

# Hacking with Assembly

Τμήμα μηχανολόγων μηχανικών  
21 Νοεμβρίου 2012

Γκανετσίδης Κωνσταντίνος-Ραφαήλ  
Επιβλέπων Καθηγητής: Δασυγένης Μηνάς  
Διάλεξη 5λεπτών για το μάθημα Αρχιτεκτονικής ΗΥ

# Τι είναι;

- Assembly hacking/ASM Hacking

Τροποποίηση του source code μιας εφαρμογής



- ROM Hacking

Assembly Hacking στον τομέα του gaming



# Απαιτήσεις και δυσκολίες

- Απαιτήσεις
  - Λεπτομερή γνώση γλώσσας Assembly
  - Λεπτομερή γνώση της εσωτερικής λειτουργίας του προγράμματος
  
- Δυσκολίες
  - Έλλειψη συγκεκριμένου μοτίβου ενεργειών
  - Κάθε σύστημα απαιτεί λεπτομερή γνώση διαφορετικής γλώσσας Assembly
    - » Π.χ. [MIPS \(N64, PS1, PS2\)](#)
      - [M68000 \(Genesis\)](#)
      - [Z80 \(Game Boy\)](#)
      - [65c816 \(SNES\)](#)
      - [6502 \(Atari, NES, PC Engine\)](#)

# Δυσκολίες(συνέχεια)

- Εύρεση ζητούμενου σημείου κώδικα προς τροποποίηση
  - Πλήθος γραμμών του κώδικα
  - Κρυπτογράφηση του κώδικα από τους Developers για την αποτροπή της τροποποίησης του
- ROM Expansion(σε συγκεκριμενους τομείς του game hacking)
  - Το μέγεθος του ROM Image είναι καθορισμένο με αποτέλεσμα να μην μπορεί θεωρητικά να προσθεθεί επιπλέον περιεχόμενο στο παιχνίδι.
    - Η διαδικασία του ROM Expansion ήταν σχεδόν αδύνατη στην Consola NES και απαιτούσε ιδιαίτερα δύσκολες μετατροπές των mapper.
    - Στην κονσόλα SNES έπρεπε να ξαναγραφούν μεγάλα μέρη του κώδικα ώστε να δημιουργηθεί χώρος για τα καινούργια.
    - Στην κονσόλα Gameboy η υπάρχουσα μνήμη ξεπερνούσε έως και 17 φορές την χρησιμοποιούμενη και που καθιστούσε αρκετά εύκολο το ROM Expansion

# Γιατί Assembly;

- Έλλειψη του source code
  - Λόγω μη ελεύθερης διανομής του από τους Developers
- Εφικτοί τρόποι ανάκτησης του source code σε γλώσσα assembly
  - Reverse engineering(decompile-modify-compile)
    - Disassemblers: Idapro,Interactive Disassembler etc.
    - Debuggers: OllyDbg,WinDbg,Tseach,Cheat Engine etc.

# Παραδείγματα

- Always Valid Registration KEY

Αλλαγή της εντολής που ελέγχει αν ικανοποιείται η συνθήκη έτσι ώστε να πραγματοποιείται ανεξάρτητα από αυτήν.

```
JNZ SHORT 00450C56
```



```
JMP SHORT 00450C56
```

# Παραδείγματα(Code injection)

Εύρεση των ζητούμενων εντολών στον κώδικα(εδώ καταμετρηση πόντων/υπολογισμός τελικού σκορ στο παιχνίδι 3D Pinball των Windows).

The screenshot displays a Windows debugger interface with three main windows:

- Memory Viewer - Currently debugging thread 2054:** Shows assembly code for the thread. The instruction at address 0101755D is highlighted: `add [eax],2`. Other instructions include `cmp [eax],edx`, `jmp 01017563`, `mov [eax],edx`, `mov eax,[ecx+0000012E]`, `mov esi,[eax+4+01024756]`, `imul esi,edi`, `add esi,[ecx-0000012A]`, `lea eax,[ecx+52]`, `mov edx,[eax]`, `cmp edx,3994CA00`, `jmp 01017593`, `add edx,C4653600`, `inc [ecx+56]`, `mov [eax],edx`, `push [eax]`, `push [ecx+32]`, `call 01013C89`, `pop edi`, `mov eax,esi`, `pop esi`, `pop ebp`, and `ret 0004`.
- Registers:** Shows the state of CPU registers. EAX is 0254A2C2, EBX is 0256AA58, ECX is 0254A270, EDI is 00002710, and others are zero.
- Memory Dump:** Shows a memory dump for the module 'pinball.exe' starting at address 0104D000. The dump includes a header with 'Protect Write Copy Base=0104D000 Size=7000 Module=pinball.exe' and a table of memory addresses and values.

