

## Check Out Our New HANDS-ON Computer Forensics Training



Click Here

CONTENT ARCHIVES

HACKING CONTENT ARCHIVES

JOB BOARD

2015 2014 PHISHING

2013 2012 CERTIFICATIONS

2011 2010 FORENSICS

SECURE

CODING

PENETRATION

TESTING

GENERAL

SECURITY

CLOUD

COMPUTING

INTERVIEWS

VIRTUALIZATION

SECURITY

WIRELESS

SECURITY

SCADA

REVERSE

ENGINEERING

DATA

RECOVERY

EXPLOIT

DEVELOPMENT

# Malware Researcher's Handbook

POSTED IN MALWARE ANALYSIS ON DECEMBER 23, 2015

SHARE

## Ethical Hacking Boot Camp

OUR MOST POPULAR COURSE!

CLICK HERE!

### SKILLSET

What's this?

Practice for certification success with the [Skillset library of over 100,000 practice test questions](#). We analyze your responses and can determine when you

MANAGEMENT,

COMPLIANCE, &amp;

are ready to sit for the test.

## Demystifying PE File (Part 2: Continue)

As per our previous article we will continue on this article here by the rest of section of PE file. Those who don't know the previous section please have a look on here(<http://resources.infosecinstitute.com/2-malware-researchers-handbook-demystifying-pe-file/>).

Because this will be more advance section as well some form of automation stuff, where I will implement windows API as well as Python programming.

As we said in previous article we will discuss how TLS callback function is very helpful for attacker and also we will show demonstration by an application. But before that we should finish the rest of part of PE section

## The Export Section:

Exports are functions and values in one module that have been declared to be shared with other modules. This is done through the use of the "Export Directory", which is used to translate between the name of an export (or "Ordinal", will discuss more in later point), and a location in memory where the code or data can be found. The start of the export directory is identified by the IMAGE\_DIRECTORY\_ENTRY\_EXPORT entry of the resource directory like below:

```
struct IMAGE_EXPORT_DIRECTORY {  
  
    long Characteristics;  
  
    long TimeDateStamp;  
  
    short MajorVersion;  
  
    short MinorVersion;  
  
    long Name;
```

```
    long Base;  
  
    long NumberOfFunctions;  
  
    long NumberOfNames;  
  
    long *AddressOfFunctions;  
  
    long *AddressOfNames;  
  
    long *AddressOfNameOrdinals;  
  
}
```

This section is particularly reference to DLL file and its structure.

In Microsoft Windows A DLL are the modules that contains functions and data. A DLL is loaded at runtime by its calling module that may be exe or a DLL. When a DLL is loaded, it is mapped into address process of calling function.

A DLL can have two sections: **Exported and Internal.**

**The Exported functions can be called by other modules. Internal functions can be called within the module/DLL where they defined.**

The actual exports themselves are described through AddressOfFunctions, which is an RVA to an array of RVA's, each pointing to a different function or value to be exported. The size of this array is in the value NumberOfFunctions. Each of these functions has an ordinal. The "Base" value is used as the ordinal of the first export, and the next RVA in the array is Base+1, and so forth.

Each entry in the AddressOfFunctions array is identified by a name, found through the RVA AddressOfNames. The data where AddressOfNames points to is an array of RVA's, of the size NumberOfNames. Each RVA points to a zero terminated ASCII string, each being the name of an export. There is also a second array, pointed to by the RVA in AddressOfNameOrdinals. This is also of size NumberOfNames, but each value is a 16 bit word, each value being an ordinal. These two arrays are parallel and are used to get an export value from AddressOfFunctions. To find an export by name, search the AddressOfNames array for the correct string and then take the corresponding ordinal from the AddressOfNameOrdinals array. This ordinal is then used to get an index to a value in AddressOfFunctions.

If we analyze the members of 11 sections of Image\_Import\_Directory, we will only

discuss the important sections as below:

**nName** – The internal name of the module. This field is necessary because the name of the file can be changed by the user. If that happens, the PE loader will use this internal name.

