REVERSE ENGINEERING PROJECT

1. Introduction

Link: http://www.crackmes.de/users/vik3790/little_fish./
Tools used: IDA v6.9, Python 2.7

The executable has the following properties:

SHA256	6856D40D366A4E2E56214B66D2A71E16BF03A24FF79193C62C7E0D9DC2DF2E83
MD5	FC530BF50DD540084F08B7C66548AB0F
Type	Portable Executable (PE) Intel 80386
Platform	Windows
Language	C/C++

When executed, it shows a simple dialog as follows. However, there is no screen for input.

2. Main Window

a) Initially, check the main function and it is observed at the end of the function there is a call to 'WinMain' at offset 0x402016

```
.text:00401FF4 loc_401FF4:                              ; CODE XREF: _main+7E↑j
.text:00401FF4                 mov     dword ptr [esp], 0 ; lpModuleName
.text:00401FFB                 call    _GetModuleHandleA@4 ; GetModuleHandleA(x)
.text:00402000                 sub     esp, 4          ; Integer Subtraction
.text:00402003                 mov     [esp+0Ch], esi  ; nShowCmd
.text:00402007                 mov     [esp+8], ebx    ; lpCmdLine
.text:0040200B                 mov     dword ptr [esp+4], 0 ; hPrevInstance
.text:00402013                 mov     [esp], eax      ; hInstance
.text:00402016                 call    _WinMain@16     ; WinMain(x,x,x,x)
```

b) Now check the 'WinMain' function and observe that it creates a dialog box with 'DlgMain' passed as a dialog function.

```
.text:0040151D                 push    ebp
.text:0040151E                 mov     ebp, esp
.text:00401520                 sub     esp, 28h        ; Integer Subtraction
.text:00401523                 mov     eax, [ebp+hInstance]
.text:00401526                 mov     ds:_hInst, eax
.text:0040152B                 call    _InitCommonControls@0 ; InitCommonControls()
.text:00401530                 mov     eax, ds:_hInst
.text:00401535                 mov     dword ptr [esp+10h], 0 ; dwInitParam
.text:0040153D                 mov     dword ptr [esp+0Ch], offset __Z7DlgMainP6HWND__jjl@16 ; lpDialogFunc
.text:00401545                 mov     dword ptr [esp+8], 0 ; hWndParent
.text:0040154D                 mov     dword ptr [esp+4], 64h ; lpTemplateName
.text:00401555                 mov     [esp], eax      ; hInstance
.text:00401558                 call    _DialogBoxParamA@20 ; DialogBoxParamA(x,x,x,x,x)
.text:0040155D                 sub     esp, 14h        ; Integer Subtraction
.text:00401560                 leave                   ; High Level Procedure Exit
.text:00401561                 retn    10h             ; Return Near from Procedure
```

c) Now, observe that this function is a simple dialog with no input field. We can see that the left and the right clicks on the mouse have CounterL and CounterR associated with it. When the dialog is clicked, 'DlgMain' is called and one of the two variables CounterL or CounterR is updated. We also observe that each time one of these buttons is clicked, the function '_Z6Clicksv' is called.

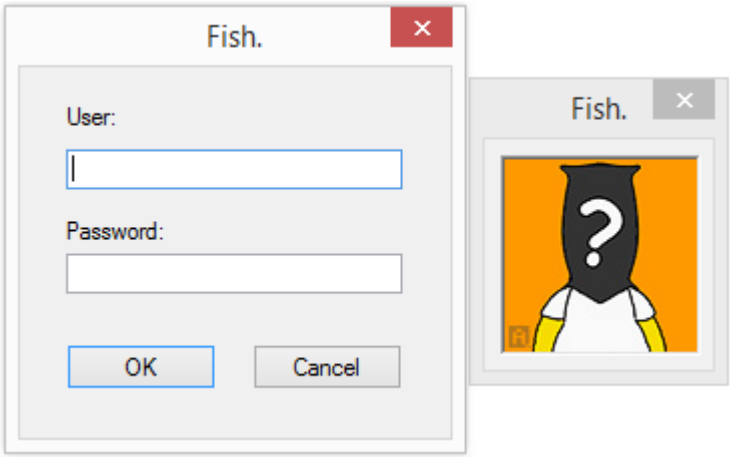
```
.text:004014AB loc_4014AB:                             ; CODE XREF: DlgMain(HWND__ *,uint,uint,long)+15↑j
.text:004014AB                 cmp     eax, WM_LBUTTONUP ; Compare Two Operands
.text:004014B0                 jz      short loc_4014BB ; Jump if Zero (ZF=1)
.text:004014B2                 cmp     eax, WM_RBUTTONUP ; Compare Two Operands
.text:004014B7                 jz      short loc_4014D2 ; Jump if Zero (ZF=1)
.text:004014B9                 jmp     short loc_401514 ; Jump
.text:004014BB ; ---------------------------------------------------------------------------
.text:004014BB
.text:004014BB loc_4014BB:                             ; CODE XREF: DlgMain(HWND__ *,uint,uint,long)+2A↑j
.text:004014BB                 mov     eax, ds:_CounterL
.text:004014C0                 inc     eax             ; Increment by 1
.text:004014C1                 mov     ds:_CounterL, eax
.text:004014C6                 call    __Z6Clicksv     ; Clicks(void)
.text:004014CB                 mov     eax, 1
.text:004014D0                 jmp     short locret_401519 ; Jump
.text:004014D2 ; ---------------------------------------------------------------------------
.text:004014D2
.text:004014D2 loc_4014D2:                             ; CODE XREF: DlgMain(HWND__ *,uint,uint,long)+31↑j
.text:004014D2                 mov     eax, ds:_CounterR
.text:004014D7                 inc     eax             ; Increment by 1
.text:004014D8                 mov     ds:_CounterR, eax
.text:004014DD                 call    __Z6Clicksv     ; Clicks(void)
.text:004014E2                 mov     eax, 1
.text:004014E7                 jmp     short locret_401519 ; Jump
```

