IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. I (Jan. 2014), PP 71-77
www.iosrjournals.org

Manual Unpacking Of Upx Packed Executable Using Ollydbg and Importrec

Asha Devi, Gaurav Aggarwal
Independent Security Researcher SecurityExploaded and M.Tech Student of ITM UNIVERSITY, Gurgaon (Haryana)INDIA 122001.

Summary: *A 'Packer' is a compression routine that compress an executable file. Packers are used on executable for two main reasons: to shrink programs or to thwart detection or analysis. When malware has been packed, an analyst typically has access to only the packed file, and cannot examine the original unpacked program or the program that packed the malware. In order to unpack an executable, we must undo the work performed by the packer, which requires that we understand how a packer operates. All packers take an executable file as input and produce an executable file as output. The packed executable is compressed, encrypted, or otherwisetransformed, making it harder to recognize and reverse-engineer. Unpacked executable are loaded by the OS. With packed programs, the* ***unpacking*** *stub is loaded by the OS, and then the unpacking stub loads the original program. The code entry point for the executable points to the unpacking stub rather than the original code. The original program is generally stored in one or more extra sections of the file.*
Key words: *Packing, Unpacking stub,PE header,Sections*

I. Introduction:

UPX unpacking is a process of decompressing the packed executable and reconstructs its import address table.When an Executable or DLL is packed its IAT table destruct and need to construct during unpacking process. Unpacked executable or DLL are loaded by the operating system.But when the program is packed,only the unpacking stub is loaded by the O.S and then unpacking stub load the original program.The code entry point points to the Unpacking stub then OEP.

The unpacking stub can be viewed by the malware analyst and understanding different parts of stub is necessary to unpack the executable.The unpacking stub is often small and its main functionality is to **unpack the Original Executable**.

The unpacking stub perform the following steps:

- Unpack the original executable into memory.
- Resolve all the imports of original executable.
- Transfer execution to the Original Entry point.

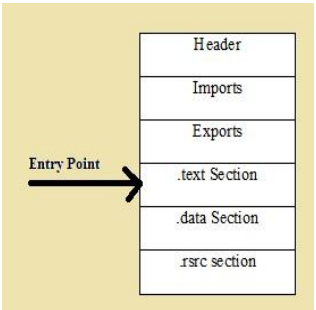


Fig 1: Original Executable prior to Packing

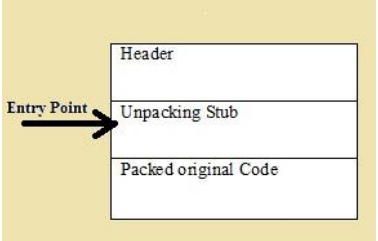


Fig 2: Packed Executable after original code is packed and unpacking stub added.

II. Theoretical Consideration:

When the UPX packed exe will execute followings will happen:

- Execution starts from OEP.
- First it saves the current register status using PUSHAD instruction
- All the packed sections are unpacked into memory
- Resolve the import table of original executable file
- Restore the original register status using POPAD instruction
- Finally jump to the original entry point to begin actual execution.

The instruction that transfers execution to the OEP is commonly referred toas the ***tail jump***.A jump instruction is the simplest and most popular way to transfer execution.Since it's so common, many malicious packers will attempt to changethis function by using a **ret**or **call**instruction. Sometimes the tail jump is obscured with OS functions that transfer control, such as NtContinueor ZwContinue.

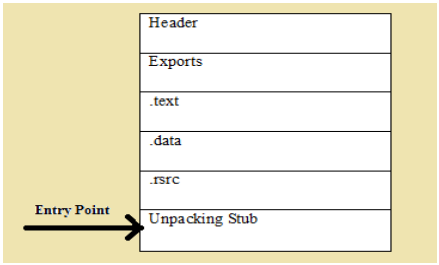


Fig 3: The program after being unpacked and loaded into memory and there is no import.

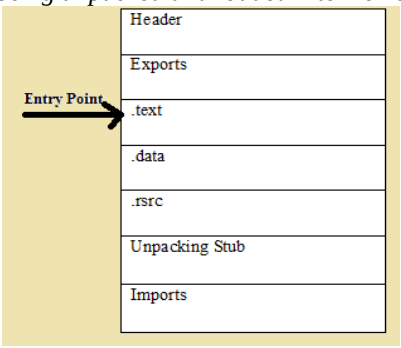


Fig 4: The fully unpacked program.The import table is reconstructedand starting point is backto OEP.