**nBase** – Starting ordinal number (needed to get the indexes into the address-of-function array – see below).

**NumberOfFunctions** – Total number of functions (also referred to as symbols) that are exported by this module.

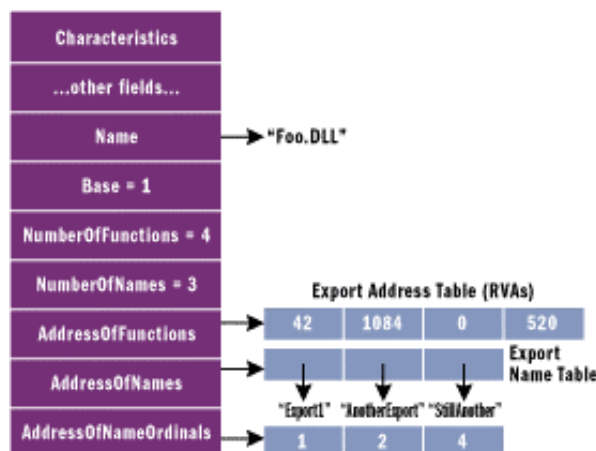
**NumberOfNames** – Number of symbols that are exported by name. This value is not the number of all functions/symbols in the module. For that number, you need to check NumberOfFunctions. It can be 0. In that case, the module may export by ordinal only. If there is no function/symbol to be exported in the first case, the RVA of the export table in the data directory will be 0.

**AddressOfFunctions** – An RVA that points to an array of pointers to (RVAs of) the functions in the module – the Export Address Table (EAT). To put it another way, the RVAs to all functions in the module are kept in an array and this field points to the head of that array.

**AddressOfNames** – An RVA that points to an array of RVAs of the names of functions in the module – the Export Name Table (ENT).

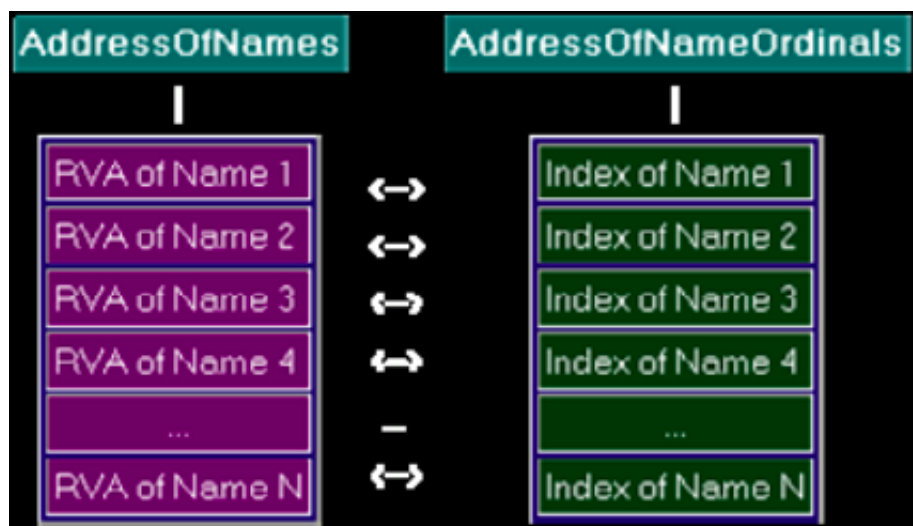
**AddressOfNameOrdinals** – An RVA that points to a 16-bit array that contains the ordinals of the named functions – the Export Ordinal Table (EOT).

We will discuss more in graphical way below:



So **Image\_Export\_Directory** points to three arrays and a table of ASCII strings. The important is Export Address Table, which is an array of function

pointer, that contains the address of exported function. The other 2 arrays (EAT & EOT) run parallel in ascending order based on the name of the function so that a binary search for a function's name can be performed and will result in its ordinal being found in the other array. The ordinal is simply an index into the EAT for that function.



So we can say if function is exported by name we need to walk both **AddressOfNames** and **AddressOfNameOrdinals** arrays to obtain the index into the **AddressOfFunctions** array.