d) Observe offset 0x40156F and 0x401579. This shows the number of clicks on the left mouse button and right mouse button respectively. This will result in another function called 'RegisterDlg' which displays the authentication dialog.

```
.text:00401564 __Z6Clicksv     proc near               ; CODE XREF: DlgMain(HWND__ *,uint,uint,long)+40↑p
.text:00401564                                         ; DlgMain(HWND__ *,uint,uint,long)+57↑p
.text:00401564                 push    ebp
.text:00401565                 mov     ebp, esp
.text:00401567                 sub     esp, 28h        ; Integer Subtraction
.text:0040156A                 mov     eax, ds:_CounterL
.text:0040156F                 cmp     eax, 3          ; Compare Two Operands
.text:00401572                 jnz     short locret_4015C2 ; Jump if Not Zero (ZF=0)
.text:00401574                 mov     eax, ds:_CounterR
.text:00401579                 cmp     eax, 2          ; Compare Two Operands
.text:0040157C                 jnz     short locret_4015C2 ; Jump if Not Zero (ZF=0)
.text:0040157E                 mov     eax, ds:_hInst
.text:00401583                 mov     dword ptr [esp+10h], 0 ; dwInitParam
.text:0040158B                 mov     dword ptr [esp+0Ch], offset __Z11RegisterDlgP6HWND__jjl@16 ; lpDialogFunc
.text:00401593                 mov     dword ptr [esp+8], 0 ; hWndParent
.text:0040159B                 mov     dword ptr [esp+4], 66h ; lpTemplateName
.text:004015A3                 mov     [esp], eax      ; hInstance
.text:004015A6                 call    _DialogBoxParamA@20 ; DialogBoxParamA(x,x,x,x,x)
.text:004015AB                 sub     esp, 14h        ; Integer Subtraction
.text:004015AE                 mov     ds:_CounterL, 0
.text:004015B8                 mov     ds:_CounterR, 0
.text:004015C2
.text:004015C2 locret_4015C2:                          ; CODE XREF: Clicks(void)+E↑j
.text:004015C2                                         ; Clicks(void)+18↑j
.text:004015C2                 leave                   ; High Level Procedure Exit
.text:004015C3                 retn                    ; Return Near from Procedure
.text:004015C3 __Z6Clicksv     endp
```

3. Authentication Dialog

a) After clicking on the left mouse button 3 times and right mouse button 2 times, we observe another dialog which asks for authentication details.



