

Practical Reversing (I)

Harsimran Walia



www.SecurityXploded.com

Disclaimer

The Content, Demonstration, Source Code and Programs presented here is "AS IS" without any warranty or conditions of any kind. Also the views/ideas/knowledge expressed here are solely of the trainer's only and nothing to do with the company or the organization in which the trainer is currently working.

However in no circumstances neither the trainer nor SecurityXploded is responsible for any damage or loss caused due to use or misuse of the information presented here.

Acknowledgement

- Special thanks to **null & Garage4Hackers** community for their extended support and cooperation.
- Thanks to all the **Trainers** who have devoted their precious time and countless hours to make it happen.

Reversing & Malware Analysis Training

This presentation is part of our **Reverse Engineering & Malware Analysis** Training program. Currently it is delivered only during our local meet for FREE of cost.



For complete details of this course, visit our [Security Training page](#).

whoami

harsimranwalia.info

- ◎ **b44nz0r**
- ◎ Research Scientist @ McAfee Labs
- ◎ Mechanical Engineer @IIT Delhi
- ◎ Independent Security Researcher
- ◎ RE, Exploit Analysis/Development, Malware Analysis

Twitter : [b44nz0r](https://twitter.com/b44nz0r)

Email : walia.harsimran@gmail.com

Course Q&A

- ⦿ Keep yourself up to date with latest security news
 - <http://www.securityphresh.com>

- ⦿ For Q&A, join our mailing list.
 - <http://groups.google.com/group/securityxploded>

Outline

- ⦿ Break Point
- ⦿ Debug Registers
- ⦿ Flags
- ⦿ API Help

Types of Breakpoints

- Software
- Hardware
- Memory

Breakpoint

- ⦿ Software breakpoints are set by replacing the instruction at the target address with 0xCC (INT3/ Breakpoint interrupt)
- ⦿ Hardware breakpoints are set via debug registers. Only 4 hardware breakpoints can be set
- ⦿ Debug registers:
 - 8 debug registers present
 - DR0 – DR3 : Address of breakpoint
 - DR6 : Debug Status – To determine which breakpoint is active
 - DR7 : Debug Control – Flags to control the breakpoints such as break on read or on-write
- ⦿ Debug registers are not accessible in Ring 3

Hardware Breakpoints

Immunity Debugger - crackme.exe - [CPU - main thread, module crackme]

File View Debug Plugins ImmLib Options Window Help Jobs

Immunity Consulting Services Manager

```

0040130A . 890424 MOV DWORD PTR SS:[ESP],EAX
0040130D . E8 6E060000 CALL <JMP.&msvcrt.strlen>
00401312 . 8945 AC MOV DWORD PTR SS:[EBP-54],EAX
00401315 . 8045 D8 LEA EAX,DWORD PTR SS:[EBP-28]
00401318 . 890424 MOV DWORD PTR SS:[ESP],EAX
0040131B . E8 60060000 CALL <JMP.&msvcrt.strlen>
00401320 . 8945 A8 MOV DWORD PTR SS:[EBP-58],EAX
00401323 . 837D AC 04 CMP DWORD PTR SS:[EBP-54],4
00401327 . 7E 06 JLE SHORT crackme.00401331
00401329 . 837D AC 09 CMP DWORD PTR SS:[EBP-54],9
0040132D . 7F 02 JG SHORT crackme.00401331
0040132F . EB 22 JMP SHORT crackme.00401353
00401331 > C70424 303040 MOV DWORD PTR SS:[ESP],crackme.00403030
00401338 . E8 63060000 CALL <JMP.&msvcrt.printf>
0040133D . C70424 4B3040 MOV DWORD PTR SS:[ESP],crackme.0040304B
00401344 . E8 57060000 CALL <JMP.&msvcrt.printf>
00401349 . E8 22070000 CALL crackme.00401A70
0040134E . E9 D7000000 JMP crackme.0040142A
00401353 > 8B45 A8 MOV EAX,DWORD PTR SS:[EBP-58]
00401356 . 3B45 AC CMP EAX,DWORD PTR SS:[EBP-54]
00401359 . 74 22 JE SHORT crackme.0040137D
0040135B . C70424 5F3040 MOV DWORD PTR SS:[ESP],crackme.0040305F
00401362 . E8 39060000 CALL <JMP.&msvcrt.printf>
00401367 . C70424 4B3040 MOV DWORD PTR SS:[ESP],crackme.0040304B
0040136E . E8 2D060000 CALL <JMP.&msvcrt.printf>
00401373 . E8 F8060000 CALL crackme.00401A70
00401378 . E9 AD000000 JMP crackme.0040142A
0040137D . 8B45 AC MOV EAX,DWORD PTR SS:[EBP-54]
00403030=crackme.00403030 (ASCII 0A,"Invalid! U")
Jumps from 00401327, 0040132D
    
```

