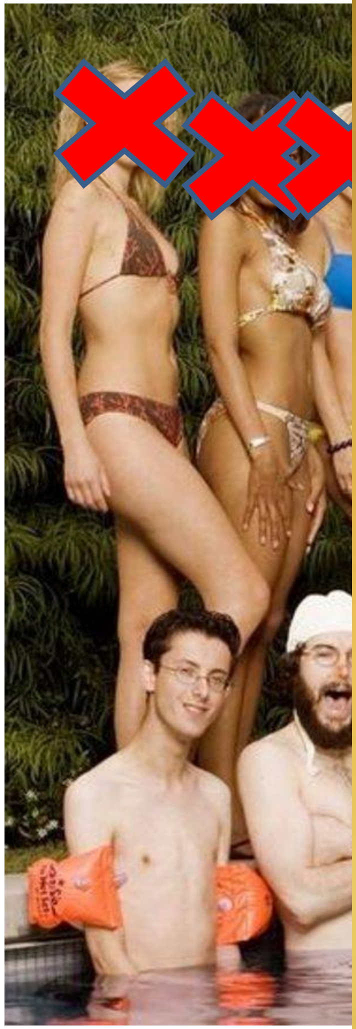
AVOID HANGOVERS
STAY DRUNK

Unofficial name: "Israeli idol of the geeks"



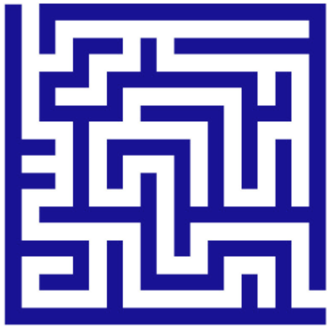
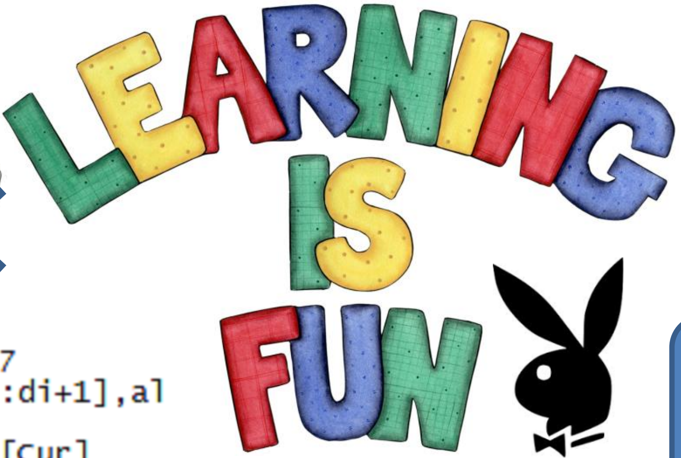
Unofficial name: “Israeli idol of the geeks”





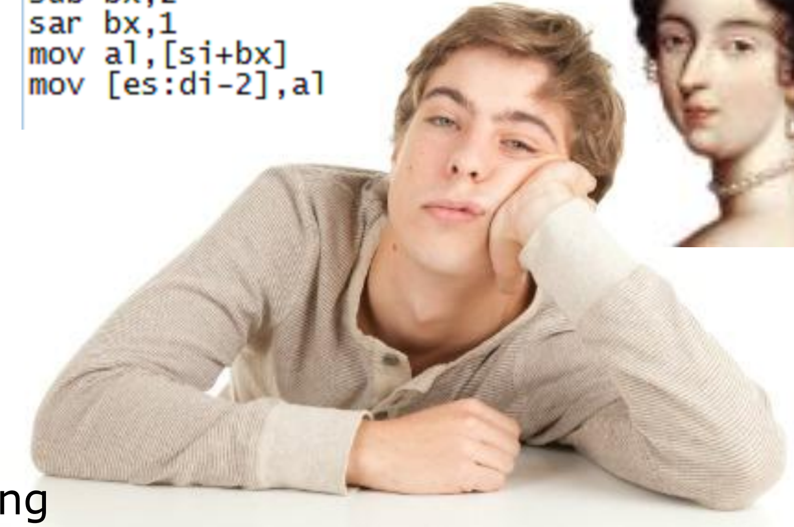
A total of 300 tickets will be sold for this year's conference.

Why CoreWars8086?

```
mov al,7
mov [es:di+1],al

mov di,[cur]

mov bx, di
sub bx,2
sar bx,1
mov al,[si+bx]
mov [es:di-2],al
```

Does it got any sports in it?



“No Starch”.. ☺

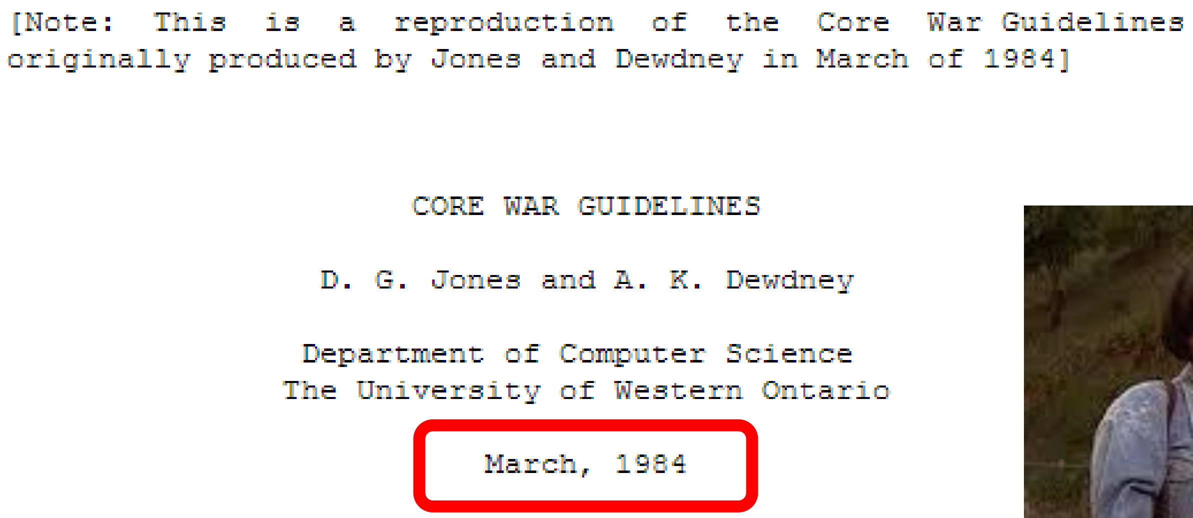
ng

Agenda

- Timeline of the CoreWars8086 competition.
- Arena, Engines and rules.
- How to analyze and write survivors.
- Optimization.
- Anti reversing techniques.
- Future / Improvements.
- Share ideas → Create new ideas!
- Hangover.

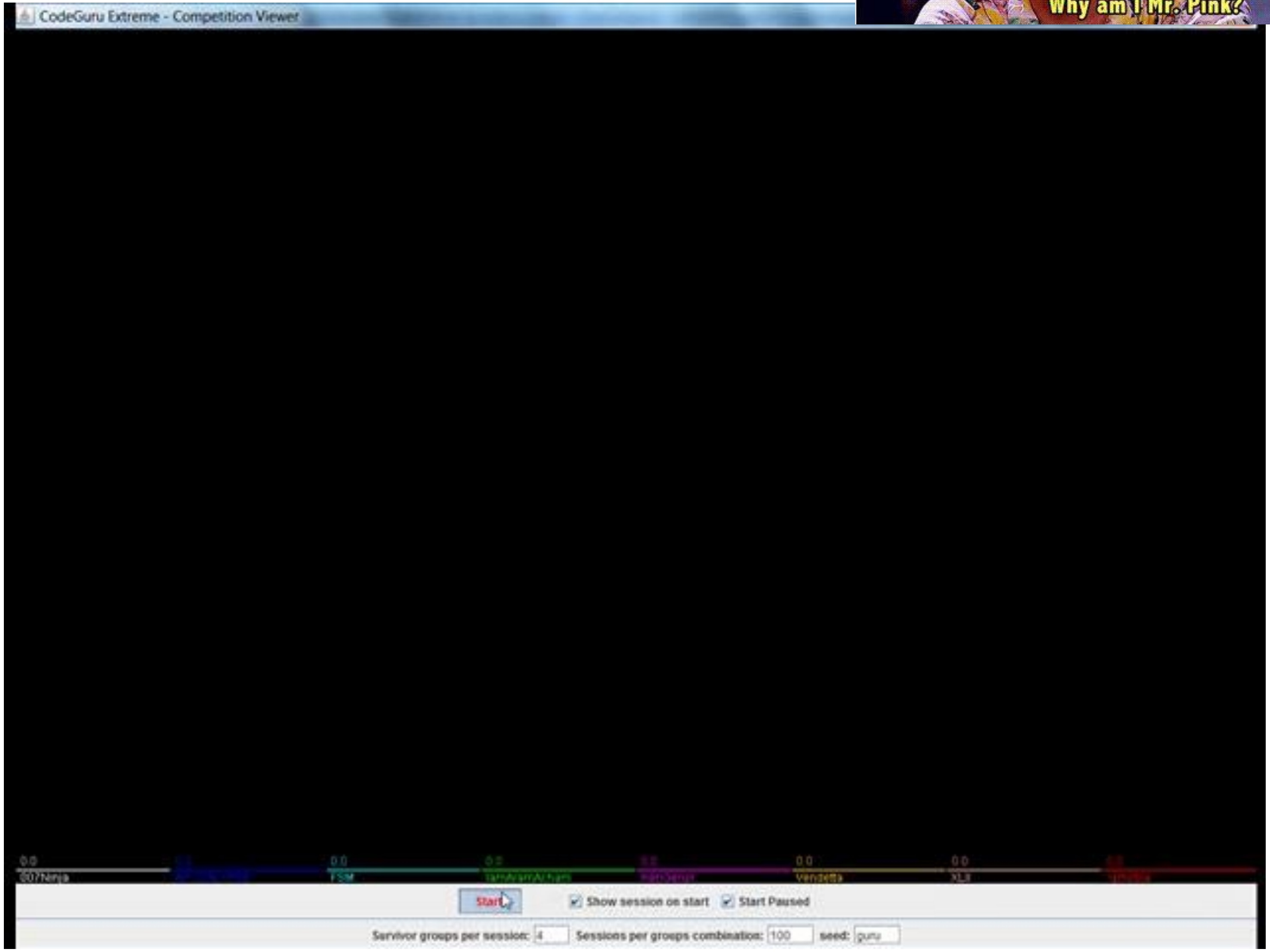
Origin

- Alexander Dewdney / D.G. Jones.
- CoreWars / RedCode
- http://vyznev.net/corewar/guide.html



```
[Note:  This  is  a  reproduction  of  the  Core  War Guidelines
originally produced by Jones and Dewdney in March of 1984]


                    CORE WAR GUIDELINES

                 D. G. Jones and A. K. Dewdney

                Department of Computer Science
               The University of Western Ontario

                       March, 1984
```