# Εισαγωγή εμβόλιμου κώδικα για την τροποποίηση της καταμέτρησης πόντων.

The screenshot displays a debugger interface with two main windows: Memory Viewer and Auto Assembler.

**Memory Viewer - Currently debugging thread 2054**

Address	Bytes	Opcode	Comment
0101755D			
0101755D	39 10	cmp [eax,edx]	
0101755F	7E 02	jle 01017563	
01017561	09 10	mov [eax,edx]	
01017563	0E 01 26010000	mov eax,[ecx+00000126]	
01017569	0E 34 05 59470201	mov esi,[eax*4+01024750]	
01017570	0FAF F7	imul esi,edi	
01017573	03 B1 2A010000	add esi,[ecx-0000012A]	
01017579	0D 41 52	lea eax,[ecx+52]	
0101757C	01 30	add [eax,esi]	
0101757E	0E 10	mov edx,[eax]	
01017580	01 FA 00CA9A3D	cmp edx,3B9ACA00	0.00
01017586	7E 0B	jle 01017593	
01017588	01 C2 003665C4	add edx,C4653600	-916.04
0101758E	FF 41 56	inc [ecx+56]	
01017591	09 10	mov [eax,edx]	
01017593	FF 30	push [eax]	
01017595	FF 71 32	push [ecx+32]	
01017598	E9 ECC6FFFF	call 01013C89	
0101759D	5F	pop ed	
0101759E	0B C6	mov eax,esi	
010175A0	5E	pop esi	
010175A1	5D	pop ebp	
010175A2	C2 0400	ret 0004	4

**Auto Assembler**

```
4  label(exit)
5
6  newmem:
7  push eax
8  push edx
9  push ebx
10 xor edx,edx
11 mov eax,esi
12 mov ebx,64 //100 DEC
13 div ebx
14 mov esi,eax //EAX=ESI/100
15 pop ebx
16 pop edx
17 pop eax
18
19 originalcode:
20 add [eax],esi
21 mov edx,[eax]
22 cmp edx,3B9ACA00
23
24
25 exit:
26 jmp returnhere
27
28 "pinball.exe"+1757C:
29 jmp newmem
30 nop
```

**Memory Dump**

Protect	Write	Copy	Base=0104D000	Size=7000	Module=pinball.exe	hex												
address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	0123456789ABCDEF	
0104D000	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0104D010	02	00	00	00	58	00	00	80	03	00	00	00	98	00	00	80	...	X
0104D020	04	00	00	00	58	02	00	80	05	00	00	00	98	02	00	80	...	X
0104D030	06	00	00	00	C8	03	00	80	0A	00	00	00	18	06	00	80	...	X
0104D040	0E	00	00	00	58	06	00	80	10	00	00	00	98	06	00	80	...	X
0104D050	18	00	00	00	D8	06	00	80	00	00	00	00	00	00	00	00	...	X
0104D060	00	00	00	00	01	00	00	00	18	07	00	80	70	00	00	80	...	X
0104D070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	...	X
0104D080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...	X
0104D090	09	04	00	00	00	00	00	00	20	70	03	00	A8	19	01	00	...	X
0104D0A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	...	X
0104D0B0	02	00	00	00	18	01	00	80	03	00	00	00	40	01	00	80	...	X
0104D0C0	04	00	00	00	68	01	00	80	05	00	00	00	90	01	00	80	...	X
0104D0D0	06	00	00	00	B8	01	00	80	07	00	00	00	80	01	00	80	...	X
0104D0E0	08	00	00	00	08	02	00	80	09	00	00	00	30	02	00	80	...	X
0104D0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01	00	...	X
0104D100	09	04	00	00	08	01	00	00	04	D8	04	00	68	06	00	00	...	X