If we already have the ordinal of a function, we can find its address by going directly to the EAT. Although obtaining the address of a function from an ordinal is much easier and faster than using the name of the function, the disadvantage is the difficulty in the maintaining the module. We can see that when we're using the ordinals, obtaining the address of the function is much faster because we only have to calculate one subtract operation

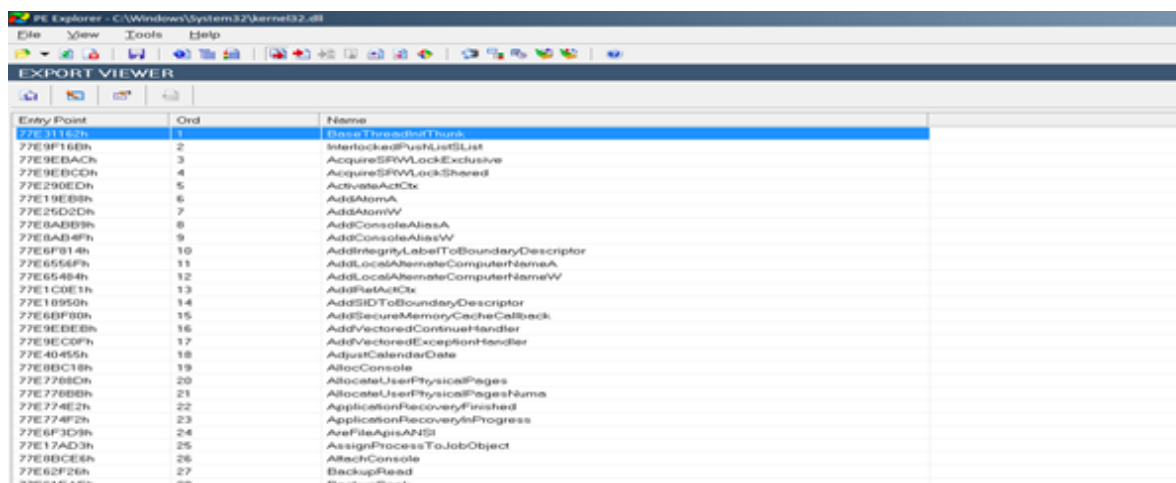
**Export by Ordinal Only:** Number of functions must be equal to the number of names. Sometimes number of name is less than number of functions. So the function that don't have a name are exported by ordinal only.

**Export Forwarding:** Sometimes functions that appear to be exported by a particular DLL that actually reside in a completely different DLL. This is so called Export Forwarding.

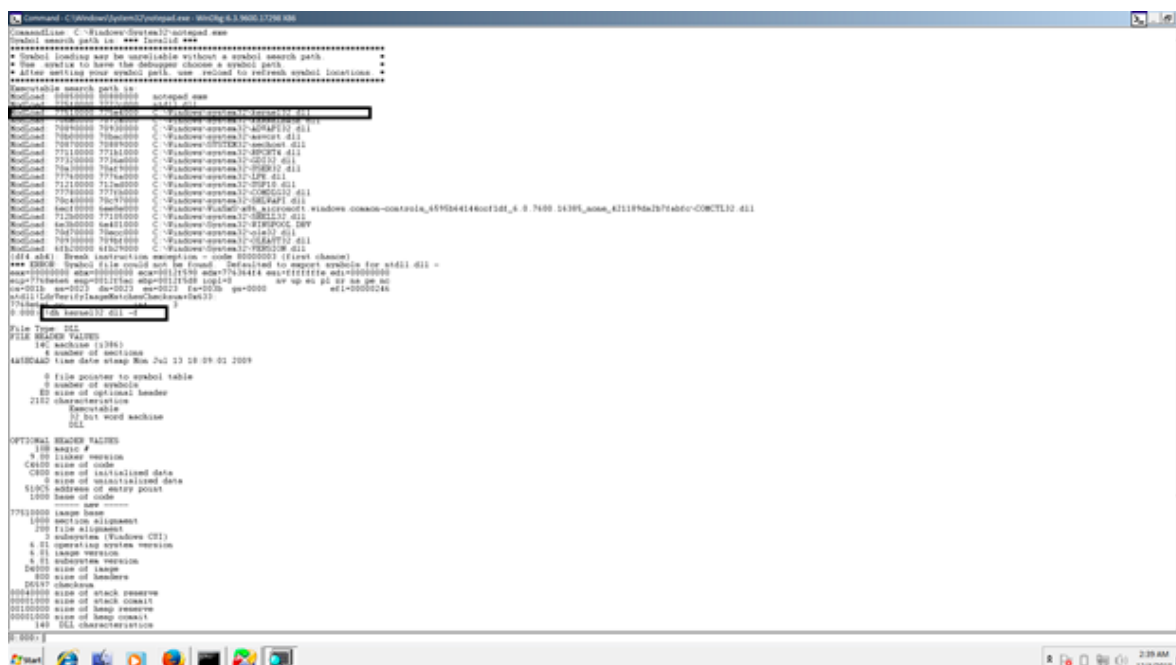
For example, in WinNT, Win2k and XP, the kernel32.dll function `HeapAlloc` is forwarded to the `RtlAllocHeap` function exported by `ntdll.dll`. `NTDLL.DLL` also contains the native API set which is the direct interface with the windows kernel.