Fight Club – The digital version



Survivor groups per session: 4 Sessions per groups combination: 100 seed: guru

Cameras usually add 5 kg ..

Timeline of the competition

We got cool T-shirts from our sponsors!

- Getting zombies from the organizers.

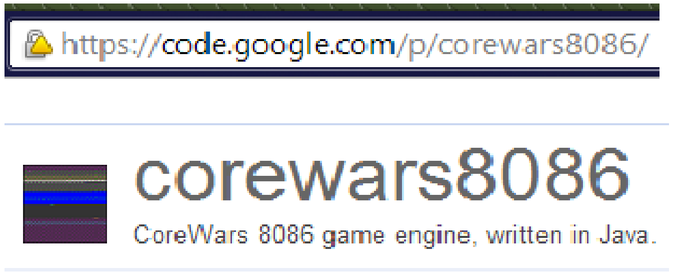
Zombies

- 1st round (remote) – 25%
- 2nd round (Face-2-face) – 25% 09:00 AM

Other competitors

- 3rd round (Face-2-face) – 50% 12:01PM
- Top 4 survivors get to the final.
- Final → Winners!

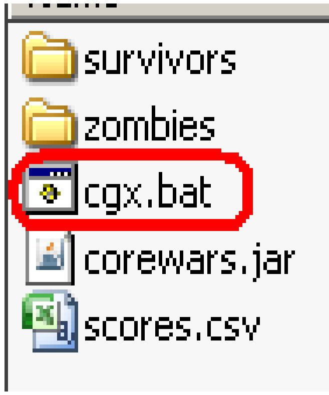
Survivors in general


https://code.google.com/p/corewars8086/

corewars8086

CoreWars 8086 game engine, written in Java.

- **Download, Unzip & play** (Google Code).
- Survivor's name == file's name (without extension).
- 8086 opcodes, **16bit** instructions.
- Not all instructions are supported (Pusha,Popa,..).
- Compiled as **'com'** file
  - DOS command file format.
- **Maximal** survivor size - **512** bytes.
- **Each team** can submit **two survivors**.
  - Rocky1 & Rocky2.


survivors
zombies
cgx.bat
corewars.jar
scores.csv

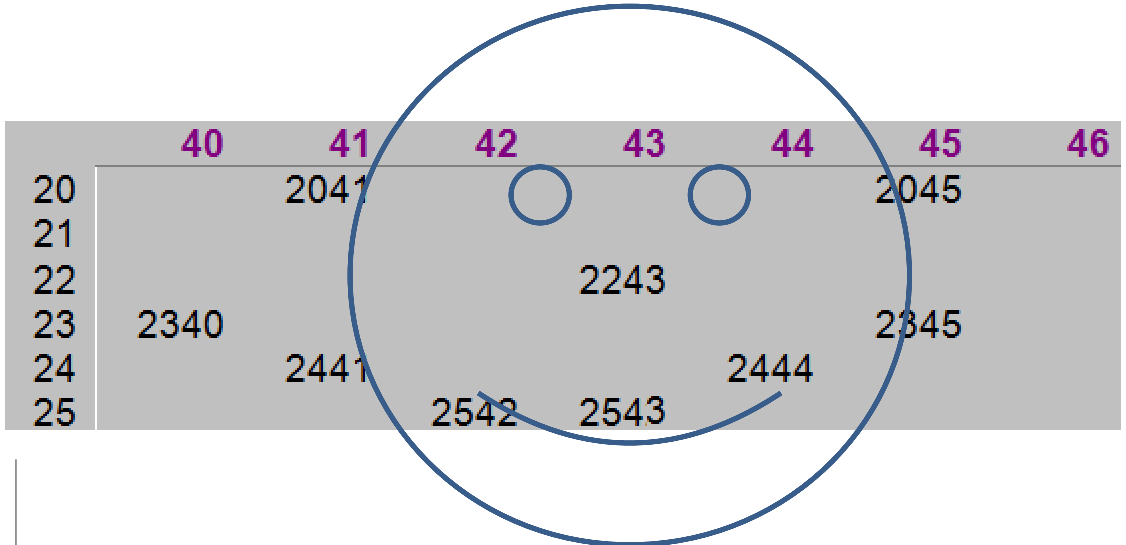
Virtual Arena

- Loaded to the virtual arena each time with **random address** (copied “as is”).
- **Distance** between two survivors and the sides is at least **1024** bytes.
- **All cells** initialized to ‘**CCh**‘ before start.
- End of the battle
    - **200,000 rounds** or **one survivor left**.
- **Order of the survivors is determined randomly** at the beginning and cannot be changed.

Arena (NOT virtual)

Arena & Addresses

	00	01	..	..	FE	FF
00	0000	0001	..	..	00FE	00FF
01	0100	0101	..	..	01FE	01FF
:	:	:			:	:
:	:	:			:	:
FE	FE00	FF01	..	..	FEFE	FEFF
FF	FF00	FF01	..	..	FFFE	FFFF



```
mov [2041h], al
mov [2045h], al
mov [2243h], al
mov [2340h], al
mov [2441h], al
mov [2542h], ax
mov [2444h], al
mov [2345h], al
```

Survivor's Registers (before 1st round)

- **BX,CX,DX,SI,DI,BP** = 00s.
- **Flags** = 00s.
- **AX, IP** - Initial location of the survivor, offset.
- **CS, DS** - Segment that was assigned to the survivors.
- **ES** - Segment for survivors from same team (shared memory) – 2048 bytes.
- **SS** - Beginning of the personal stack (2048).
    - ss:0x00 - ss:0x7ff, initialized to 0x00.
- **SP** - Offset of beginning of personal stack (00s).

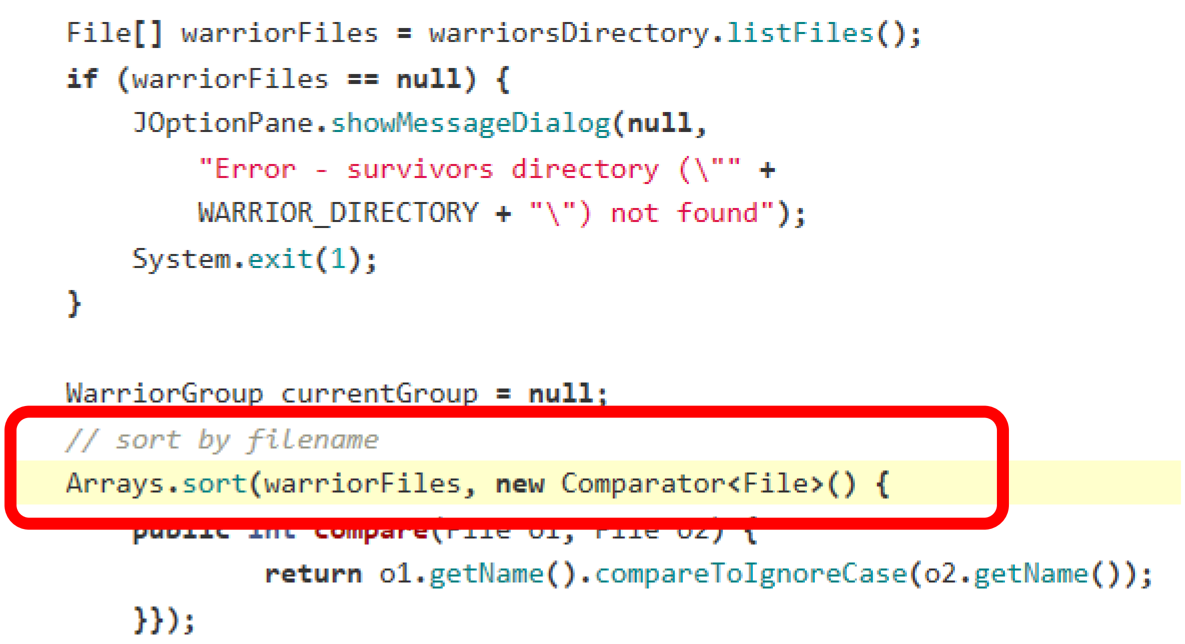
How survivor gets killed

- **Running illegal command**
  - The 060h byte does not translated to an assembly command.
  - Engine: "Died due to CPU".
- Running **commands** that are **not supported by the engine**
  - For example 'int 21h'.
- **Access to memory** not in the arena or not in the range of the survivor's personal stack.
  - For example ES:0x1234.
  - Engine: "Died to memory exception".

