Krypton

Author : Karthik Selvaraj at Symantec Corporation, contact : neoxfx at gmail dot com

Krypton is an IDA Plugin that assists one in reverse engineering x86 binary executables, by executing a function from IDB (IDA database) using IDA's powerful Appcall feature.

krypton takes xrefs from a given function (say a possible decoder) to find all function calls to it and then parses and finds the parameters use (including prototype, no of arguments, and the arguments themselves) from instructions and uses them to execute the function using Appcall, this is most useful in analyzing a malware binary with encryption.

Features

- lists top referenced functions list to start analysis.
- Assists in identifying encryption/decryption sub-routine.
- Decrypts the encrypted strings/contents in a binary without a need to understand encryption method or an invasive debugging.
  thus helping in overall analysis of the binary being reverse engineered.

Installation

- To Install, Copy the plugin to IDA Plugin folder,
- Std path: "C:\Program Files\IDA\plugins\"
- NOTE: plugin requires IDA >= 5.6

Quick Start Guide

Issue **Ctrl+F8**

To Decrypt Strings

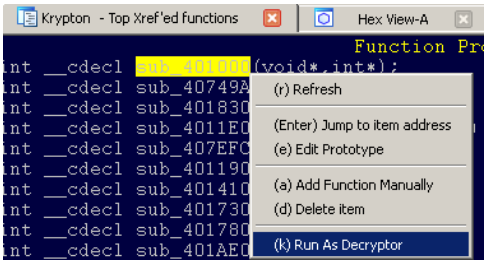
Say you see a function that looks like a string decryptor call, e.g.

```
mov     edx, [ebp+lpProcName]
push    edx
push    offset aTrgguernqpbagrkg ; "TrgGuernqPbagrkg"
call    sub_401000
add     esp, 8
mov     eax, [ebp+lpProcName]
push    eax             ; lpProcName
mov     ecx, [ebp+hModule]
push    ecx             ; hModule
call    ds:GetProcAddress
mov     dword_40E94C, eax
mov     edx, [ebp+lpProcName]
push    edx
push    offset aFrgguernqpbagrkg ; "FrgGuernqPbagrkg"
call    sub_401000
```

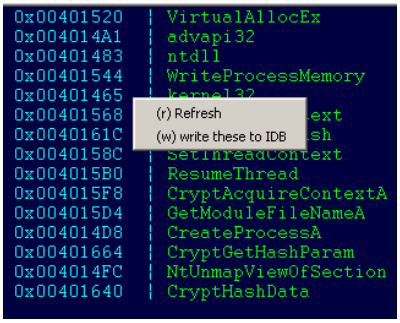
Hit Ctrl+F8

Krypton would list the top xref'ed functions, select suspected function, right click and say "(k) run as decryptor"

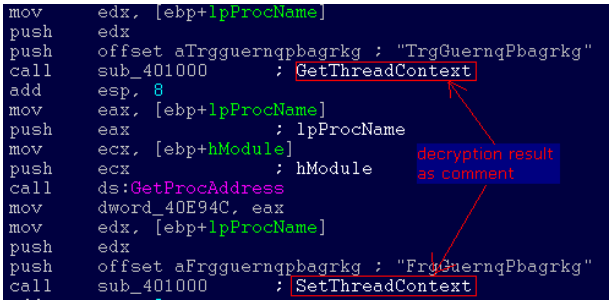
NOTE: If the analyzed file is a DLL, then IMAGE_FILE_DLL flag must be cleared from the characteristics entry in the PE header prior to running krypton. That is, bit 13 must be cleared in the characteristics WORD at offset 0x18 of the PE header. Be sure to reload the DLL in IDA prior to trying to run Krypton once the flag is cleared.



krypton decrypts it, you can right click and have it written as comment near call