**Example/Demonstration:**

Let's print the export header for Kernel32.dll and its subsection. First I will use PE Explorer and after that I will dump all EAT and IAT from notepad.exe by Windbg. After that I will hook on kernel32.dll



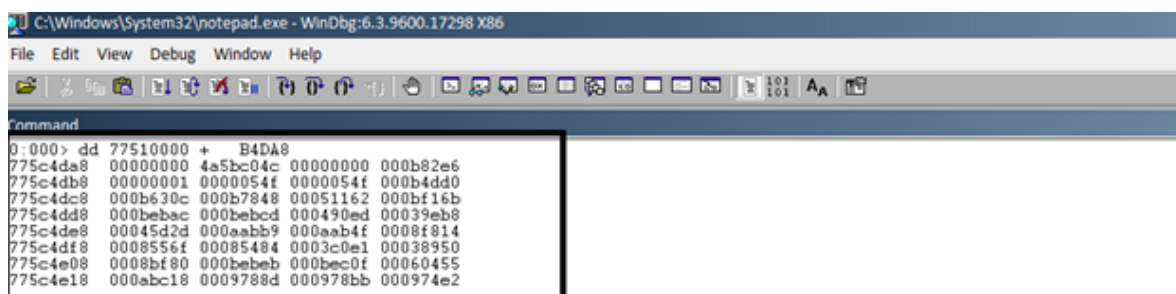
Entry Point	Ord	Name
77E1E1C0h	1	GetProcAddress
77E1E1B0h	2	InterlockedPushList
77E1E1B8h	3	AcquireSRWLockExclusive
77E1E1C8h	4	AcquireSRWLockShared
77E1E1D0h	5	ActivateActCtx
77E1E1E0h	6	AddAtomA
77E1E1F0h	7	AddAtomW
77E1E200h	8	AddConsoleAliasA
77E1E210h	9	AddConsoleAliasW
77E1E220h	10	AddIntegrityLabelToBoundaryDescriptor
77E1E230h	11	AddLocalAlternateComputerNameA
77E1E240h	12	AddLocalAlternateComputerNameW
77E1E250h	13	AdjustActCtx
77E1E260h	14	AddSIDToBoundaryDescriptor
77E1E270h	15	AddSecureMemoryCacheCallback
77E1E280h	16	AddVectoredContinueHandler
77E1E290h	17	AddVectoredExceptionHandler
77E1E2A0h	18	AdjustCalendarDate
77E1E2B0h	19	AllocConsole
77E1E2C0h	20	AllocateUserPhysicalPages
77E1E2D0h	21	AllocateUserPhysicalPagesNuma
77E1E2E0h	22	ApplicationRecoveryFinished
77E1E2F0h	23	ApplicationRecoveryInProgress
77E1E300h	24	AreFileApisANSI
77E1E310h	25	AssignProcessToJobObject
77E1E320h	26	AttachConsole
77E1E330h	27	BackupRead
77E1E340h	28	BackupSeek



```

C:\Windows\System32>dd 77510000 + B4DA8
0:000> dd 77510000 + B4DA8
775c4da8 00000000 4a5bc04c 00000000 000b82e6
775c4db8 00000001 0000054f 0000054f 000b4dd0
775c4db8 000b630c 000b7848 00051162 000bf16b
775c4dd8 000bebac 000bebcd 000490ed 00039eb8
775c4de8 00045d2d 000aabb9 000aabb4 0008f814
775c4df8 0008556f 00085484 0003c0e1 00038950
775c4e08 0008bf80 000bebeb 000bec0f 00060455
775c4e18 000abc18 0009788d 000978bb 000974e2
  