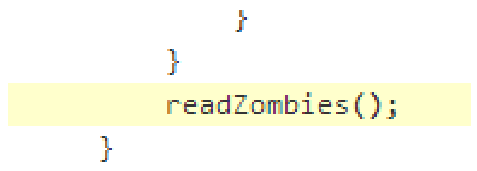
Zombies

- **Sent by organizers** before competition begins.
- Regular survivors that **do not get points**.
- **Different CPU states** problem.
  - Direction flag (MOVSW will kill master).
- Zombies **can still win the battle**
  - less points for us.
  - We need to encourage them to **commit suicide.**
- **Contain Math Riddles** (That you need to solve).

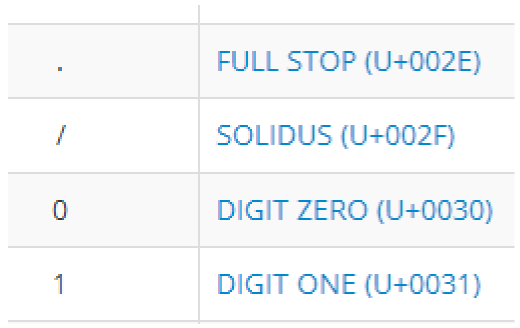
Pwning bugs in the engine



```
File[] warriorFiles = warriorsDirectory.listFiles();
if (warriorFiles == null) {
    JOptionPane.showMessageDialog(null,
        "Error - survivors directory (\"" +
        WARRIOR_DIRECTORY + "\") not found");
    System.exit(1);
}

WarriorGroup currentGroup = null;
// sort by filename
Arrays.sort(warriorFiles, new Comparator<File>() {
    public int compare(File o1, File o2) {
            return o1.getName().compareToIgnoreCase(o2.getName());
    }});
```



```
                }
            }
            readZombies();
        }
```



.	FULL STOP (U+002E)
/	SOLIDUS (U+002F)
0	DIGIT ZERO (U+0030)
1	DIGIT ONE (U+0031)

How to make your survivors be the firsts to run?

0SurvivorName

What is the advantage?

Zombies can fix your survivor’s code

~~0SurvivorTeam1 (x2)~~

SurvivorTeam2 (x2)

SurvivorTeam3 (x2)

Zombie1

Zombie2

Zombies can fix your survivors code

0SurvivorTeam1 (x2)
SurvivorTeam2 (x2)
SurvivorTeam3 (x2)
Zombie1
Zombie2

Zombies can fix your survivors code

0SurvivorTeam1 (x2)

SurvivorTeam2 (x2)

SurvivorTeam3 (x2)

Zombie1

Zombie2

Zombies can fix your survivors code

0S[illegible]v[illegible]am1 (x2)

SurvivorTeam2 (x2)

SurvivorTeam3 (x2)

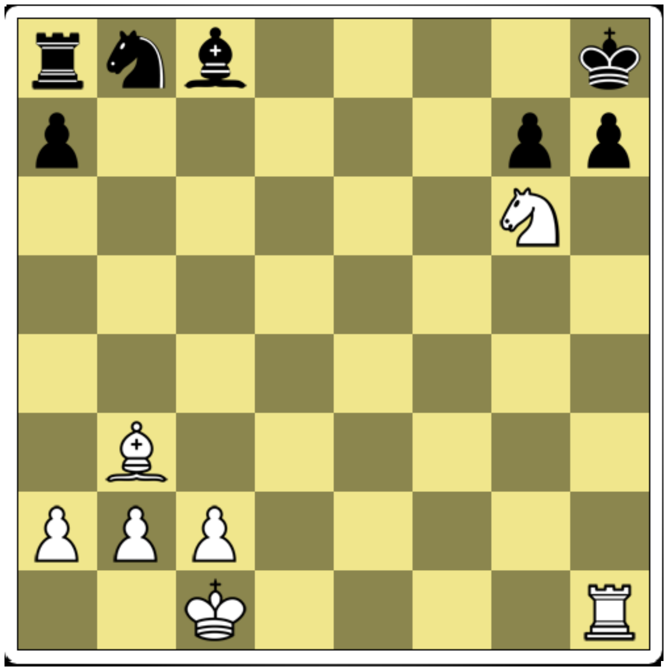
Zombie1

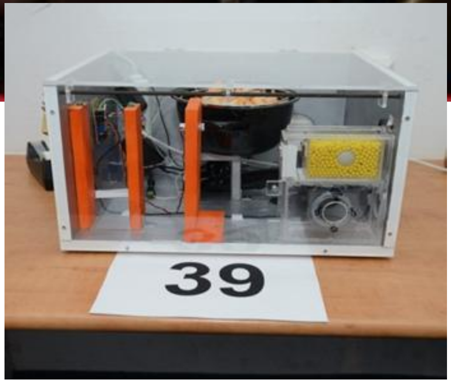
Zombie2

To stay on the safe side..

SOMEONE NAME HIS SURVIVOR WITH 0 AS FIRST CHAR
MY SURVIVOR BEGINS WITH FEW 0S
imgflip.com

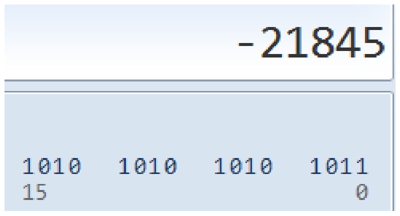
Safe Cracking



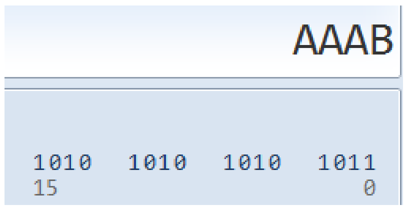


Safe example#1

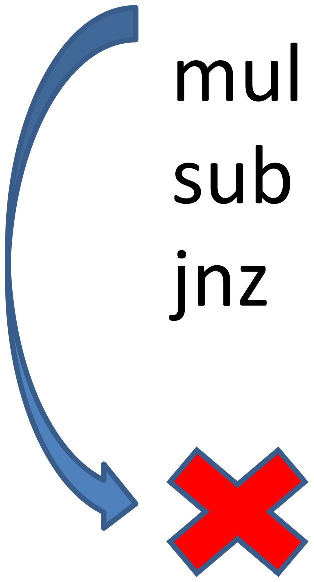
```
loop:
mov  AX,[1234]     [1234] = AAAB
mov  BX,3          3*AX=1
mul  AX            BX*AX=1
sub  AX,1          AX=1
jnz  loop          ZF=1
```



```
                  -21845
1010   1010   1010   1011
15                      0
```



```
                    AAAB
1010   1010   1010   1011
15                      0
```



Solution:

```
killer:
mov   AX, AAAB
mov   ptr word [1234], AX
JMP   killer
```

Safe example#2

```
loop:
mov  AL,[111]      [111] = 49H
add  AL,0A8h       73+168=241(F1)
mov  AH, [112]     [112] = 42H
xor  AH,0ADh       ADH xor 42H = EFH (239d)
mul  AH            AX = AH * AL = 239 * 241 = 57599
cmp  AX,0xe0ff     AX=57599d
jne  loop          ZF=1
```

Safe example#2

```
loop:
mov  AL,[111]
add  AL,0A8h
mov  AH, [112]
xor  AH,0ADh
mul  AH
cmp  AX,0xe0ff
jne  loop
```

Solution:

```
killer:
mov  AL, 49H
mov  AH, 42H
mov  ptr byte [111], AL
mov  ptr byte [112], AH
jmp  killer
```

Important factors

- Survivors usually contain
  - **Initialization.**
  - **Bombing loop**.
    - Write -> Update address for next writing -> Jumping to beginning of loop
- We usually measure survivors by
  - **‘Area of vulnerability’**
  - **‘Attack rate’**.
- We can cause unexpected phenomenon
  - mov AX, 0000 -> mov ax, 0cccch (2,3 bytes).