b) RegisterDlg

This is where we input the Username and Password. When the 'OK' button is pressed, the function '_Z8CheckingPcS_' is called with Username and Password as the arguments. The "Good Boy" or "Bad Boy" messages are shown depending on the value returned by '_Z8CheckingPcS_.' The function '_Z8CheckingPcS_' should return '1' to jump to "Good boy"

```
loc_4013BE:                 ; nMaxCount
mov     dword ptr [esp+0Ch], 0FFh
mov     dword ptr [esp+8], offset _Name ; lpString
mov     dword ptr [esp+4], 9C40h ; nIDDlgItem
mov     eax, [ebp+hwndDlg]
mov     [esp], eax          ; hDlg
call    _GetDlgItemTextA@16 ; GetDlgItemTextA(x,x,x,x)
sub     esp, 10h            ; Integer Subtraction
mov     dword ptr [esp+0Ch], 0FFh ; nMaxCount
mov     dword ptr [esp+8], offset _Password ; lpString
mov     dword ptr [esp+4], 9C42h ; nIDDlgItem
mov     eax, [ebp+hwndDlg]
mov     [esp], eax          ; hDlg
call    _GetDlgItemTextA@16 ; GetDlgItemTextA(x,x,x,x)
sub     esp, 10h            ; Integer Subtraction
mov     dword ptr [esp+4], offset _Password ; Password
mov     dword ptr [esp], offset _Name ; Name
call    __Z8CheckingPcS_ ; Checking(char *,char *)
cmp     eax, 1              ; Compare Two Operands
setz    al                  ; Set Byte if Zero (ZF=1)
test    al, al              ; Logical Compare
jz      short loc_401450 ; Jump if Zero (ZF=1)
```

```
mov     dword ptr [esp+0Ch], 0 ; uType
mov     dword ptr [esp+8], offset Caption ; "Good boy."
mov     dword ptr [esp+4], offset Text ; "Excellent job cracker!, now write a key"...
mov     eax, [ebp+hwndDlg]
mov     [esp], eax          ; hWnd
call    _MessageBoxA@16 ; MessageBoxA(x,x,x,x)
sub     esp, 10h            ; Integer Subtraction
jmp     short loc_401476 ; Jump
```

```
loc_401450:                 ; uType
mov     dword ptr [esp+0Ch], 0
mov     dword ptr [esp+8], offset aBadBoy_ ; "Bad boy."
mov     dword ptr [esp+4], offset aIDonTThinkSoTr ; "I don¦t think so, try harder"
mov     eax, [ebp+hwndDlg]
mov     [esp], eax          ; hWnd
call    _MessageBoxA@16 ; MessageBoxA(x,x,x,x)
sub     esp, 10h            ; Integer Subtraction
```

c) The main loop of the function '_Z8CheckingPcS_' is as shown below. We can observe the transformations that are applied to the first four characters of the username which is taken from the user as input.

These transformations can be simplified as follows:

C |= ord(Name[i]) << (3-i)*8

```
.text:004015DA loc_4015DA:                            ; CODE XREF: Checking(char *,char *)+48↓j
.text:004015DA                 mov     edx, [ebp+i]
.text:004015DD                 mov     eax, [ebp+Name]
.text:004015E0                 add     eax, edx       ; Add
.text:004015E2                 mov     al, [eax]
.text:004015E4                 movsx   eax, al        ; Move with Sign-Extend
.text:004015E7                 mov     [ebp+B], eax
.text:004015EA                 mov     eax, 3
.text:004015EF                 sub     eax, [ebp+i]   ; Integer Subtraction
.text:004015F2                 shl     eax, 3         ; Shift Logical Left
.text:004015F5                 mov     cl, al
.text:004015F7                 shl     [ebp+B], cl    ; Shift Logical Left
.text:004015FA                 mov     eax, [ebp+B]
.text:004015FD                 or      [ebp+C], eax   ; Logical Inclusive OR
.text:00401600                 inc     [ebp+i]        ; Increment by 1
.text:00401603
.text:00401603 loc_401603:                            ; CODE XREF: Checking(char *,char *)+14↑j
.text:00401603                 cmp     [ebp+i], 3     ; Compare Two Operands
.text:00401607                 setle   al             ; Set Byte if Less or Equal (ZF=1 | SF!=OF)
.text:0040160A                 test    al, al         ; Logical Compare
.text:0040160C                 jnz     short loc_4015DA ; Jump if Not Zero (ZF=0)
```

d) This is the code for the final transformation. The function simply computes an integer from the first four characters of the username and compares it with the password.

```
.text:0040160E                 mov     edx, [ebp+C]
.text:00401611                 mov     eax, edx
.text:00401613                 shl     eax, 1          ; Shift Logical Left
.text:00401615                 add     eax, edx        ; Add
.text:00401617                 lea     edx, ds:0[eax*4] ; Load Effective Address
.text:0040161E                 add     eax, edx        ; Add
.text:00401620                 mov     [ebp+C], eax
.text:00401623                 add     [ebp+C], 0FFh   ; Add
.text:0040162A                 mov     eax, [ebp+C]
.text:0040162D                 mov     [esp+8], eax
.text:00401631                 mov     dword ptr [esp+4], offset aIx ; "%IX"
.text:00401639                 mov     eax, [ebp+Name]
.text:0040163C                 mov     [esp], eax      ; LPSTR
.text:0040163F                 call    _wsprintfA      ; Call Procedure
.text:00401644                 mov     eax, [ebp+Password]
.text:00401647                 mov     [esp+4], eax    ; char *
.text:0040164B                 mov     eax, [ebp+Name]
.text:0040164E                 mov     [esp], eax      ; char *
.text:00401651                 call    _strcmp         ; Call Procedure
.text:00401656                 test    eax, eax        ; Logical Compare
.text:00401658                 jnz     short loc_401661 ; Jump if Not Zero (ZF=0)
```