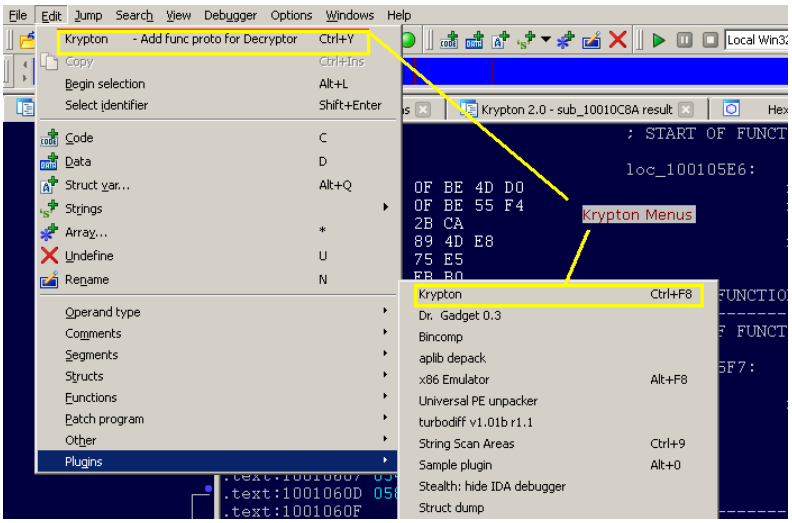


result of decryption is written as comment in respective places



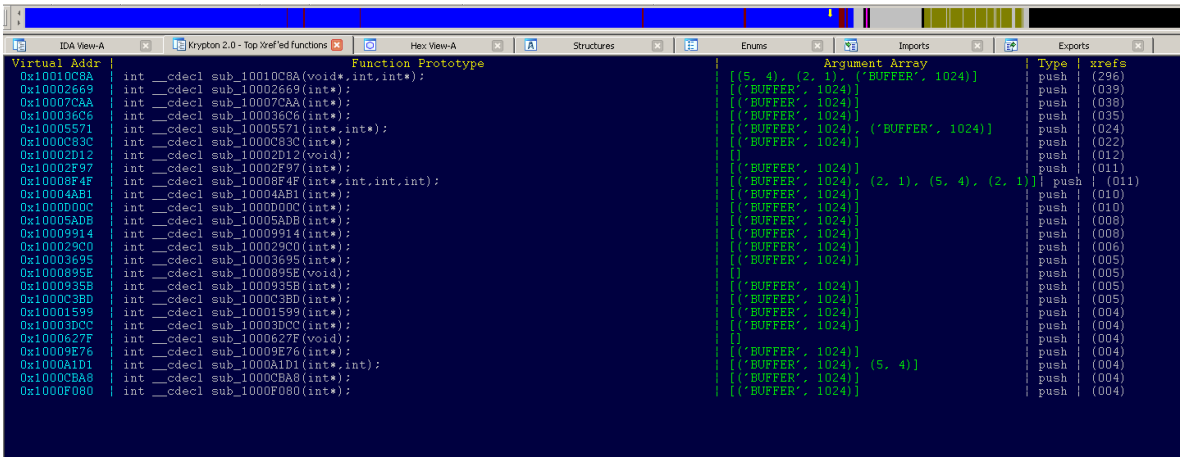
Detailed Usage Guide - Decryption

Once installed, krypton adds the below menus and HotKeys, to launch the Plugin use hotkey **Ctrl+F8**



Krypton waits for IDA Auto-Analysis to finish and then brings up a Krypton Plugin prototype view, which lists top Cross referenced functions in IDB

Usually decryptor functions are called in multiple places thus increasing their cross reference numbers, so one might possibly find the decryptor functions at the start of the
list and this is also a good starting point for analysis, the more one marks top referenced functions, the more clearer IDB gets.



Above figure shows top referenced functions. The 3rd column named "argument array" is a series of (argument_instruction_size, data_size) pairs

So for below example call

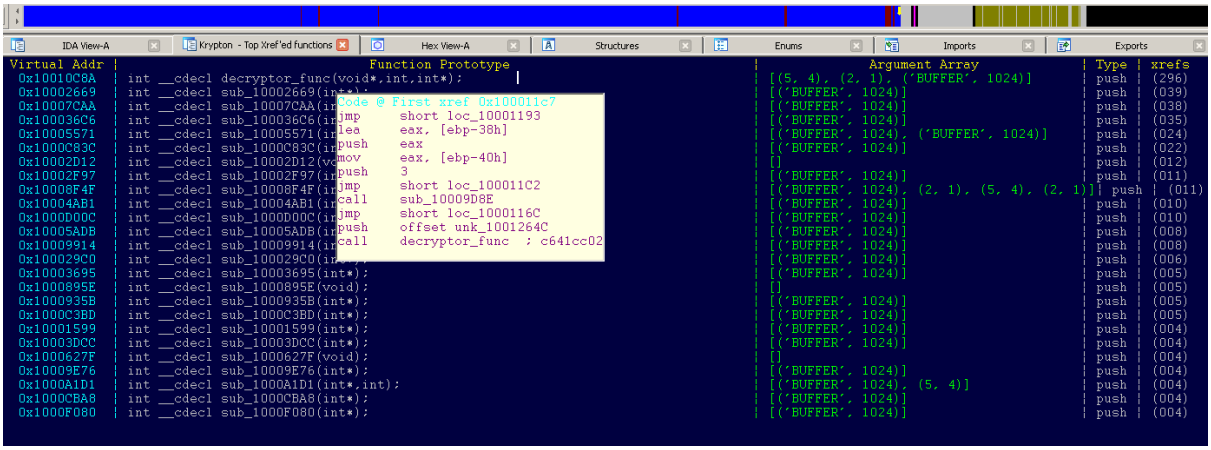
```
push 0x04010010     (68 10 00 01 04)
push 3              (6A 03)
push 0x03b09000     (68 00 90 b0 03)
call 0x10010C8A
```