Looper

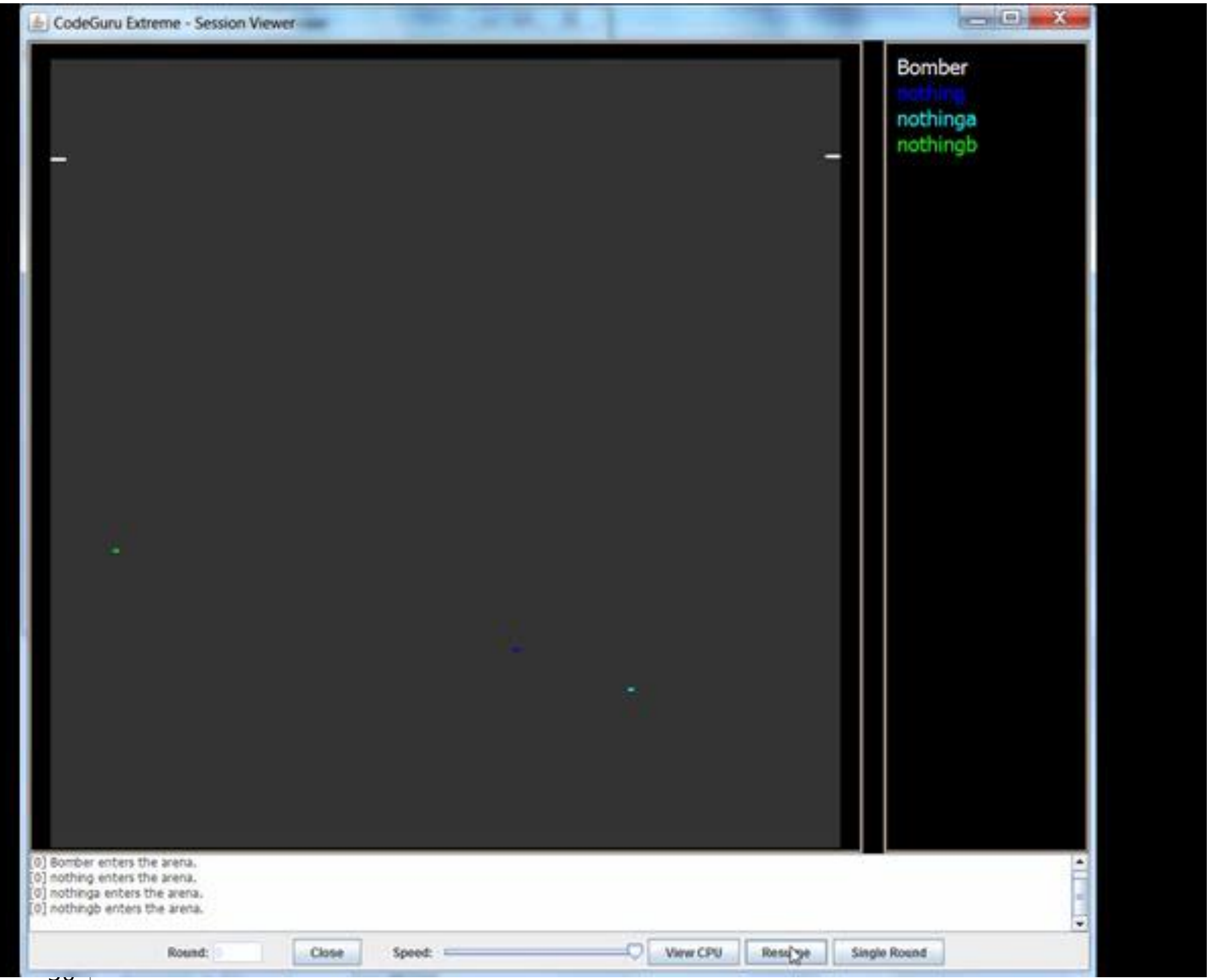
- Smallest functional survivor (EBFE, jmp $):

```
Loop:
        Jmp loop
```

- Good to test other survivors.

Bomber Demo

Attack sequence	Vulnerability profile
3 / 1	5

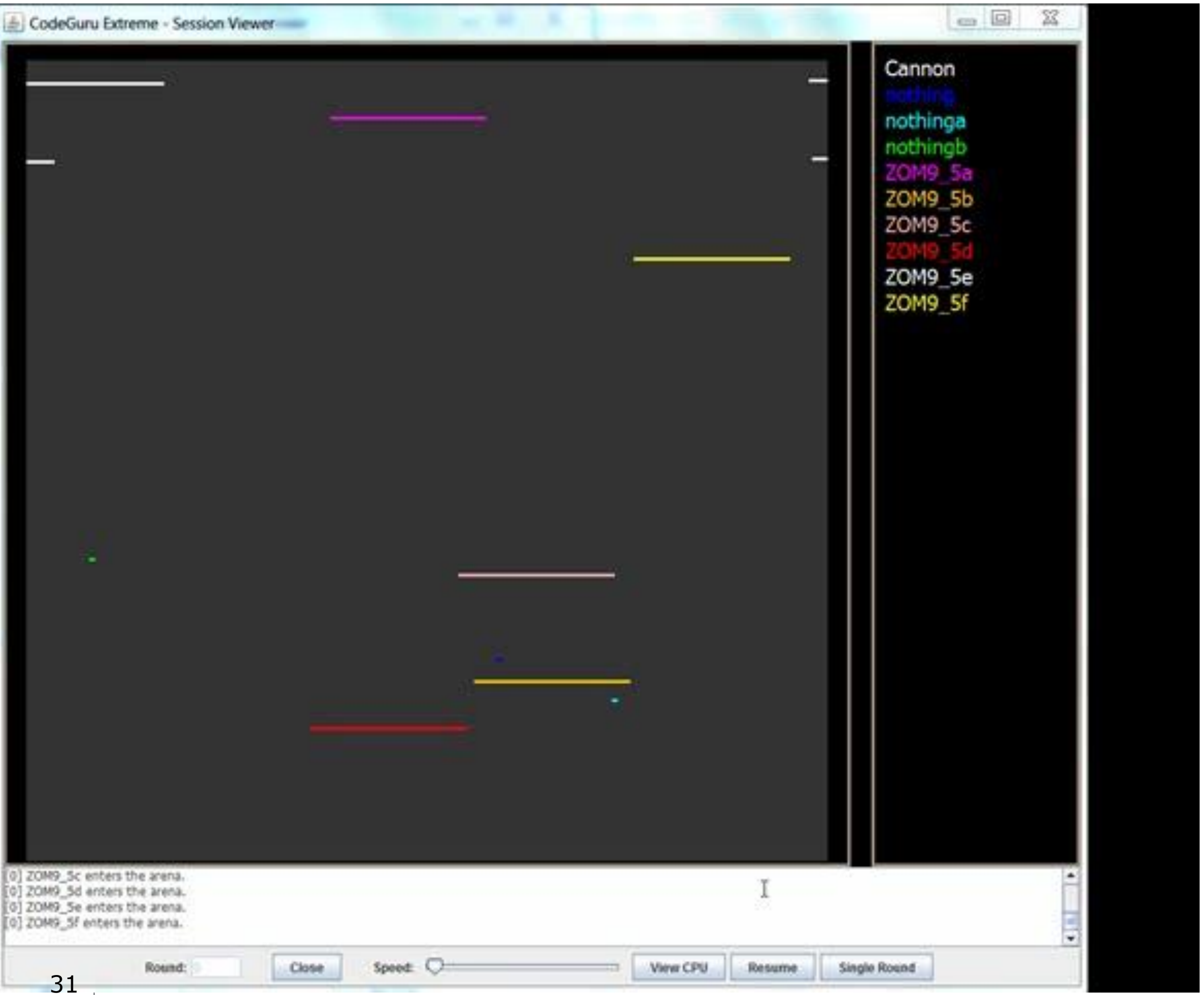


```
[0] Bomber enters the arena.
[0] nothing enters the arena.
[0] nothinga enters the arena.
[0] nothingb enters the arena.
```

```
mov al, 0CCh
mov bx, 0

@loop:
mov [bx], al
inc bx
jmp @loop
```

Cannon Demo

Attack sequence	Vulnerability profile
3 / 1	7

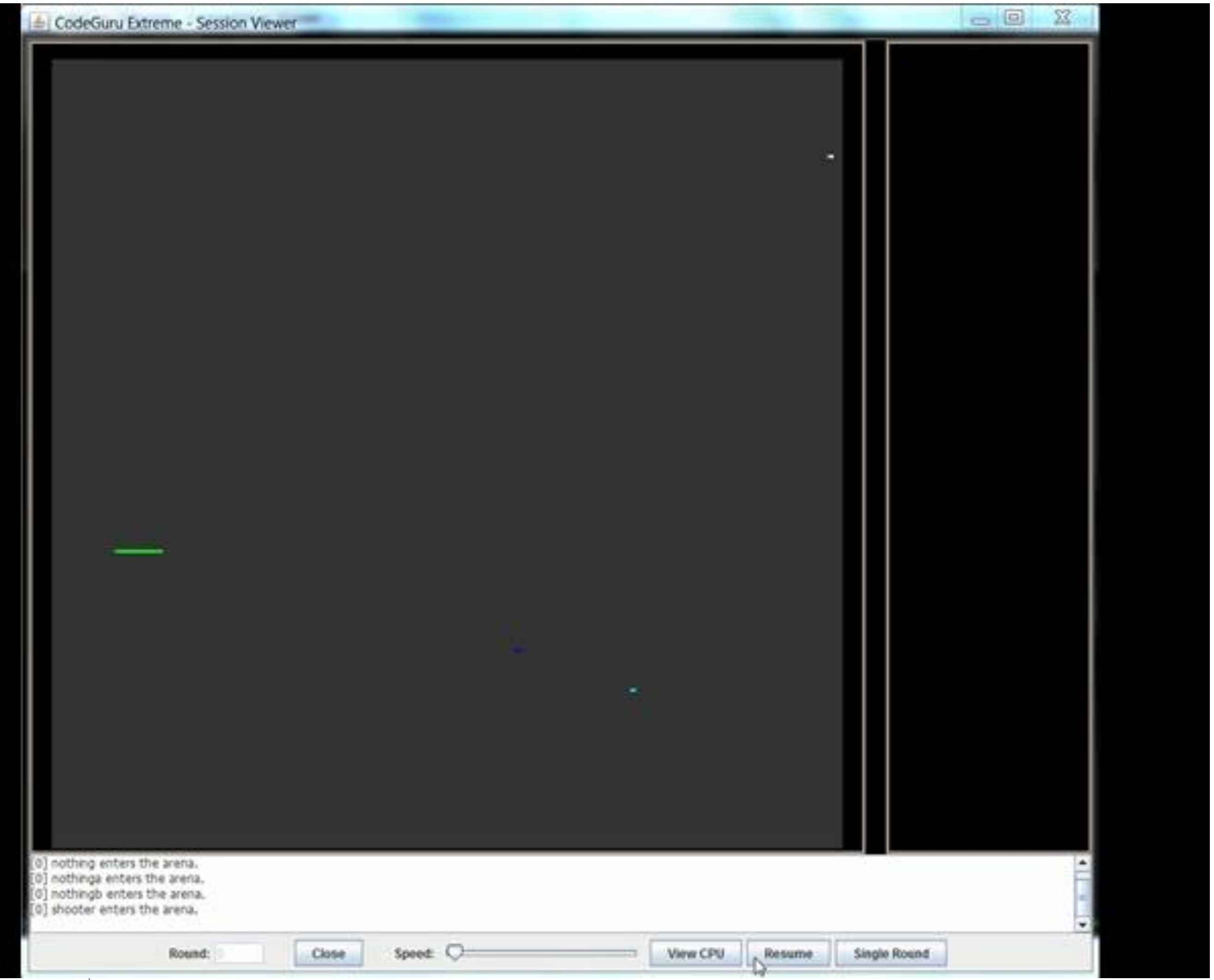


```
[0] ZOM9_5c enters the arena.
[0] ZOM9_5d enters the arena.
[0] ZOM9_5e enters the arena.
[0] ZOM9_5f enters the arena.
```

```
@start:
mov bx, ax
add bx, (@end - @start)
mov al, 0CCh

@loop:
mov [bx], al
add bx, 8

jmp @loop
@end:
```