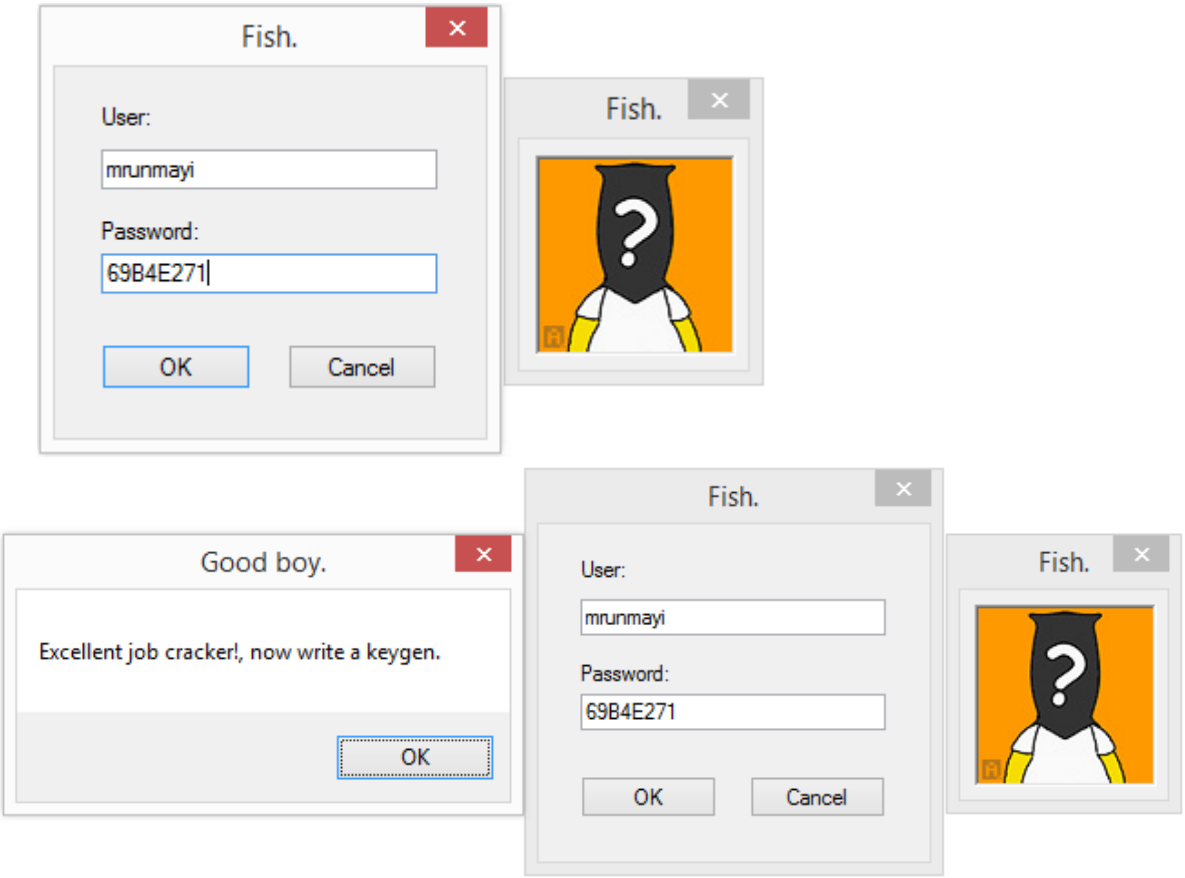
4. SOLUTION

Based on the understanding of the transformations, a password generator or keygen was developed. Below is the screenshot for the same.

```
mrunmayi@mrunmayi-VirtualBox: ~/Desktop
  GNU nano 2.2.6            File: fish_keygen.py

#! /usr/bin/python
#Mrunmayi Churi
#Keygen for http://crackmes.de/users/vik3790/little_fish./

import sys

def generate_key(username):
    C = 0
    m = min(len(username),4)

    for i in range (0, m):
        C |= ord(username[i]) << (3 - i) * 8
    C = (C*15 + 0xFF) & 0xFFFFFFFF
    return "%X" %C

if __name__ == "__main__":

    if len(sys.argv) !=2:
        print "Use the following format: %s username" % sys.argv[0]
        sys.exit(1)

print "The password is: %s" % generate_key(sys.argv[1])
```

```
mrunmayi@mrunmayi-VirtualBox: ~/Desktop
mrunmayi@mrunmayi-VirtualBox:~/Desktop$ python fish_keygen.py mrunmayi
The password is: 69B4E271
```

5. Test



References:
[1] https://www.hex-rays.com/products/ida/support/idadoc/
[2] http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html
[3] http://www.crackmes.de/users/vik3790/little_fish./solutions/rouse/browse/solution.txt