Identifying Packed Programs:

There are few indicators which show that the file is packed or not:

- The program has few imports, and particularly if the only imports are LoadLibraryand GetProcAddress.
- When the program is opened in IDA Pro, only a small amount of code is recognized by the automatic analysis.
- When the program is opened in OllyDbg, there is a warning that the program may be packed. The program shows section names that indicate a particular packer (such as UPX0).
- The program has abnormal section sizes, such as a .text section with a Size of Raw Data of 0 and Virtual Size of nonzero.
- Packer-detection tools such as PEiD,ExeScan can also be used to determine if an executable is packed.

Entropy Calculation

Packed executables can also be detected via a technique known as *entropycalculation*. Entropy is a measure of the disorder in a system or program.Compressed or encrypted data more closely resembles random data, and therefore has high entropy; executables that are not encrypted or compressed have lower entropy.

Unpacking Options:

There are mainly three options for the unpacking of executable:

- Automated static unpacking,
- Automated dynamic unpacking
- Manual dynamic unpacking

I will explain Manual Unpacking:

Manual Unpacking:

Sometimes, packed malware can be unpacked automatically by an existing program, but more often it must be unpacked manually. Manual unpacking can sometimes be done quickly, with minimal effort; other times it can be a long process.

There are two common approaches to manually unpacking a program:

- Discover the packing algorithm and write a program to run it in reverse. By running the algorithm in reverse, the program undoes each of the steps of the packing program. There are automated tools that do this, but this approach is still inefficient, since the program written to unpack the malware will be specific to the individual packing program used. So, even with automation, this process takes a significant amount of time to complete.
- Run the packed program so that the unpacking stub does the work for you, and then dump the process out of memory, and manually fix up the PE header so that the program is complete. ***This is the more efficient approach.***

- ***Rebuilding the Import Table with Import Reconstructor:***

Rebuilding the import table is complicated, and it doesn't always work in OllyDump. The unpacking stub must resolve the imports to allow the application to run, but it does not need to rebuild the original import table. When OllyDbg fails, it's useful to try to use Import Reconstructor (ImpRec) to perform these steps.

III. Experimental Consideration:

In order to unpack the packed program need the following steps:

1. **Load the UPX packed EXE into PEid or ExeScan to confirm that it is UPX Packed**.

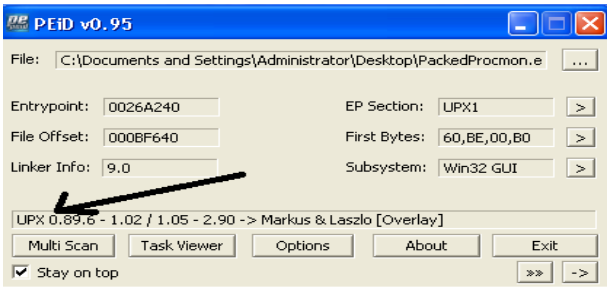


Fig 5: PEid result showing UPX Packed File.

ExeScanConfirmed that it is packed.