there are 3 arguments, whose argument array pair would be, [(5,4),(2,1),(5,4)] since the 3rd argument is a memory buffer we have to pass allocated buffer address.

To accomplish this, Plugin allows a few "Keywords" to be specified in Argument array
To pass a allocated buffer, use **"BUFFER"**
- this instructs Plugin to allocate memory and pass the address as argument, and buffer size is the second value within parenthesis. As in ('BUFFER', 0x400)

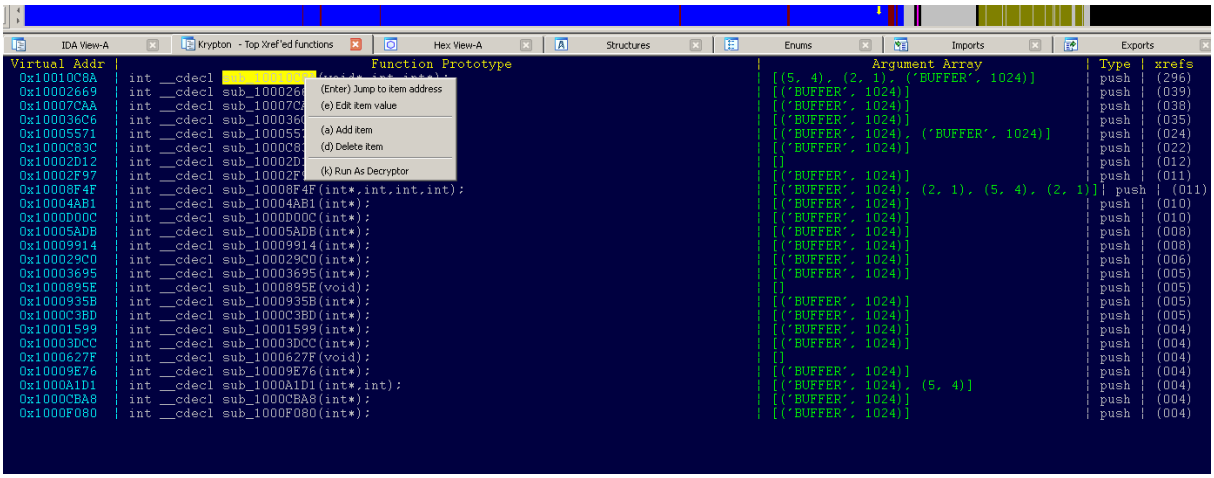
To pass a const value in place of an argument, use **"CONST"**
- this instructs Plugin to pass a constant value as a parameter, const value is the second value within parenthesis. As in ('CONST', 0x11).

In our example, first function sub_10010C8A has 296 references, when analyzed we can confirm that it is a decryptor. Hovering over these functions will bring up hint that shows, 10 disassembly lines from a first cross reference of that function. This helps one to verify and fix the argument array.



Right clicking on the Protolist view will bring a Popup Menu as shown in the below screen shot, bringing options to edit/delete/add function prototypes.