Attack sequence	Vulnerability profile
3 / 2	6

Shooter Demo

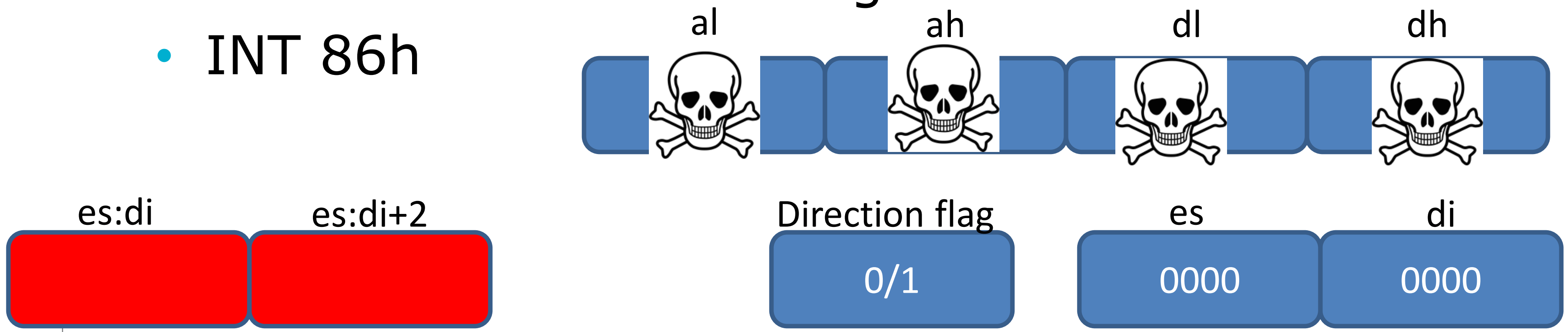


```
[0] nothing enters the arena.
[0] nothinga enters the arena.
[0] nothingb enters the arena.
[0] shooter enters the arena.
```

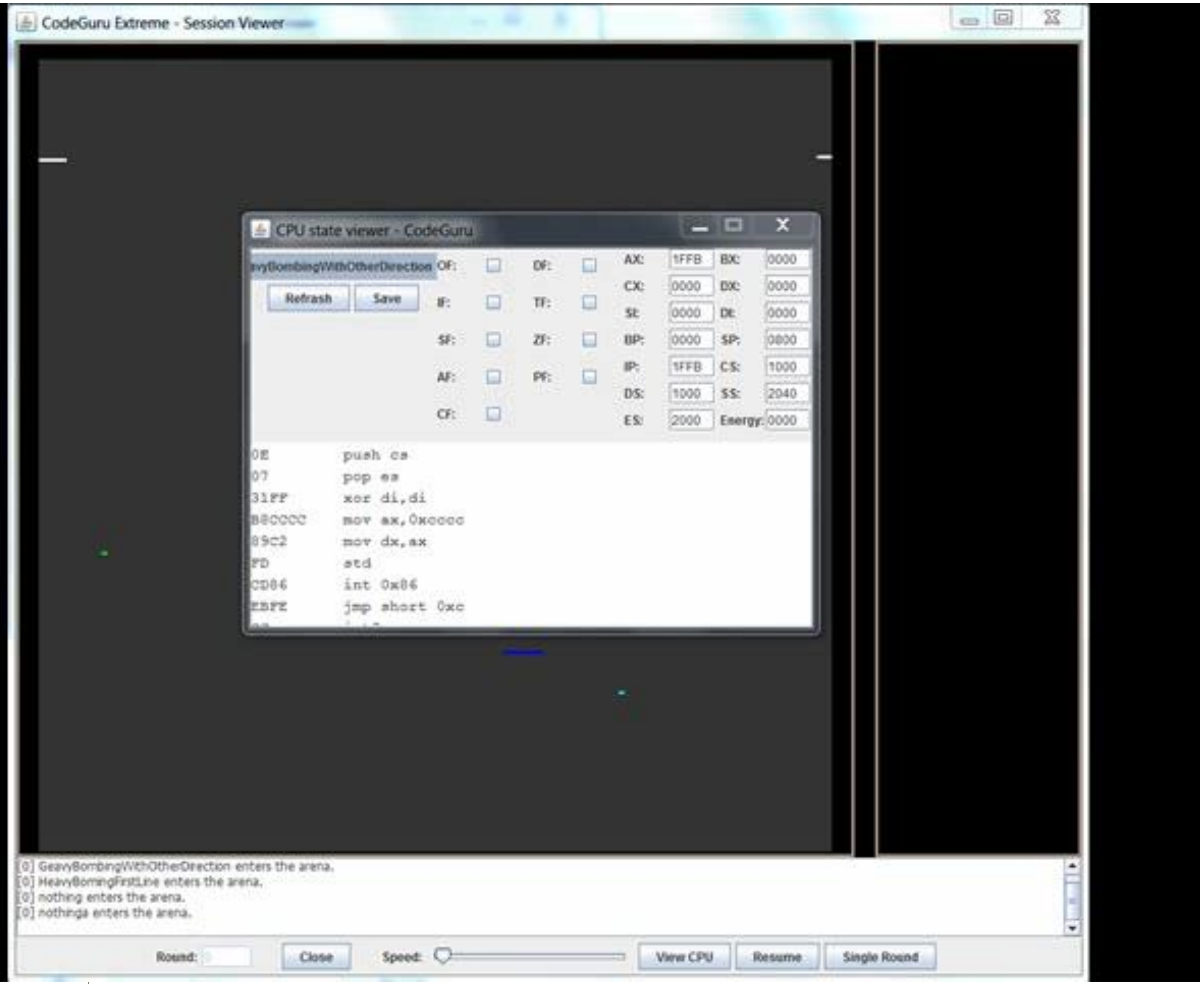
```
MOV DI,AX
MOV AX,0CCCCh

@loop:
STOSW
ADD     DI,9
JMP     @loop
```

Heavy Bombing

- Writes on 256 bytes (es:di -> 255 addresses)
- es same value as cs -> if not memory exception after the interrupt
- CLD/STD -> change direction
- 2 Heavy Bombing each battle
- We can bomb shared segment
- INT 86h



Heavy Bombing Demo (Opposite direction)


CodeGuru Extreme - Session Viewer
CPU state viewer - CodeGuru
vyBombingWithOtherDirection
Refrash
Save
Round:
Close
Speed:
View CPU
Resume
Single Round

OF:	☐	DF:	☐	AX:	1FFB	BX:	0000
				CX:	0000	DX:	0000
IF:	☐	TF:	☐	SI:	0000	DI:	0000
SF:	☐	ZF:	☐	BP:	0000	SP:	0800
AF:	☐	PF:	☐	IP:	1FFB	CS:	1000
				DS:	1000	SS:	2040
CF:	☐			ES:	2000	Energy:	0000

```
0E          push cs
07          pop es
31FF        xor di,di
B8CCCC      mov ax,0xcccc
89C2        mov dx,ax
FD          std
CD86        int 0x86
EBFE        jmp short 0xc
```

```
[0] GeavyBombingWithOtherDirection enters the arena.
[0] HeavyBomingFirstLine enters the arena.
[0] nothing enters the arena.
[0] nothinga enters the arena.
```

push cs

pop es

xor di,di

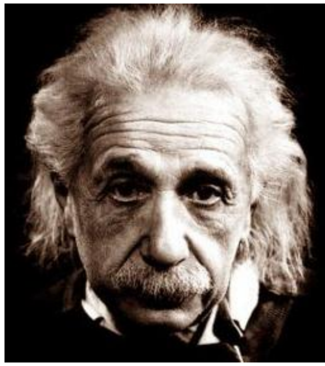
mov ax, 0ccccch

mov dx, ax

std

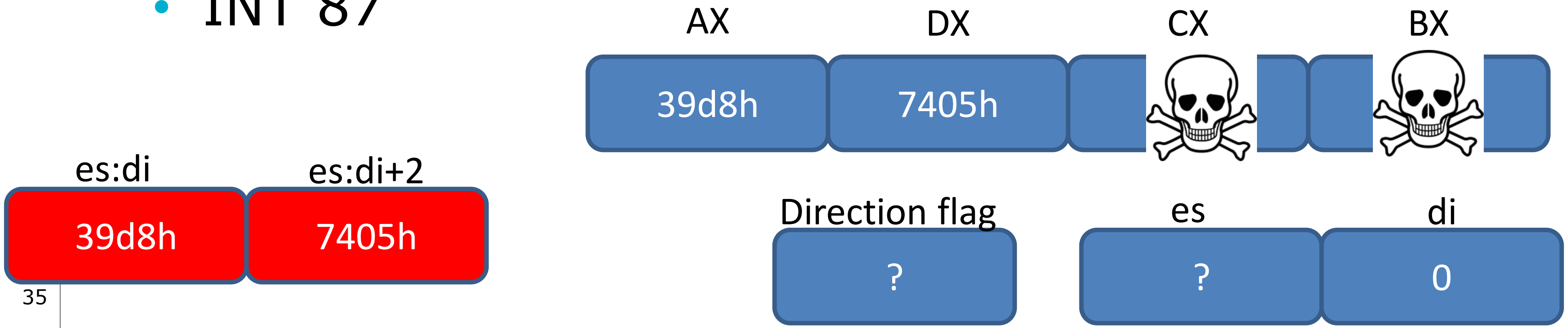
int 86h

jmp $



Smart Bombing

- Bombing the first occurrence of AX:DX in memory.
- Replacing it with data we want
    - Illegal commands or jmp to our code.
- We can attack ourselves..
- 1 Smart Bombing each battle.
- INT 87



Protection from Smart Bombing

- Change **functionality of registers** (BX <-> BP).
    - Usually does not matter.
- Change **order of independent commands**
    - Put 3 values to 3 registers = Few different ways.
- **copy parts of the code**
    - To the beginning and the end.
- **Variable that changed during runtime** near main loop/code part (SP).
- **Encoding with random numbers.**
- **XORing** (will be discussed later).