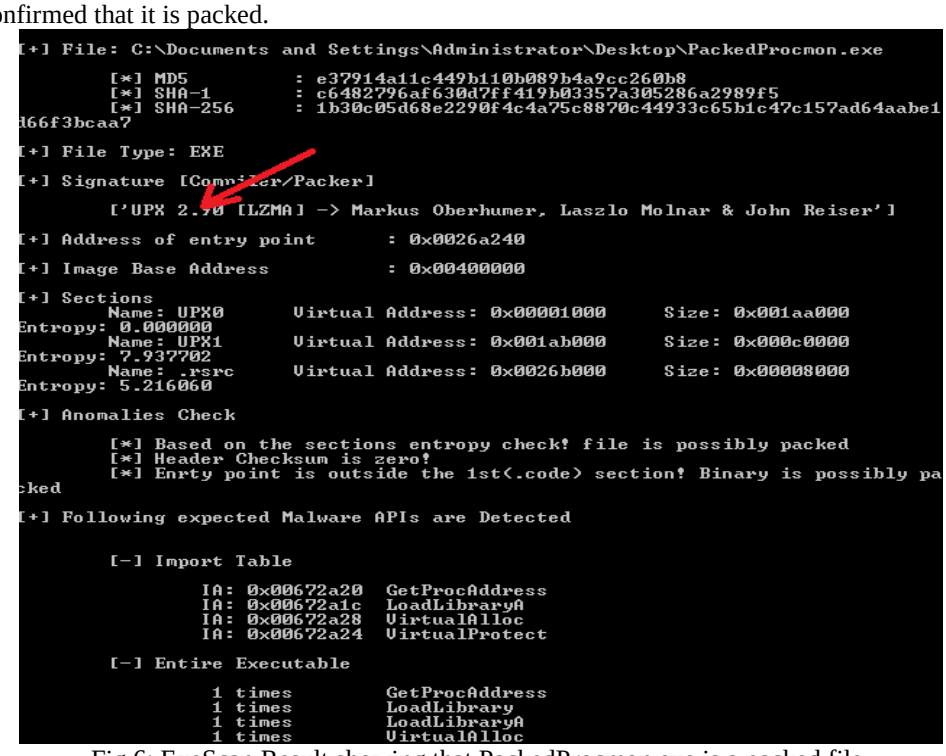
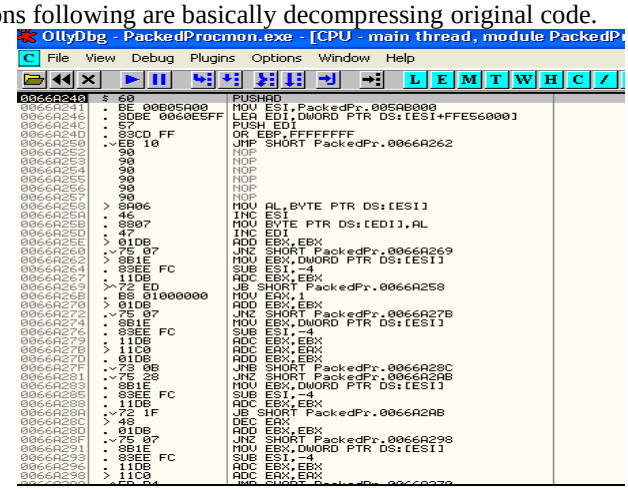


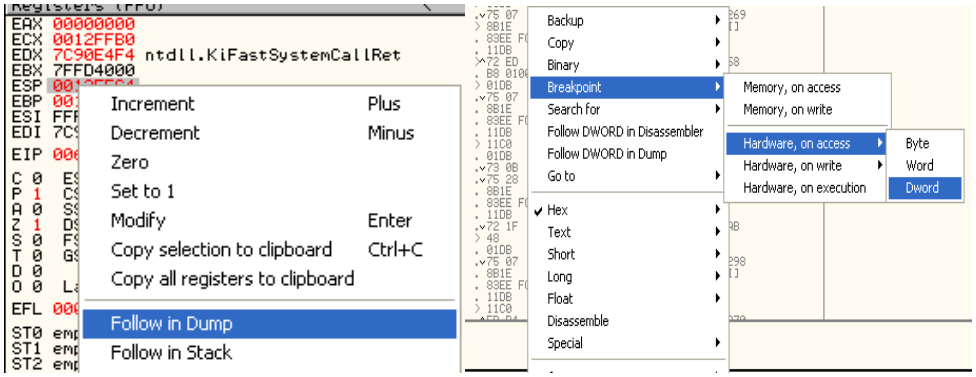
Fig 6: ExeScan Result showing that PackedProcmon.exe is a packed file.

2. **Load the UPX Packed EXE into Ollydbg**.

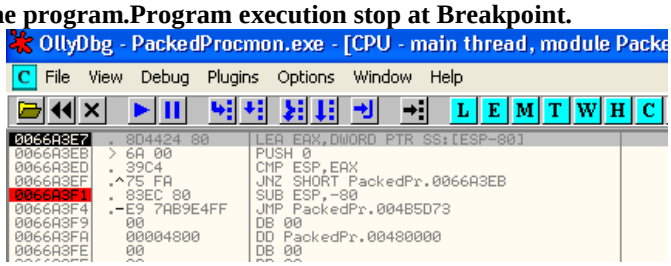
The first instruction is usually PUSHAD, before starting decompression routine,it saves all the register content.Now the instructions following are basically decompressing original code.