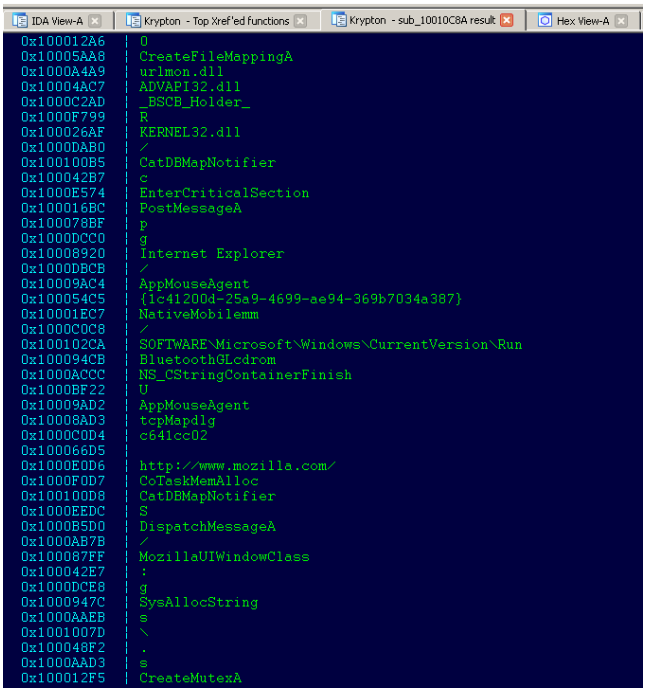
One can add new function to the prototype list, by placing cursor in the interested function from IDA dis-assembly view and issuing **Ctrl+Y** shortcut.



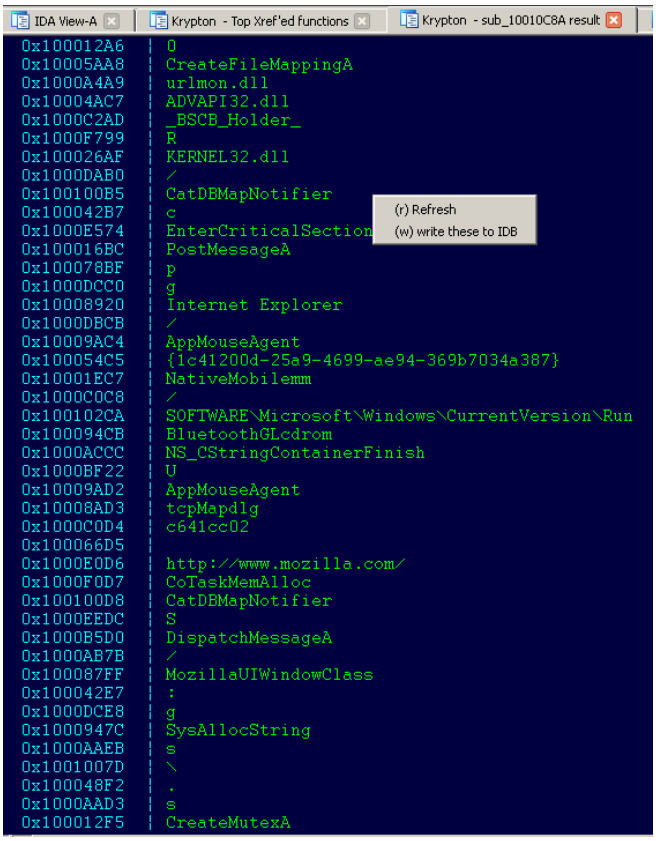
Once we verified/corrected that the function Prototype and Argument Array listed, we can run the function for all its cross references, by issuing **"k"** key in keyboard or by right clicking and selecting menu item **"(k) Run as Decryptor"**

When issued, Plugin automatically sets up necessary breakpoints and runs the selected function in default debugger and returns the result in another view as shown in the below figure.