Registers (FPU)

```

EAX 00000000
ECX 0022FFB8
EDX 7C90E4F4 ntdll.KiFastSystemCallRet
EBX 7FFDF000
ESP 0022FFC4
EBP 0022FFF0
ESI 00720065
EDI 00670067
EIP 00401220 crackme.<ModuleEntryPoint>

C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_NO_IMPERSONATION_TOKEN (0000051D)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty 0.00000000000000000000
ST1 empty 0.00000000000000000000
ST2 empty 0.00000000000000000000
ST3 empty 0.00000000000000000000
ST4 empty 0.00000000000000000000
ST5 empty 0.00000000000000000000
ST6 empty 1.96875000000000000000
ST7 empty 1.2519775166695187000e-312
    
```

Hardware breakpoints

#	Base	Size	Stop on	Follow	Delete
1	004013F3		Execute	Follow 1	Delete 1
2	00401331		Execute	Follow 2	Delete 2
3				Follow 3	Delete 3
4				Follow 4	Delete 4

OK

```

0022FFC4 7C817067 gpui! RETURN to kernel!32.7C817067
0022FFC8 00670067 g.g.
0022FFCC 00720065 e.r.
0022FFD0 7FFDF000 "=Δ
0022FFD4 805448FD "KTC
0022FFD8 0022FFC8 "
0022FFDC 866F2AC0 "wō
0022FFE0 FFFFFFFF End of SEH chain
0022FFE4 7C809AC0 "ū; SE handler
0022FFE8 7C817070 ppui! kernel!32.7C817070
0022FFEC 00000000 ....
0022FFF0 00000000 ....
0022FFF4 00000000 ....
0022FFF8 00401220 #0. crackme.<ModuleEntryPoint>
0022FFFC 00000000 ....
    
```

Address	Hex dump	ASCII
00402000	FF FF FF 00 00 00 00
00402008	00 00 00 00 00 00 00
00402010	00 40 00 00 00 00 00	.@.....
00402018	00 00 00 00 00 00 00
00402020	BC 1A 40 00 00 00 00	" +@.....
00402028	00 00 00 00 00 00 00
00402030	00 00 00 FF FF FF FF
00402038	00 00 00 FF FF FF FF
00402040	00 00 00 00 00 00 00
00402048	00 00 00 00 00 00 00
00402050	00 00 00 00 00 00 00
00402058	00 00 00 00 00 00 00
00402060	00 00 00 00 00 00 00
00402068	00 00 00 00 00 00 00
00402070	00 00 00 00 00 00 00
00402078	00 00 00 00 00 00 00
00402080	00 00 00 00 00 00 00

Memory

- To access memory, need of permissions
- Lots of permissions
 - PAGE_GUARD
 - PAGE_READWRITE
 - PAGE_EXECUTE
 - PAGE_EXECUTE_READ
- To set memory breakpoint,
 - the permissions of that memory region is set to PAGE_GUARD
 - whenever an access is made to that memory STATUS_GUARD_PAGE_VIOLATION exception is raised
 - On getting the exception the debugger changes the permission back to the original
 - Notifies the user of the breakpoint

Breakpoints

DEMO

Flags (Eflags Register)

- 1 register – 32 bits
- Each bit signifies a flag
- Few important ones are:

Bit #	Abbreviation	Description
0	CF	Carry flag
2	PF	Parity flag
4	AF	Adjust flag
6	ZF	Zero flag
7	SF	Sign flag
8	TF	Trap flag (single step)
9	IF	Interrupt enable flag
11	OF	Overflow flag

Flags Demystified

- ⦿ **Carry flag** is used to indicate when an arithmetic carry or borrow has been generated out of the most significant ALU bit position
- ⦿ **Parity flag** indicates if the number of set bits is odd or even in the binary representation of the result of the last operation
- ⦿ **Adjust flag** is used to indicate when an arithmetic carry or borrow has been generated out of the 4 least significant bits
- ⦿ **Zero Flag** is used to check the result of an arithmetic operation, including bitwise logical instructions. It is set if an arithmetic result is zero, and reset otherwise
- ⦿ **Sign flag** is used to indicate whether the result of last mathematic operation resulted in a value whose most significant bit was set
- ⦿ A **trap flag** permits operation of a processor in single-step mode
- ⦿ **Overflow flag** is used to indicate when an arithmetic overflow has occurred in an operation, indicating that the signed two's-complement result would not fit in the number of bits used for the operation

Basic Reversing Techniques

- ⦿ Check for readable strings
- ⦿ Import table (IAT) for imported Windows API
- ⦿ Setting breakpoint on interesting API
- ⦿ Single stepping

Variables

- Found under **Names** tab

- L - library function
- F - regular function
- C - instruction
- A - ascii string
- D - data
- I - imported name