```

So the offset (A915) is Export directory of imagebase(77510000).so the actual address is 77510000+A915.we will see this address by **dd 77510000+B4DA8**



```

C:\Windows\System32>dd 77510000 + B4DA8
0:000> dd 77510000 + B4DA8
775c4da8 00000000 4a5bc04c 00000000 000b82e6
775c4db8 00000001 0000054f 0000054f 000b4dd0
775c4db8 000b630c 000b7848 00051162 000bf16b
775c4dd8 000bebac 000bebcd 000490ed 00039eb8
775c4de8 00045d2d 000aabb9 000aabb4 0008f814
775c4df8 0008556f 00085484 0003c0e1 00038950
775c4e08 0008bf80 000bebeb 000bec0f 00060455
775c4e18 000abc18 0009788d 000978bb 000974e2
  
```

Focus on the first 3 rows. The first column has the memory address. We need to focus on the values in the next 4 columns for the first 3 rows. Parsing these values and comparing them with the IMAGE\_EXPORT\_DIRECTORY structure

definition we get:

```
Characteristics = 00000000
```

```
TimeStamp = 4a5bc04c
```

```
MajorVersion = 0000
```

```
MinorVersion = 000b82e6
```

```
lpName = 00000001
```

```
Base = 0000054f
```

```
NumberOfFunctions = 0000054f
```

```
NumberOfNames = 000b4dd0
```

```
lpAddressOfFunctions = 000b630c
```

```
lpAddressOfNames = 000b7848
```

```
lpAddressOfNameOrdinals = 00051162
```

**Getting RVA:** Now we will get to know the RVA of a different function address.

The RVA of the pointer to AddressOfNames array is: 000b7848

to dump the contents of this array, let's add it to the base address and display:

dd imagebase +AddressOfName

```
0:000> dd 76810000 + 000b7848
768c7848 00030002 00050004 00070006 00090008
768c7858 000b000a 000d000c 000f000e 00110010
768c7868 00130012 00150014 00170016 00190018
768c7878 001b001a 001d001c 001f001e 00210020
768c7888 00230022 00250024 00270026 00290028
768c7898 002b002a 002c0000 002e002d 0030002f
768c78a8 00320031 00340033 00360035 00380037
768c78b8 003a0039 003c003b 003e003d 0040003f
```

---

So, we got the list of RVAs now. Each of these RVAs when added to the base address of gdi32.dll will point to the Function Name string. Let's check by taking the first RVA from this list: 00030002

```

0:000> dd kernel32 + 00030002
76840002 74636572 4579726f 90004178 48539090
76840012 61657243 65446574 6c756166 74784574
76840022 74636172 6e6f6349 48539000 61657243
76840032 65446574 6c756166 6e6f4374 74786574
76840042 756e654d 48539000 61657243 61446574
76840052 624f6174 7463656a 48539000 6e616843
76840062 6f4e6567 79666974 70737553 52646e65
76840072 6d757365 48530065 6e616843 6f4e6567
0:000> dd kernel32.dll + 00030002
76840002 74636572 4579726f 90004178 48539090
76840012 61657243 65446574 6c756166 74784574
76840022 74636172 6e6f6349 48539000 61657243
76840032 65446574 6c756166 6e6f4374 74786574
76840042 756e654d 48539000 61657243 61446574
76840052 624f6174 7463656a 48539000 6e616843
76840062 6f4e6567 79666974 70737553 52646e65
76840072 6d757365 48530065 6e616843 6f4e6567

```

## The IMPORT Section

The Import section contains information about all the functions imported by executable from DLLs. This information is stored in several data structures. The most important data structure is import directory and Import Address Table. In some executable, there are also bound\_import and delay\_Import directories, in which the important one is bound\_import.