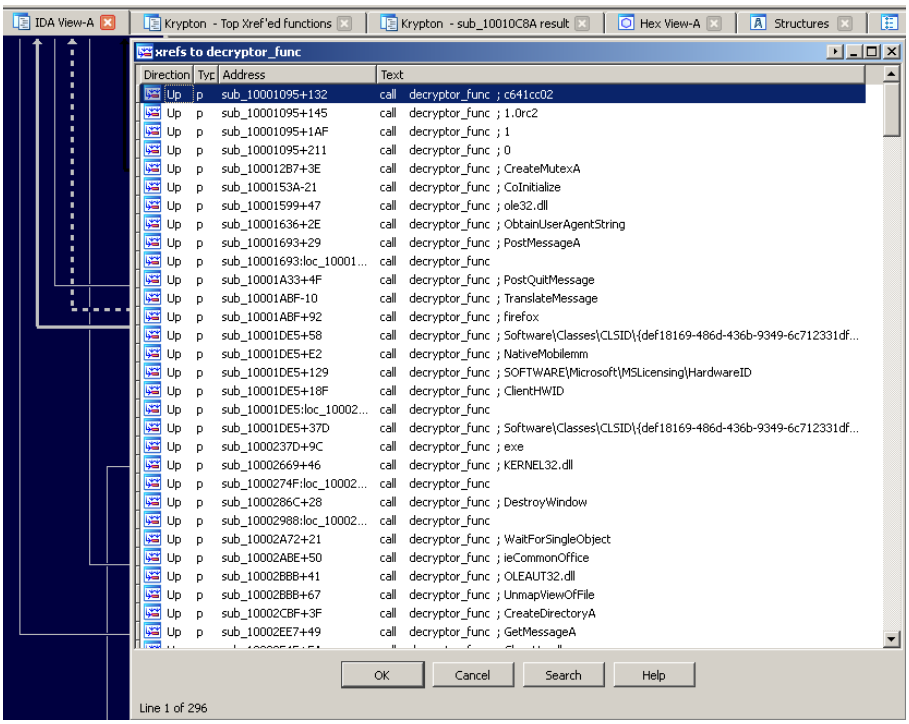
Note: Krypton does not execute the binary like any other debugger, it only runs the selected function by setting up necessary stack with the arguments it found at the respective reference points of the function.



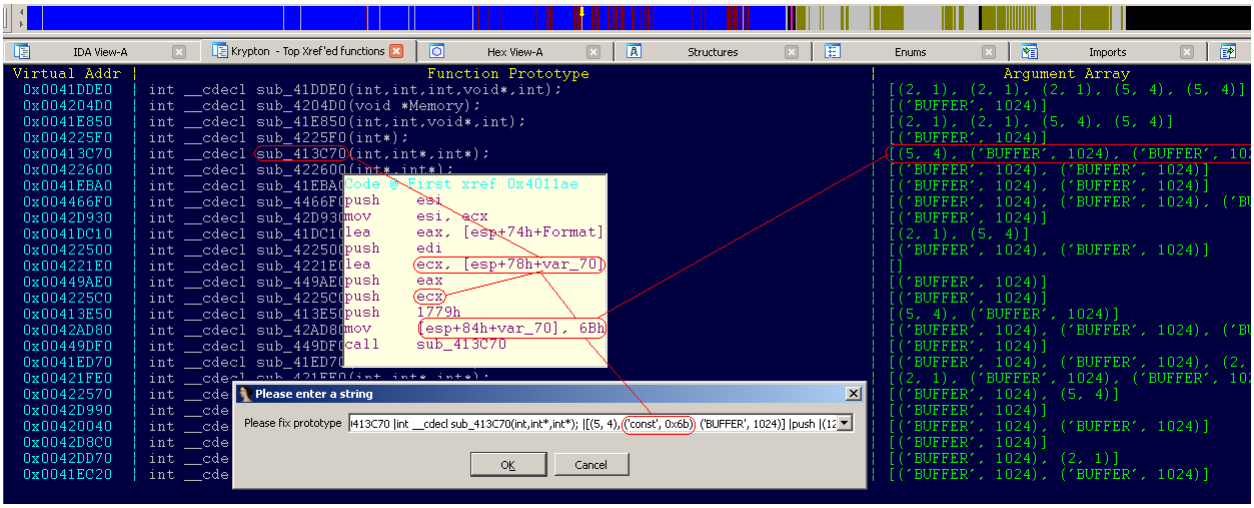
When the decryption results are satisfactory one can have the results written back to IDB as comments at their appropriate call references by issuing **"w"** key in keyboard or by right clicking and selecting menu item **"(w) write these to IDB"** as shown below.



result of writing into IDB can be easily seen in the cross reference of the decrypt function as below.



Let see an example of a decryptor function where prototype editing can be used,



In the above figure, we can see that Plugin guessed the prototype to be sub_413C70(int, int*, int*), and the argument array as [(5,4), (‘BUFFER’, 1024), (‘BUFFER’, 1024)].

However, when looked at the disassembly in the hint, one can see that a constant value **0x6B** is passed through a local variable var_70.

So we can tell krypton to use a const value for second argument by editing the prototype listed to have argument array value [(5,4), (‘const’, 0x6b), (‘BUFFER’, 1024)].

Now when run with the modified prototype, krypton executes correctly, decrypts and shows us result.