

Hook Analyser Project

Project Home – www.hookanalyser.com

Author – Beenu Arora (www.beenuarora.com)

Email – beenudel1986@gmail.com

Description: Hook Analyser is a freeware project, started in 2011, to analyse an application during the run-time. The project can be potentially useful in analysing malwares (static and run time), and for performing application crash analysis.

The following sections break down the features (and functionality) of the Hook Analyser, and attempts to answer 'How-to' and 'so-what' queries.

Application UI – Significant updates have been performed on the latest release (v 2.2) to make it more verbose.

```
-----
: beenude11986 [ @ ] gmail [ dot ] com
: 10/2012      Hook Analyser 2.2
: Do Visit    www.BeenuArora.com & www.HookAnalyser.com
: Usage - Interactive : HookAnalyser2.2.exe
: For bugs and improvements - Please send an email
-----
[*] Welcome to HookAnalyser Interactive Mode

[1] Spawn and Hook to Application
[2] Hook to a specific running process
[3] Perform quick Static Malware Analysis
[4] Application crash analysis

[-] Please enter your choice [1/2/3/4] : _
```

Figure 1

1. *Spawn and Hook to Application*

This feature allows analyst to spawn an application, and hook into it. Module flow is as following–

- a. *PE validation*
- b. *Quick static malware analysis.*
- c. *Other options (such as pattern search or dump all)*
- d. *Type of hooking (Automatic, Smart or manual)*
- e. *Spawn and hook*

Currently, there are three types of hooking being supported –

- *Automatic – The tool will parse the application import tables, and based upon that will hook into specified APIs.*
- *Manual – On this, the tool will ask end-user for each API, if it needs to be hooked.*
- *Smart – This is essentially a subset of automatic hooking however, excludes uninteresting APIs.*

Once an application is specified, the tool will perform a quick static analysis to identify any anomalies or malware traces. Refer to the **section 4** for detailed information.

```

[*] Exe name with path. Ex: "c:\test.exe" : infected.exe
[-] Doing initial static analysis on the file

[*] Analysing if valid PE file
[*] Valid PE File
[*] File Size : 204 KB
[+] Verifying CRC from file
[-] CRC Seems fine
[+] Verifying timestamp from file
[-] Timestamps seems fine
[+] Image Base : 0x10000000L
[+] Address Of Entry Point: 0x1BB0L
[+] Number of RVA and Sizes: 16
[+] Subsystem: IMAGE_SUBSYSTEM_WINDOWS_GUI
[+] Searching for TLS entries..
[-] No TLS entries identified
[+] Found Entry Point at section: .text
[-] Entry point in known section. Seems fine
[+] Identifying Suspicious section
[!] SUSPICIOUS
Section Name: IMAGE_SECTION_HEADER      Entropy 7.20453591583

[IMAGE_SECTION_HEADER]
0x278      0x0      Name:                .data2
0x280      0x8      Misc:                0x218B0
0x280      0x8      Misc_PhysicalAddress: 0x218B0
0x280      0x8      Misc_VirtualSize:    0x218B0
0x284      0xC      VirtualAddress:       0x7000
0x288      0x10     SizeOfRawData:        0x21A00
0x28C      0x14     PointerToRawData:     0x3200
0x290      0x18     PointerToRelocations: 0x0
0x294      0x1C     PointerToLinenumbers: 0x0
0x298      0x20     NumberOfRelocations:  0x0
0x29A      0x22     NumberOfLinenumbers:  0x0
0x29C      0x24     Characteristics:      0xC0000040

[!] Executable is Debug aware
[!] Executable is exception aware
[!] Executable can hook to keyboard
[!] Executable could change DEP setting. This is suspicious
[!] Executable could spawn a new process
[!] Executable is potentially anti-debug aware
[-] Extracting file information from executable
    LegalCopyright ---
    InternalName --- OOBEBALN.EXE
    FileVersion --- 5.1.2600.5512 (xpsp.080413-2111)
    CompanyName ---
    ProductName --- Microsoft Windows
    ProductVersion --- 5.1.2600.5512
    FileDescription --- Windows OOBE
    OriginalFilename --- OOBEBALN.EXE
[!] Found Anti VM Trick in the executable
[+] MD5 checksum of the executable is :164b52398a16852a0413f769557b0445
[-] Extracting strings from the executable...
[=====Start=====]
\Rrp.exe
Attgapi32
%1!d!
.%0
...%0
.%0
.%0
...%0
%1%0
Windows%0
...%0
ISP %1%0
.%0