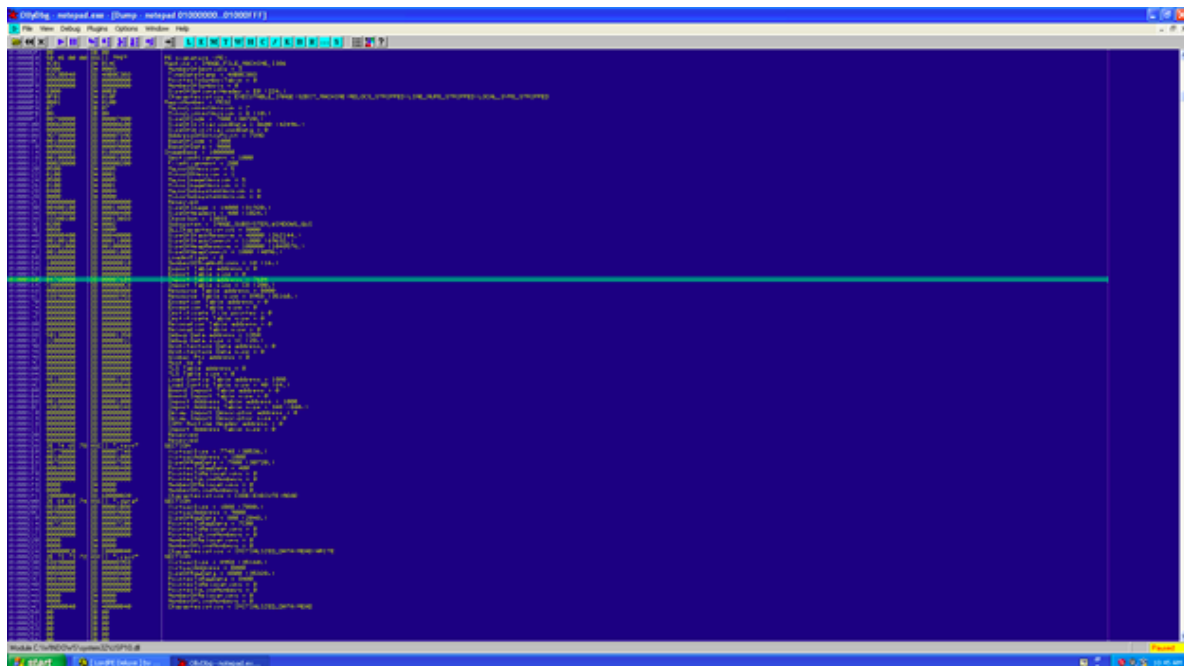
The Windows loader is responsible for loading all of the DLLs that application uses and mapping them into process address space. It has to find all imported functions in the various DLL and make them available for the executable being loaded.

### IMPORT DIRECTORY:

The IMPORT Directory structure is 80th offset of PE header. Check the below ollydbg screen.

010000E0+80 = 01000160





The Import Directory is actually an array of **IMAGE\_IMPORT\_DESCRIPTOR** structures.

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {

union {

    DWORD Characteristics; // 0 for terminating null import
    descriptor

    DWORD OriginalFirstThunk; // RVA to original unbound IAT
    (PIMAGE_THUNK_DATA)

} DUMMYUNIONNAME;

    DWORD TimeDateStamp; // 0 if not bound,
    // -1 if bound, and real date\time stamp
    // in IMAGE_DIRECTORY_ENTRY_BOUND_IMPORT (new BIND)
    // O.W. date/time stamp of DLL bound to (Old BIND)

    DWORD ForwarderChain; // -1 if no forwarders

    DWORD Name;

    DWORD FirstThunk; // RVA to IAT (if bound this IAT has actual
    addresses)

}
```

Each structure is 20 bytes and contains information about a DLL which our PE file imports functions from. Let's fire up windbg and extract the detailed symbol

```

i:000> !dh 006a0000 -f
File Type: EXECUTABLE IMAGE
FILE HEADER VALUES
14C machine (i386)
4 number of sections
.A5BC60F time date stamp Mon Jul 13 16:41:03 2009
0 file pointer to symbol table
0 number of symbols
E0 size of optional header
102 characteristics
Executable
32 bit word machine