Smart bombing FAIL protection (CGX#9.5)

```
jmp short 0x12
mov si,0x95a0
xchg ax,bx
cld
lodsw
std
cmp ax,bx
jnc 0xc
or al,0x90
lodsw
loop 0x6
mov si,0x95a0
xchg ax,bx
Cld
lodsw
```

E2F4
BEA0

```
std
cmp ax,bx
jnc 0x1c
or al,0x90
lodsw
loop 0x16
mov si,0x95a0
xchg ax,bx
cld
lodsw
std
cmp ax,bx
jnc 0x2c
or al,0x90
lodsw
loop 0x26
```

```
push cs
pop es
mov ax, 0F4E2h
mov dx, 0A0BEh
mov cx, 0cccch
mov bx,cx
STD
Int 87h
Jmp $
```

Zombie ==?

Binary search (“Lion in the desert”)

```
jmp short 0×12
..
mov si,0x95a0
xchg ax,bx
cld
lodsw
std
cmp ax,bx
jnc 0x1c
or al,0×90
lodsw
loop 0×16
```

Jumping to body

The "talking location" that the survivors and the zombie talk in

Keep loading address on the side (LODSW will change AX)

Clears the direction flag (DF=0)

LODSW === MOV AX,[SI++ or SI--]

AX will hold the ‘talking location’

DF=1 (later SUB SI, 2 to change back)

Binary search (“Lion in the desert”)

```
jmp short 0×12
..
mov si,0x95a0
xchg ax,bx
cld
lodsw
std
cmp ax,bx
jnc 0x1c
or al,0×90
lodsw
loop 0×16
```

Compare his address (BX) to talking location (AX) - change only flags.

AX >= BX

jumps into itself (IP increased by 1)

73 FF   73 **FF**   Dec [si] Next cell

0C 90   **0C** 90   nop

changes AL + AX changed again?

hidden Dec[Si] command ☺

DF=1 (sub si, 2 to change back)

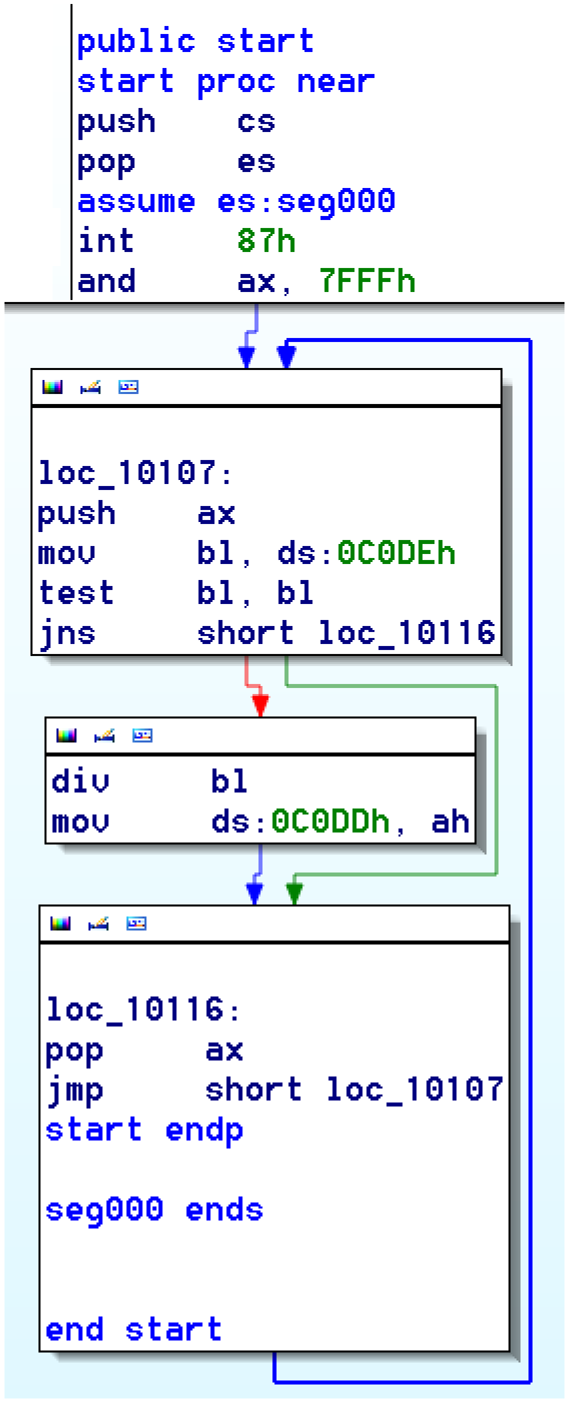
Zombie ==?

6 Zombies

```
push cs
pop es
int 0x87
and ax,0x7fff
push ax
mov bl,[0xc0de]
test bl,bl
jns 0x16
div bl
mov [0xc0dd],ah
pop ax
jmp short 0x7
```

```
mov bl,[0xc0de]
mov bl,[0xc1de]
mov bl,[0xc2de]
mov bl,[0xc3de]
mov bl,[0xc4de]
mov bl,[0xc4de]
```

```
mov [0xc0dd],ah
mov [0xc1dd],ah
mov [0xc2dd],ah
mov [0xc3dd],ah
mov [0xc4dd],ah
mov [0xc4dd],ah
```



```
public start
start proc near
push    cs
pop     es
assume es:seg000
int     87h
and     ax, 7FFFh

loc_10107:
push    ax
mov     bl, ds:0C0DEh
test    bl, bl
jns     short loc_10116

div     bl
mov     ds:0C0DDh, ah

loc_10116:
pop     ax
jmp     short loc_10107
start endp

seg000 ends


end start
```

Not to be confused with the military theorist Sun Tzu

Chinese Remainder Theorem

$$\begin{aligned}2x&\equiv 1 \pmod{3}\\ 3x&\equiv 2 \pmod{4}\\ 4x&\equiv 3 \pmod{5}\end{aligned} \iff \begin{aligned}x&\equiv 2 \pmod{3}\\ x&\equiv 2 \pmod{4}\\ x&\equiv 2 \pmod{5}\end{aligned}$$

$$x \equiv 2 \pmod{60}$$

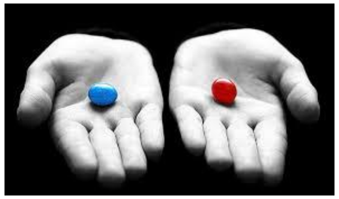
Formula used to find all the zombies:

```
input = ?
a1 = (input%254);
a2 = (input%255);
input = ( a1*255*1 + a2*254*254 )%( 255*254 );
```

Sometime, the organizers send invalid zombies…

WORKS ALL NIGHT ON PWNING THE ZOMBIE
"SORRY TEAMS, WE FOUND A BUG IN THE ZOMBIE"
imgflip.com

Optimization



#	bytes	opcodes	rounds	Not optimized	Optimized	bytes	opcodes	rounds
1	3	0x83 0xc0 0x01	1	add ax,1	inc ax	1	0x40	1
2	3	0x83 0xe8 0x01	1	sub ax,1	dec ax	1	0x48	1
3	3	0xb8 0x00 0x00	1	Mov AX,0	Sub AX,AX	2	0x29 0xc0	1
4	5	0xb9 0x02 0x00 0xf7 0xf1	2	mov CX,2 div CX	shr ax,1	3	0xc1 0xe8 0x02	1
5	6	0x89 0xc2 0x89 0xd8 0x89 0xd3	3	mov dx,ax mov ax,bx mov bx,dx	XCHG ax,bx	1	0x93	1
6	7	0xa2 0x00 0x00 0x88 0x26 0x01 0x00	2	mov [0],al mov[1],ah	mov [0],ax	3	0xa3 0x00 0x00	1

Bit Twiddling Hacks

By Sean Eron Anderson
seander@cs.stanford.edu

How not to be seen

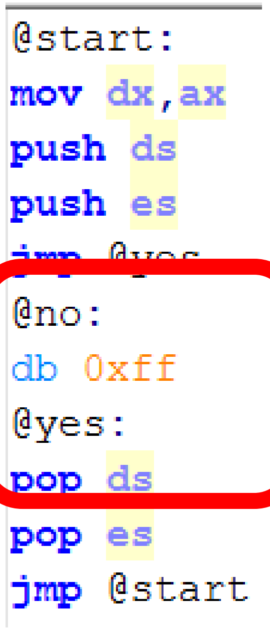
IamAramAcham
CGX9

#1 – Anti Disassembly

Disassembly

```
00000000  89C2   mov dx,ax
00000002  1E     push ds
00000003  06     push es
00000004  EB01   jmp short 0x7
00000006  FF1F   call word far [bx]
00000008  07     pop es
00000009  EBF5   jmp short 0x0
```