```

Figure 2

```

[=====Start=====]
\Rrp.exe
Attgapi32
%1!d!
.%0
...%0
.%0
.%0
...%0
%1%0
Windows%0
...%0
ISP %1%0
.%0
US_VERSION_INFO
StringFileInfo
041904B0
CompanyName
FileDescription
Windows 00BE
FileVersion
5.1.2600.5512 (xsp.080413-2111)
InternalName
OOBEBALN.EXE
LegalCopyright
OriginalFilename
OOBEBALN.EXE
ProductName
Microsoft
Windows
ProductVersion
5.1.2600.5512
UarFileInfo
Translation
<<<Obsolete>>
&fffffffd2942xAJCgwf1Hf3a618xaIrsL21m9ym2z p56a5jh13 Kno7hJvjLyCohC
oBqr4di3E8FL G3i11Cljdz1FIefCEf722IhH6babidLuweAEJAkh2b1 tqb CoGwtoyu5ek8jnpievl
Hzlnq5xB8d yj9DzKAuMB4JAmw1fBHKGu35x Ij2gLo4fijg 9hdAp65ctHw1Evuzdt zKM31Ggs6F35
ocyyg5ElgbughJprd3MGzxoIKfqbM74oJkuwv26trdm5q5wiAqk4AEmaAb383vJ9mLklqCngI2ev1qd
MmB41z1bKwu3LFC4uHhzFmvyx1zwt15h1ctyMJac58tsAHH 1er2FrFm6pq5h7qHyrvyD6Ctkb51CyA
bkH7GCEzE31jGF6erIrfB AxFIb7o6Jzx78j4F62Ap1xfAuyM68785C4dbiKcfaLej7 Fbgwoy2x7jdx
iJcmb2Hgw229xgem2GbFMwiHCw4wdBlua2iME 7tmfofbFdlvobf7tiGofc1kgdeveDEpyphuLBoxzre
bGJKEpdH fqaKim8h5 7gtY29eskd4JMuzb89CoC 9FcxGbC5aBpfdwj9d4Cr7mxxJ64B2GEAEen9ufs
g8BxtsKb iJCjwsAGriFD2o9Md1iFfDI8C4AtnnuHoEL7GDtaBMwHF7yw? wkwJghfj6EJmidbH1kL6w
fC1vr6z9Lyl6oEofcx486kxL26hxLypj6m6EpaJch4mJAraJAvKjGdJaDAhwmyEan6GkuBodEz6koKp
EIEIKjl1xoDjrpqkgm1C95M9hr3e1lz59Kv5 u59rky19GkeHE1MBBsZkpyE2u2CpK2g8vA1vuzrp F1
uGjMgGinJEMJlsdch52uir4squ48nH1M4JCjGeiHCBDuB 2dzp3pItABtrzyKzcofkkIma69a49j8hfI

```

Figure 3

Once static analysis is performed, the tool allows analyst to perform one of the following –

- *List APIs* – The option is used to list API calls only, and this doesn't display buffer data (on stack or heap).
- *Pattern search* – The option allow searching for patterns or 'strings' in memory after application is spawned and hooked.
- *Dump all* – The option will list API calls and output data from respective buffer (stack or heap).