OPTIONAL HEADER VALUES
10B magic #
9.00 linker version
A800 size of code
22400 size of initialized data
0 size of uninitialized data
3689 address of entry point
1000 base of code
----- new -----
06a0000 image base
1000 section alignment
200 file alignment
2 subsystem (Windows GUI)
6.01 operating system version
6.01 image version
6.01 subsystem version
30000 size of image
400 size of headers
39741 checksum
0040000 size of stack reserve
0011000 size of stack commit
0100000 size of heap reserve
0001000 size of heap commit
8140 DLL characteristics
Dynamic base
NX compatible
Terminal server aware
0 address (size) of Export Directory
A048 address (size) of Import Directory
F000 address (size) of Resource Directory
0 address (size) of Exception Directory
0 address (size) of Security Directory
2F000 address (size) of Base Relocation Directory
B62C address (size) of Debug Directory
0 address (size) of Description Directory
0 address (size) of Special Directory
0 address (size) of Thread Storage Directory
6D5B address (size) of Load Configuration Directory
77B address (size) of Bound Import Directory
1000 400 address (size) of Import Address Table Directory
0 address (size) of Import-Export Directory
0 address (size) of COR20 Header Directory
0 address (size) of Reserved Directory

```

Here 1000 is the RVA to the Image base(06a0000).so Image Base Address +RVA will point to Import Address Table. We will Use the “dps” command to dump the address at that offset and try to resolve them to symbols. You will likely need to run dps a number of times to cycle through the entire import table.we already knew that size is 400,so we will force windbg using this address to show all relevant addresses. We can also see the address.