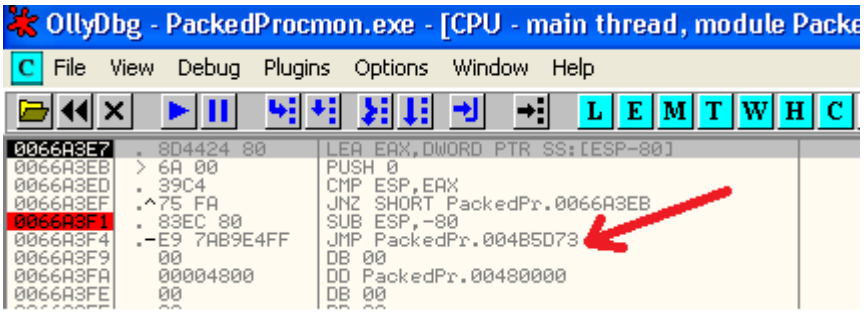
3 : Now we reach PUSHAD,put a hardware breakpoint so as to stop at POPAD instruction.This will help us to stop execution when POPAD instruction is executed later on.



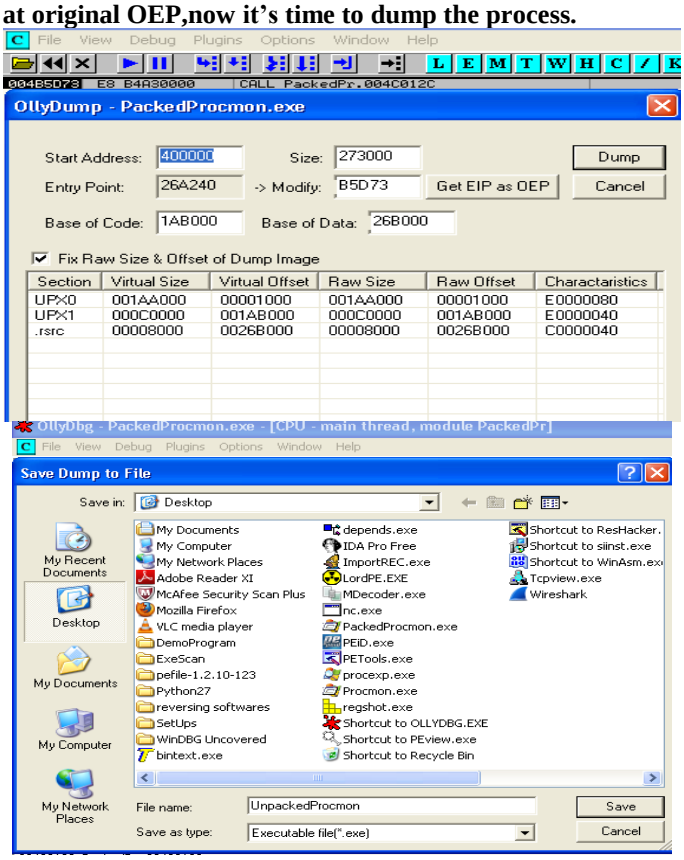
1. Now execute the program.Program execution stop at Breakpoint.



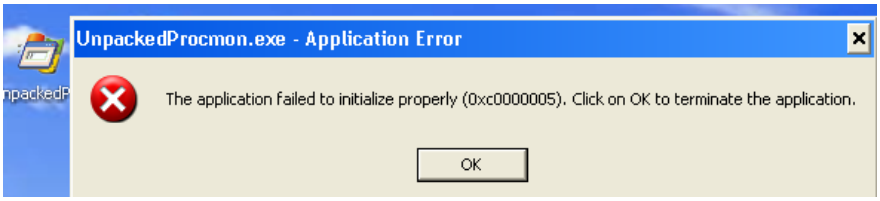
2. Now Search for Jmp instruction which will take us to the actual OEP in the original program.At this point we know the decompression stub is executed and ready to jump at original code.



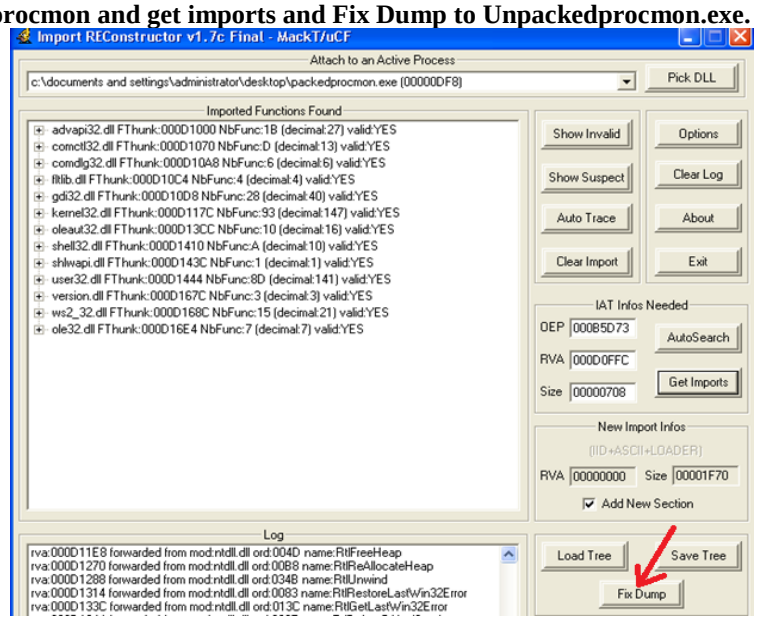
3. Now we reach at original OEP,now it's time to dump the process.