Original



```
@start:
mov dx,ax
push ds
push es
jmp @yes
@no:
db 0xff
@yes:
pop ds
pop es
jmp @start
```

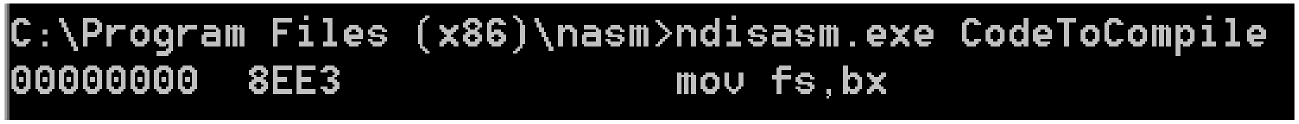
FF =
will never
happen

1F =
pop ds

#2 – Usage of unsupported registers (1/2)

- FS is unsupported by engine.
- difference between opcode interpretation between 8086 and later processors like 80386
- 8086 processor will read it like ‘ES’.

HutsHuts
CGX3



```
C:\Program Files (x86)\nasm>ndisasm.exe CodeToCompile
00000000  8EE3              mov fs,bx
```

push es, ds

move bx, ds

mov fs, bx

#2 – Usage of unsupported registers (2/2)

- mov <segment register>, <general purpose register>
- Binary value: 10001110dosssmmm

Advanced processors

000 - ES
001 - CS
010 - SS
011 - DS
100 FS (only 386+)
101 GS (only 386+)

8086 processor

Ignore first bit

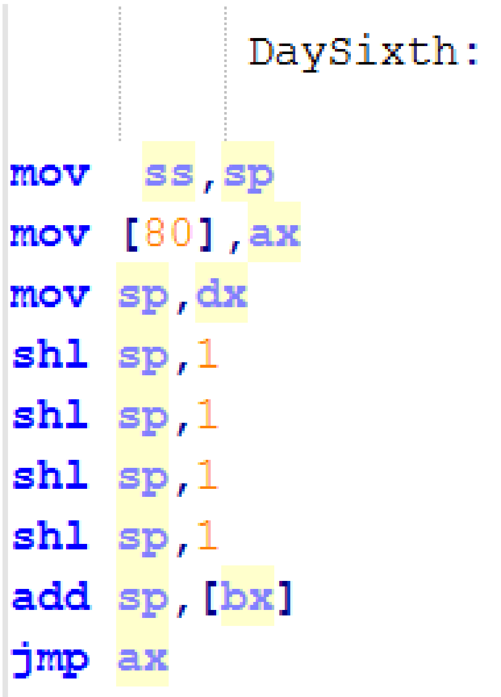
mov fs,bx =>

mov es, bx

#3 - Problems with old debuggers

Apocalypse CGX4

- Targeting flaws that can be found in debuggers.
  - Example: debug.exe.
- SP (Stack Pointer) gets really small value like '4' -> debugger crush.

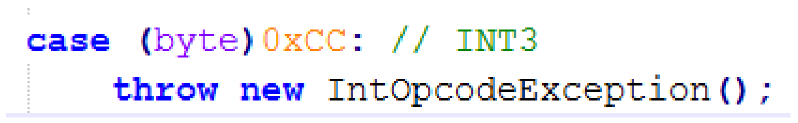


```
DaySixth:

mov  ss,sp
mov [80],ax
mov sp,dx
shl sp,1
shl sp,1
shl sp,1
shl sp,1
add sp,[bx]
jmp ax
```

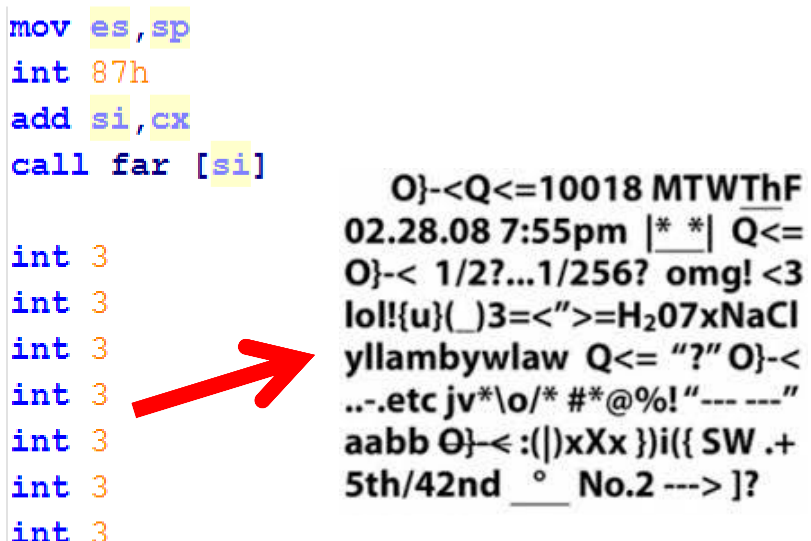
#4 - Random bits

Apocalypse CGX4

- Write multiple INT3 (‘CC’ , unsupported opcode) in places that are not part of the code flow.
- After compilation replace all occurrences of ‘CC’ to random bits (Hex Editor/script).
- For example F1, D6 etc.



```
case (byte)0xCC: // INT3
    throw new IntOpcodeException();
```



```
mov es,sp
int 87h
add si,cx
call far [si]

int 3
int 3
int 3
int 3
int 3
int 3
int 3
```

O}-<Q<=10018 MTWThF
02.28.08 7:55pm |* *| Q<=
O}-< 1/2?...1/256? omg! <3
lol!{u}(_)3=<”>=$H_2$07xNaCl
yllambywlaw Q<= “?” O}-<
..-.etc jv*\o/* #*@%! “--- ---”
aabb O}-< :(|)xXx })i({ SW .+
5th/42nd _°_ No.2 --->]?

#5 – XORing the code

XLII
CGX5

- Taking survivor's body and **generate two binary strings** - XOR of them will be the body of the original survivor.
- **During runtime** every survivor copy his part to the shared memory and they **calculate XOR of the two parts** before it is run by the survivor.
- Also **Smart bombing protection**..

#6 – Copy of a zombie

HutsHuts
CGX3

- Copy zombie into our survivors so **others will pwn a fake zombie** instead of the real one.
- Cons: valuable space is wasted.

#7 – Different Versions

All survivors
CGX5

- Getting zombies from the organizers.

- 1st round (remote) – 25%
- 2nd round (Face-2-face) – 25% 09:00 AM

- 3rd round (Face-2-face) – 50% 12:01PM
- Top 4 survivors get to the final.
- Final -> Winners!

Let them reverse vulnerable, lame version

Keep all your good stuff for this version

That's what happens to a team that achieves 1st place before the final round..

Detect Relationship

1 – Generating ASM instruction trace

1	PUSH
2	MOV
..	
17	MOV
18	CALL
..	
34	MOV
35	CALL
...	
42	MOV
43	CALL

1

2

3

2 –ASM counts

$$\begin{array}{c|ccc} & \text{PUSH} & \text{MOV} & \text{CALL} \\ \hline \text{PUSH} & 0 & 0 & 0 \\ \text{MOV} & 0 & 0 & \boxed{3} \\ \text{CALL} & 0 & 0 & 0 \end{array}$$

3 –Probabilities

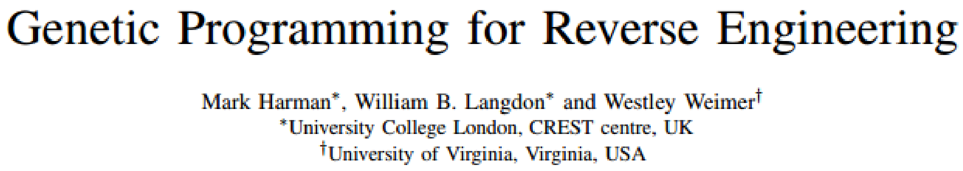
$$\begin{array}{c|ccc} & \text{PUSH} & \text{MOV} & \text{CALL} \\ \hline \text{PUSH} & 0 & 0 & 0 \\ \text{MOV} & 0 & 0 & \boxed{1/6} \\ \text{CALL} & 0 & 0 & 0 \end{array}$$

4 – Weighted directed graph for code

$$\begin{tikzcd} \text{Successor} \arrow[r] & \text{MOV} \arrow[r, "1/6"] & \text{CALL} \end{tikzcd}$$

5 – Weighted directed graph for code

$$\texttt{score}(A, B) = \frac{1}{N^2}\left(\sum_{i,j=0}^{N-1} |a_{ij} - b_{ij}|\right)^2$$


Genetic Programming for Reverse Engineering

Mark Harman*, William B. Langdon* and Westley Weimer†
*University College London, CREST centre, UK
†University of Virginia, Virginia, USA

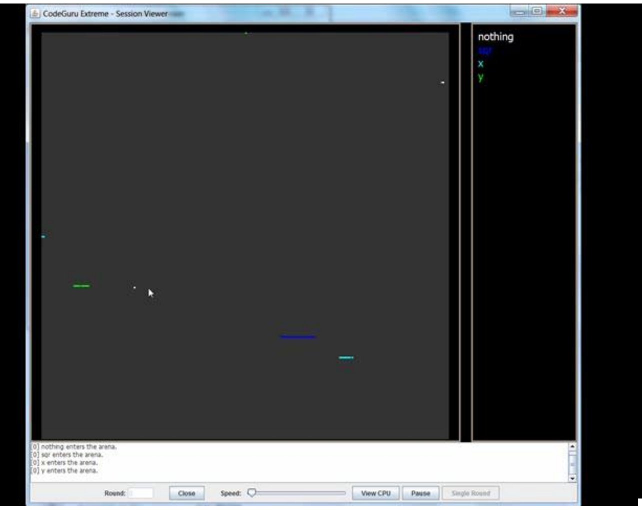
Genetic Programming

darwin8086
genetic 8086 program generator

- **A lot of work was done on RedCode**
  - John Perry , Jason Boar, Ryan Colman, Wilkies Benchmark, Dave Hillis and others.
- **One effort was done on CoreWars8086.**
  - Darwin8086.
- Gen = Warrior = String 1-512 bytes.
- Chromosome
  - Bit, Command, Meta-command, Combination,..
- Fitness function – Endogeny, Exogeny.