# Αποθήκευση νέου κώδικα και επισκόπηση αποτελεσμάτων

**3D Pinball for Windows - Space Cadet**

Game Options Help

3D Pinball  
**Space Cadet**

BALL 1

1 300

Skill Shot  
300

Hit Mission  
Targets To  
Select Mission

**Memory Viewer - Running**

File Search View Debug Tools Kernel

Address	Bytes	Disassembly
0101755D	39 10	
0101755F	7E 02	
01017561	69 10	
01017563	6B 01	
01017569	6B 34	
01017570	0FAF	
01017573	03 B1	
01017579	6D 41	
0101757C	E9 2F	
01017581	90	
01017582	90	
01017583	90	
01017584	90	
01017585	90	
01017586	7E 0B	
01017588	61 C2	
0101758C	FF 41	inc [ecx+30]
01017591	69 10	mov [eax,edx]
01017593	FF 30	push [eax]
01017595	FF 71 32	push [ecx+32]
01017598	E8 ECC6FFFF	call 01013C09
0101759D	5F	pop edi
0101759E	6B C6	mov eax,esi

copy memory

Protect	Write	Copy	Base=0104D000	Size=7000	Module=pinball.exe
0104D000	00	00	00	00	00
0104D010	02	00	00	58	00
0104D020	04	00	00	58	02
0104D030	06	00	00	C8	03
0104D040	0E	00	00	58	06
0104D050	18	00	00	08	06
0104D060	00	00	00	01	00
0104D070	00	00	00	00	00
0104D080	09	04	00	88	00
0104D090	00	00	00	00	00
0104D0A0	00	00	00	09	00
0104D0B0	02	00	00	18	01
0104D0C0	04	00	00	68	01
0104D0D0	06	00	00	98	01
0104D0E0	08	00	00	08	02
0104D0F0	00	00	00	00	00
0104D100	09	04	00	08	01

**Registers:**

Register	Value	Flags
EAX	0254A2C2	CF 1
EBX	0256AA58	PF 1
ECX	0254A270	AF 0
EDX	000085CA	ZF 0
ESI	00002710	SF 1
EDI	00002710	DF 0
EBP	000CFC2C	OF 0
ESP	000CFC18	
RIP	01013C89	

**Segment Registers:**

Register	Value
CS	0023
SS	002B
DS	002B
ES	002B
FS	0053
GS	002B

**Disassembly:**

Address	DWORD	Value
000CFC18(esp+0)	0101759D	(pointer)pinball.exe+1759D
000CFC1C(esp+4)	0248DD64	(pointer)0248DD64
000CFC20(esp+8)	000085CA	(dword)000085CA(34250)
000CFC24(esp+C)	00000000	(dword)00000000(0)
000CFC28(esp+10)	0256AA58	(pointer)0256AA58
000CFC2C(esp+14)	000CFC90	(pointer)000CFC90
000CFC30(esp+18)	0100CEE2	(pointer)pinball.exe+CEE2
000CFC34(esp+1C)	00002710	(dword)00002710(10000)
000CFC38(esp+20)	01024104	(pointer)pinball.exe+24104
000CFC3C(esp+24)	BECA5590	(float)0.40
000CFC40(esp+28)	3F2FF7D8	(float)0.69
000CFC44(esp+2C)	00000000	(dword)00000000(0)
000CFC48(esp+30)	000CFC58	(pointer)000CFC58

# Συμπεράσματα/Μελλοντικές προεκτάσεις

- What a man can do, another man can undo, redo, do better or fail trying
- Η χρήση του Assembly Hacking δύσκολα θα ξεπεραστεί ή θα αντικατασταθεί

# Βιβλιογραφία:

- <http://www.criticalsecurity.net/index.php/topic/29278-learning-asm-the-hacker-point-of-view/>
- <http://www.romhacking.net/dictionary/?page=dictionary#term2>
- [http://romhack.wikia.com/wiki/Assembly\\_hacking](http://romhack.wikia.com/wiki/Assembly_hacking)
- <http://www.happyhacker.org/indexb.shtml>
- [http://en.wikipedia.org/wiki/ROM\\_hacking#ROM\\_expansion](http://en.wikipedia.org/wiki/ROM_hacking#ROM_expansion)
- [http://ffhacktics.com/wiki/ASM\\_Hacking](http://ffhacktics.com/wiki/ASM_Hacking)

## Other Links & Guides:

- <http://www.securitytube.net/video/208>
- [http://www.youtube.com/watch?v=Nshy5XMI\\_FI](http://www.youtube.com/watch?v=Nshy5XMI_FI)
- <http://dunyainfocom.blogspot.gr/2011/10/reverse-engineering-hacking-tutorial.html>