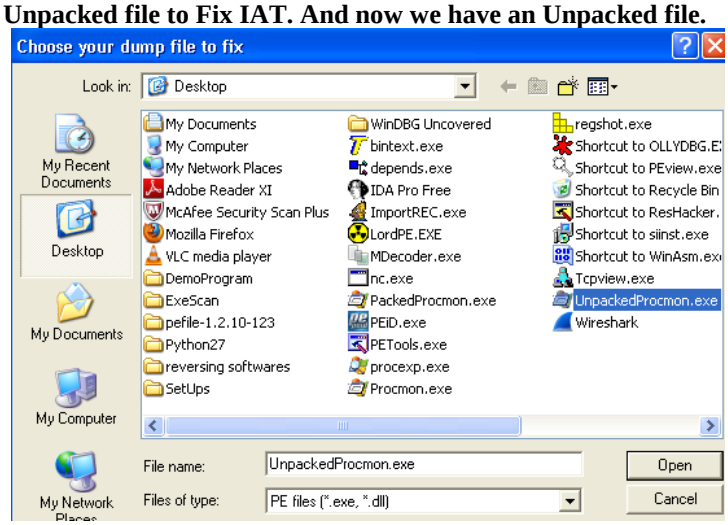
4. Now if we try to open the UnpackedProcmon.exe it gives error because IAT table is not constructed.



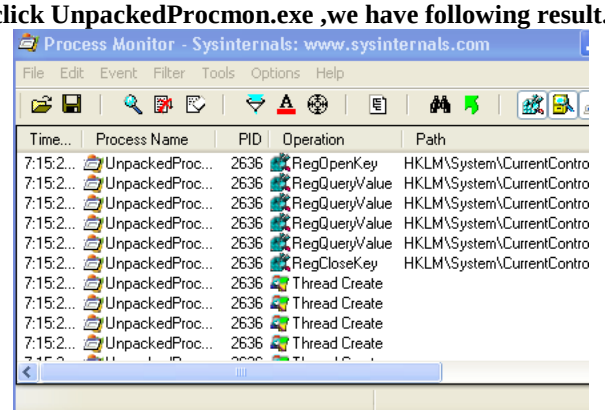
5. Now next step is to Reconstruct the IAT using importREC.
Attach the Packedprocmon and get imports and Fix Dump to Unpackedprocmon.exe.