Once the above option is selected, analyst can chose type of hooking and proceed (as described above)

```
=====End=====]
[*] Select options : l (default, list calls) , p(pattern search) , d (dump all) -
<l/p/d)? Choice :l
[*] Hooking mode ? Automatic (a) or Manual (m) or Smart (s)? Choice (a/m/s) :s
[*] Seems everything is running file
[*] Launching the executable. Press ENTER to continue
```

Figure 4

Sample malware analysis demonstration – <http://www.youtube.com/watch?v=sdnRP9oweT4>

Note – This video was created using Hook Analyser 1.4 (the old version).

2. Hook to a specific running process

The option allows analyst to hook to a running (active) process. The program flow is –

- a. List all running process
- b. Identify the running process executable path.
- c. Perform quick static malware analysis on executable (fetched from process executable path)
- d. Other options (such as pattern search or dump all)
- e. Type of hooking (Automatic, Smart or manual)
- f. Hook and continue the process

```
[2] Hook to a specific running process
[3] Perform quick Static Malware Analysis
[4] Application crash analysis

[-] Please enter your choice [1/2/3/4] :2
[-] Listing all active processes
0 - System Idle Process
4 - System
700 - smss.exe
772 - csrss.exe
804 - winlogon.exe
848 - services.exe
860 - lsass.exe
1020 - vmacthlp.exe
1036 - svchost.exe
1120 - svchost.exe
1368 - svchost.exe
1420 - svchost.exe
1472 - svchost.exe
1864 - spoolsv.exe
312 - MDM.EXE
416 - vmtoolsd.exe
784 - UMUpgradeHelper.exe
1328 - IPAutoConnSvc.exe
1744 - alg.exe
1232 - wscntfy.exe
1448 - explorer.exe
1740 - VMwareTray.exe
1756 - VMwareUser.exe
1772 - ctfmon.exe
360 - IPAutoConnect.exe
1720 - cmd.exe
160 - python.exe
3040 - cmd.exe
4032 - chrome.exe
1340 - chrome.exe
1272 - chrome.exe
132 - notepad++.exe
3484 - chrome.exe
2988 - rundll32.exe
512 - cmd.exe
2716 - HookAnalyser2.2.exe
2552 - HookAnalyser2.2.exe
628 - wmiprvse.exe
[-] Enter the process id :
```

Figure 5

```
[~] Enter the process id : 4032
[!] Hooking to spawned process
[*] Identifying process path for process 4032
[*] Process path is :c:\program files\google\chrome\application\chrome.exe
[*] Analysing if valid PE file
[*] Valid PE File
[*] File Size : 1242 KB
[+] Verifying CRC from file
[-] CRC Seems fine
[+] Verifying timestamp from file
[-] Timestamps seems fine
[+] Image Base : 0x400000L
[+] Address Of Entry Point: 0x7EA06L
[+] Number of RVA and Sizes: 16
```

Figure 6

Refer to the **section 1** for the rest of the module flow description.

3. Perform quick static malware analysis

This module is one of the most interesting and useful module of Hook Analyser, which performs scanning on PE or Widows executables to identify potential malware traces.

This module has inherited lot of feature from malware analyser project (www.malwareanalyser.com).

Currently this module perform the following –

- a. PE file validation
- b. CRC and timestamps validation
- c. PE properties such as Image Base, Entry point, sections, subsystem
- d. TLS entry detection.
- e. Entry point verification (if falls in suspicious section)
- f. Suspicious entry point detection
- g. Signature trace (extended from malware analyser project), such as Anti VM aware, debug aware, keyboard hook aware etc. This particular function searches for more than 20 unique malware behaviours (using 100's of signature).
- h. Online search of MD5 (of executable) on Threat Expert.
- i. String dump (ASCII)
- j. Executable file information
- k. ...and more.

Refer to the **figure 2** for sample screenshot.

4. Application crash analysis

This module enables exploit researcher and/or application developer to analyse memory content when an application crashes.

This module essentially displays data in different memory register (such as EIP).

Application crash analysis video demonstration – <http://www.youtube.com/watch?v=msYo7pPsu6A>