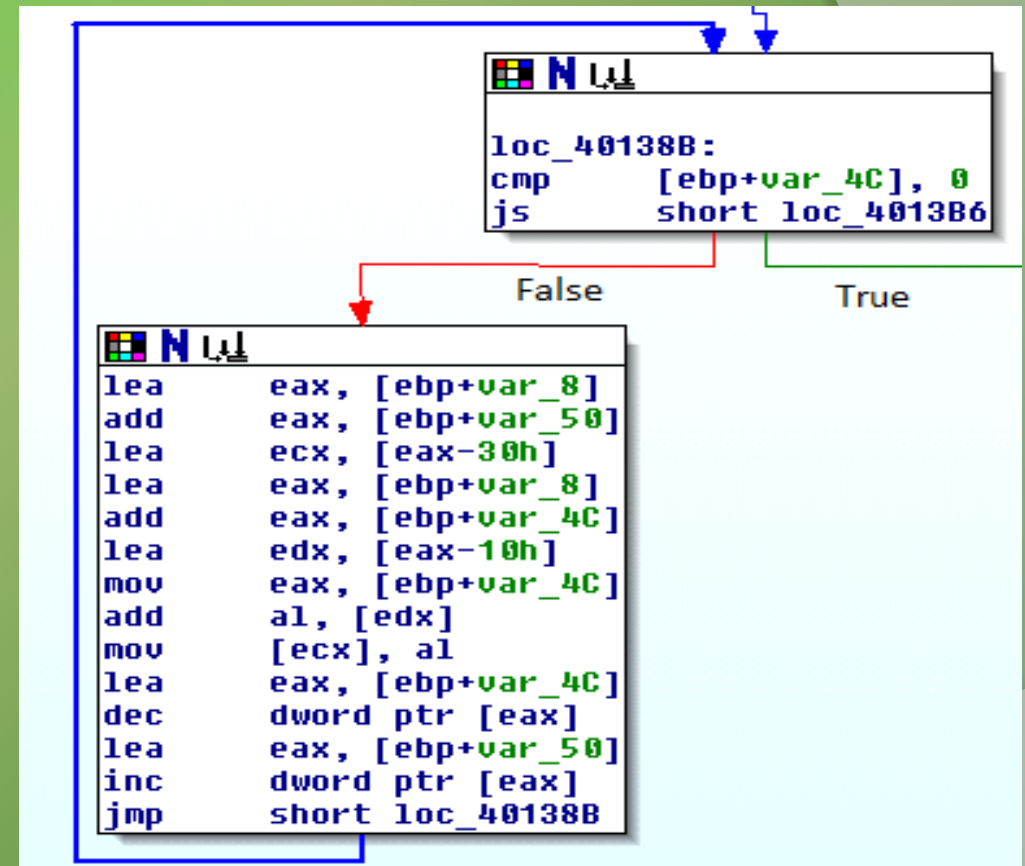
Contd..

```
shr     eax, 4
shl     eax, 4
mov     [ebp+var_6C], eax
mov     eax, [ebp+var_6C]
call    ___chkstk
call    __main
add     dword_402000, 5
mov     dword_ptr [esp], offset aCrackme ; "#Crackme\n\n"
call    printf
```

- ⦿ Global variables are generally `dword_<address>`
 - `dword_402000` – as shown in image
- ⦿ Local variables are of the form `var_<offset>`
 - `var_6C` – as shown in image

Loop in IDA

- Red Line
 - If condition is false
 - (zero flag = 0)
- Green Line
 - If condition is true
 - (zero flag = 1)



Reversing a Simple Crackme

DEMO

[HTTP://GOO.GL/BICSX](http://goo.gl/BICSX)

Crackme Code

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{
    char a[10],b[10],c[10],d[10];
    int i,j,k,l,r,s;
    printf("#Crackme\n\n");
    printf("enter username:");
    scanf("%s",a);
    printf("enter password:");
    scanf("%s",b);
    k = strlen(a);
    l = strlen(b);
    if (k < 5 || k >= 10){
        printf("\nInvalid! Username Length\n");
        printf("\nHit Enter to Exit\n");
        getchar();
    } else {
        if (l != k){
            printf("\nInvalid! Password Length\n");
            printf("\nHit Enter to Exit\n");
            getchar();
        } else {
            i = k-1;
            j = 0;
            while (i >= 0){
                c[j] = a[i]+i;
                i--;
                j++;
            }
            c[j] = 0;
            r = strlen(c);
            if (r == l){
                i = strcmp(c,b);
                if (i == 0){
                    printf("\nCongratulations! You did it..\n");
                    printf("\nHit Enter to Exit\n");
                } else {
                    printf("\nAccess Denied! Wrong Password\n");
                    printf("\nHit Enter to Exit\n");
                }
            }
        }
    }
}
```

References

- [Complete Reference Guide for Reversing & Malware Analysis Training](#)

Thank You !



www.SecurityXploded.com