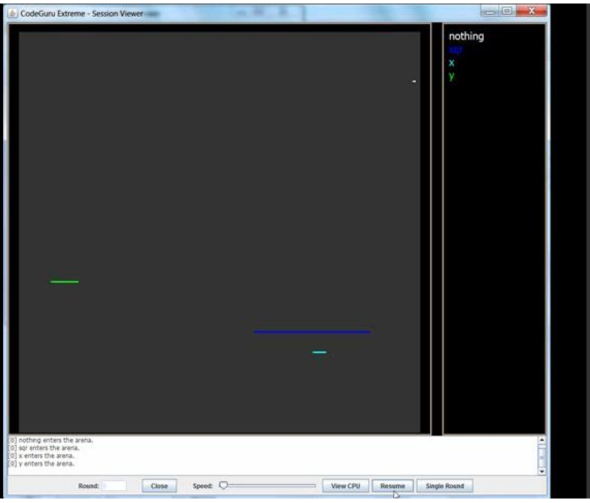
Graphical Survivors (Make Love Not War)

2D



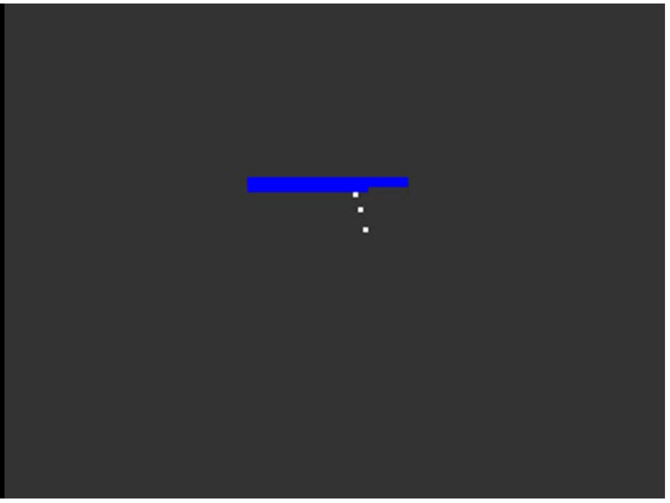
```
[0] nothing enters the arena.
[0] sqr enters the arena.
[0] x enters the arena.
[0] y enters the arena.
```

3D



```
[0] nothing enters the arena.
[0] sqr enters the arena.
[0] x enters the arena.
[0] y enters the arena.
```

BALL



SIR

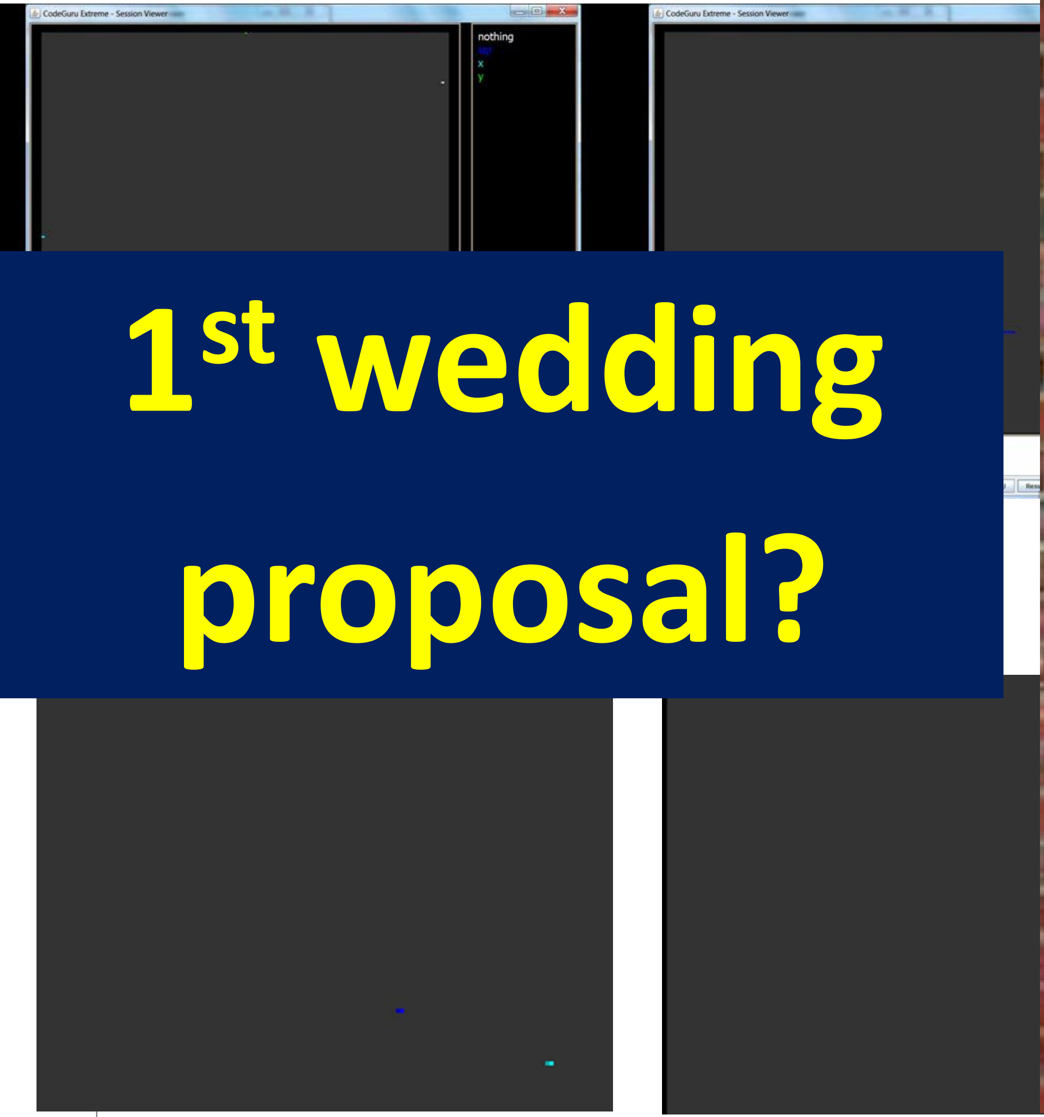
S

HALF

Graphical Survivors (Make Love Not War)

2D

3D


CodeGuru Extreme - Session Viewer
nothing
x
y
CodeGuru Extreme - Session Viewer

1st wedding proposal?

AVG

Future? Improvements?

ARM

2014
FIFA WORLD CUP
Brasil

KHANACADEMY

This is how can we add ‘hardware hacking’..

#@&%*^@& !!!!!!!
I knew I should
stick to PHP !!!

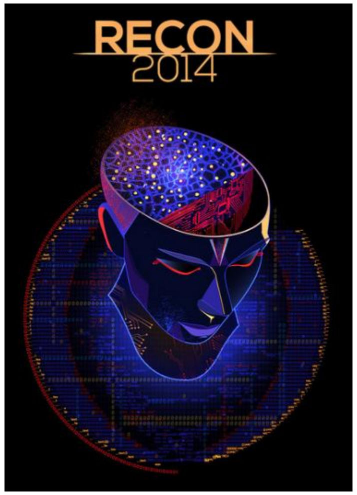
Scrum? Agile?

Q & A / Feedback

```
+---------------------------------------------------------+
|                                                         |
|             +----------+                                |
|             + Research |<-----------------+             |
|             +----------+                  |             |
|                  +                        |             |
|                  |                        |             |
|                  v                        |             |
|           +---------------+               |             |
|     +----+| Submit to CFP |               |             |
|     |     +---------------+               |             |
|     |                                     |             |
|     |                                     |             |
|     v                                     |             |
|  +------+     +-------------+      +--------+---------+|
|  | Meet |+--->| Share Ideas |---->| Create New Ideas ||
|  +------+     +----+--------+      +--------+---------+|
|     ^             |                       |             |
|     |             |                       |             |
|     |             |                       |             |
|     +-------------+<----------------------+             |
+---------------------------------------------------------+
                            |
                           ???
                            |
                            v
                  +--------------------+
                  |  Change the world  |
                  +--------------------+
```

Thank you! Merci!

Contact: Elad.Shapira@avg.com
eladexposed@gmail.com



ACK

- Hugo, Sam, Elizabeth and the ReCon team!!!
- Dr. Oded Margalit, Assaf Nativ, Ange Albertini, ShiftReduce, SonOfLilit, Danny Leshem,
  DualCore and Others..
- AVG, Oren Barad & The team.
- My (brave) Wife & kids.
- **300 Ninjas & Reversers..**