6. Now choose Unpacked file to Fix IAT. And now we have an Unpacked file.



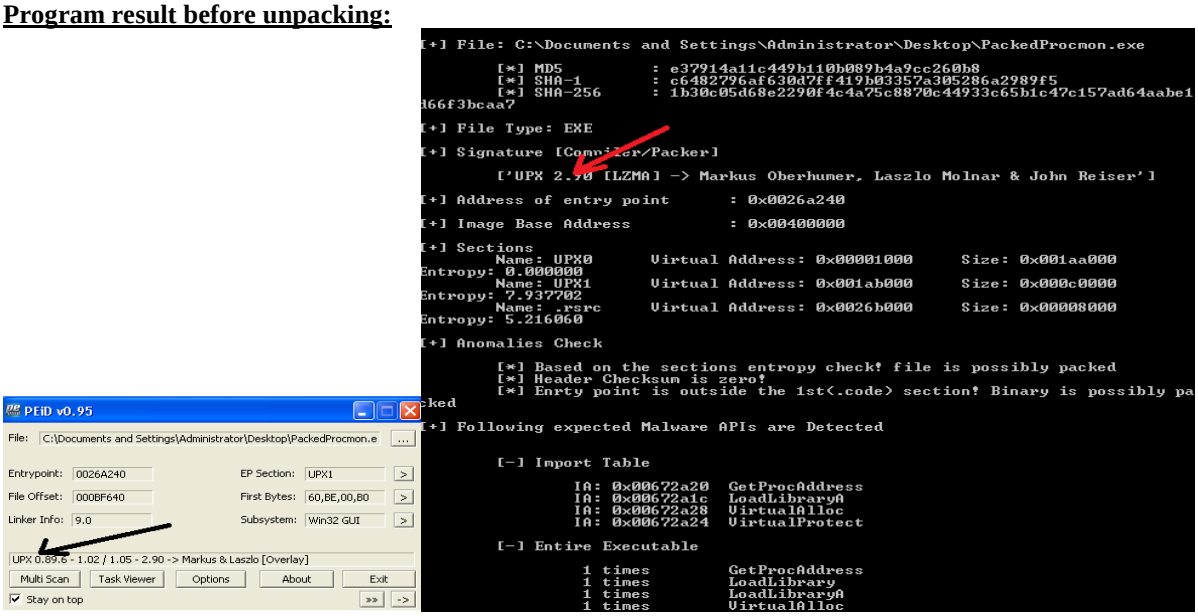
7. **When we double click UnpackedProcmon.exe ,we have following result. IAT has been fixed**.



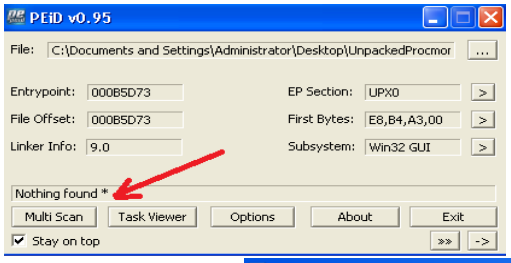
IV. Conclusion:

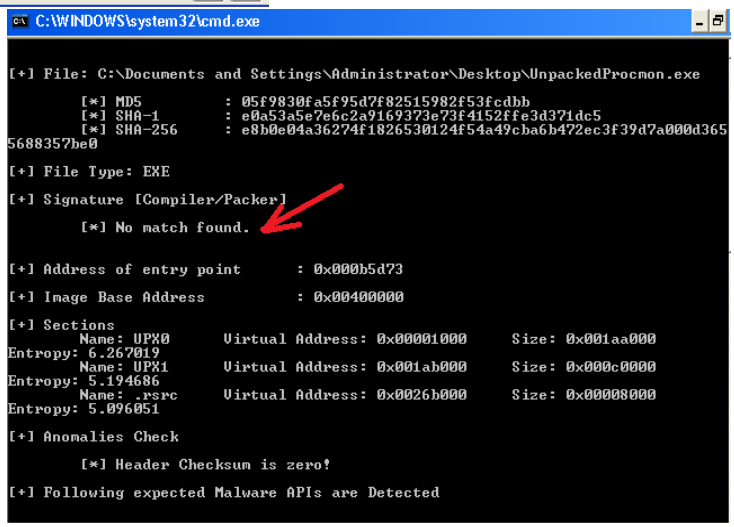
In an experiment observation we found that we can manually unpack UPX packed file and reconstruct IAT table using Ollydbg and importRec .

Program result before unpacking:



Program Result after Unpacking:





References:

[1]. Practical Malware Analysis,The hand on guide to dissecting malicious software by Michael Sikorski and Andrew Honig forward by Richard Bejtlich,No starch Press.
[2] Code Breakers Magazine Security and Anti Security.Attack and Defence by Goppit
[3] Reversing :Secret of reverse engineering by EldadEilam Foreword by Elliot Chikofsky
[4] www.securityExploaded.com
[5] Security Power Tools **By:** Bryan Burns; Dave Killion. **Publisher:** O'Reilly Media, Inc

ASHA DEVI Received B.Tech Degree from PDM college of Engineering,Bahadurgarh,INDAI in 2008 and pursuing M.Tech from ITM University Gurgaon,INDIA.I worked as Assistant profeesor in KIIT Gurgaon,INDIA for 4 year and currently working as Independent Security researcher in SecurityExploaded..

Acknowledgement:
The authors would like to express their cordial thanks to
Mr .Amit Malik(Applied Research, Security - Computer Science)
Organization: FireEye) for his valuable advice.
LinkedIn : http://www.linkedin.com/in/doublezer0
Website : http://www.securityxploded.com
http://www.securityphresh.com