```

C:\Windows\System32\cmd.exe - WinDbg 6.6.0.6002.17286 x86
dps 06a0000+1000
06a00000 00000000
06a00004 00000000
06a00008 00000000
06a0000C 00000000
06a00010 00000000
06a00014 00000000
06a00018 00000000
06a0001C 00000000
06a00020 00000000
06a00024 00000000
06a00028 00000000
06a0002C 00000000
06a00030 00000000
06a00034 00000000
06a00038 00000000
06a0003C 00000000
06a00040 00000000
06a00044 00000000
06a00048 00000000
06a0004C 00000000
06a00050 00000000
06a00054 00000000
06a00058 00000000
06a0005C 00000000
06a00060 00000000
06a00064 00000000
06a00068 00000000
06a0006C 00000000
06a00070 00000000
06a00074 00000000
06a00078 00000000
06a0007C 00000000
06a00080 00000000
06a00084 00000000
06a00088 00000000
06a0008C 00000000
06a00090 00000000
06a00094 00000000
06a00098 00000000
06a0009C 00000000
06a000A0 00000000
06a000A4 00000000
06a000A8 00000000
06a000AC 00000000
06a000B0 00000000
06a000B4 00000000
06a000B8 00000000
06a000BC 00000000
06a000C0 00000000
06a000C4 00000000
06a000C8 00000000
06a000CC 00000000
06a000D0 00000000
06a000D4 00000000
06a000D8 00000000
06a000DC 00000000
06a000E0 00000000
06a000E4 00000000
06a000E8 00000000
06a000EC 00000000
06a000F0 00000000
06a000F4 00000000
06a000F8 00000000
06a000FC 00000000
06a00100 00000000
06a00104 00000000
06a00108 00000000
06a0010C 00000000
06a00110 00000000
06a00114 00000000
06a00118 00000000
06a0011C 00000000
06a00120 00000000
06a00124 00000000
06a00128 00000000
06a0012C 00000000
06a00130 00000000
06a00134 00000000
06a00138 00000000
06a0013C 00000000
06a00140 00000000
06a00144 00000000
06a00148 00000000
06a0014C 00000000
06a00150 00000000
06a00154 00000000
06a00158 00000000
06a0015C 00000000
06a00160 00000000
06a00164 00000000
06a00168 00000000
06a0016C 00000000
06a00170 00000000
06a00174 00000000
06a00178 00000000
06a0017C 00000000
06a00180 00000000
06a00184 00000000
06a00188 00000000
06a0018C 00000000
06a00190 00000000
06a00194 00000000
06a00198 00000000
06a0019C 00000000
06a001A0 00000000
06a001A4 00000000
06a001A8 00000000
06a001AC 00000000
06a001B0 00000000
06a001B4 00000000
06a001B8 00000000
06a001BC 00000000
06a001C0 00000000
06a001C4 00000000
06a001C8 00000000
06a001CC 00000000
06a001D0 00000000
06a001D4 00000000
06a001D8 00000000
06a001DC 00000000
06a001E0 00000000
06a001E4 00000000
06a001E8 00000000
06a001EC 00000000
06a001F0 00000000
06a001F4 00000000
06a001F8 00000000
06a001FC 00000000
06a00200 00000000
06a00204 00000000
06a00208 00000000
06a0020C 00000000
06a00210 00000000
06a00214 00000000
06a00218 00000000
06a0021C 00000000
06a00220 00000000
06a00224 00000000
06a00228 00000000
06a0022C 00000000
06a00230 00000000
06a00234 00000000
06a00238 00000000
06a0023C 00000000
06a00240 00000000
06a00244 00000000
06a00248 00000000
06a0024C 00000000
06a00250 00000000
06a00254 00000000
06a00258 00000000
06a0025C 00000000
06a00260 00000000
06a00264 00000000
06a00268 00000000
06a0026C 00000000
06a00270 00000000
06a00274 00000000
06a00278 00000000
06a0027C 00000000
06a00280 00000000
06a00284 00000000
06a00288 00000000
06a0028C 00000000
06a00290 00000000
06a00294 00000000
06a00298 00000000
06a0029C 00000000
06a002A0 00000000
06a002A4 00000000
06a002A8 00000000
06a002AC 00000000
06a002B0 00000000
06a002B4 00000000
06a002B8 00000000
06a002BC 00000000
06a002C0 00000000
06a002C4 00000000
06a002C8 00000000
06a002CC 00000000
06a002D0 00000000
06a002D4 00000000
06a002D8 00000000
06a002DC 00000000
06a002E0 00000000
06a002E4 00000000
06a002E8 00000000
06a002EC 00000000
06a002F0 00000000
06a002F4 00000000
06a002F8 00000000
06a002FC 00000000
06a00300 00000000
06a00304 00000000
06a00308 00000000
06a0030C 00000000
06a00310 00000000
06a00314 00000000
06a00318 00000000
06a0031C 00000000
06a00320 00000000
06a00324 00000000
06a00328 00000000
06a0032C 00000000
06a00330 00000000
06a00334 00000000
06a00338 00000000
06a0033C 00000000
06a00340 00000000
06a00344 00000000
06a00348 00000000
06a0034C 00000000
06a00350 00000000
06a00354 00000000
06a00358 00000000
06a0035C 00000000
06a00360 00000000
06a00364 00000000
06a00368 00000000
06a0036C 00000000
06a00370 00000000
06a00374 00000000
06a00378 00000000
06a0037C 00000000
06a00380 00000000
06a00384 00000000
06a00388 00000000
06a0038C 00000000
06a00390 00000000
06a00394 00000000
06a00398 00000000
06a0039C 00000000
06a003A0 00000000
06a003A4 00000000
06a003A8 00000000
06a003AC 00000000
06a003B0 00000000
06a003B4 00000000
06a003B8 00000000
06a003BC 00000000
06a003C0 00000000
06a003C4 00000000
06a003C8 00000000
06a003CC 00000000
06a003D0 00000000
06a003D4 00000000
06a003D8 00000000
06a003DC 00000000
06a003E0 00000000
06a003E4 00000000
06a003E8 00000000
06a003EC 00000000
06a003F0 00000000
06a003F4 00000000
06a003F8 00000000
06a003FC 00000000
06a00400 00000000
06a00404 00000000
06a00408 00000000
06a0040C 00000000
06a00410 00000000
06a00414 00000000
06a00418 00000000
06a0041C 00000000
06a00420 00000000
06a00424 00000000
06a00428 00000000
06a0042C 00000000
06a00430 00000000
06a00434 00000000
06a00438 00000000
06a0043C 00000000
06a00440 00000000
06a00444 00000000
06a00448 00000000
06a0044C 00000000
06a00450 00000000
06a00454 00000000
06a00458 00000000
06a0045C 00000000
06a00460 00000000
06a00464 00000000
06a00468 00000000
06a0046C 00000000
06a00470 00000000
06a00474 00000000
06a00478 00000000
06a0047C 00000000
06a00480 00000000
06a00484 00000000
06a00488 00000000
06a0048C 00000000
06a00490 00000000
06a00494 00000000
06a00498 00000000
06a0049C 00000000
06a004A0 00000000
06a004A4 00000000
06a004A8 00000000
06a004AC 00000000
06a004B0 00000000
06a004B4 00000000
06a004B8 00000000
06a004BC 00000000
06a004C0 00000000
06a004C4 00000000
06a004C8 00000000
06a004CC 00000000
06a004D0 00000000
06a004D4 00000000
06a004D8 00000000
06a004DC 00000000
06a004E0 00000000
06a004E4 00000000
06a004E8 00000000
06a004EC 00000000
06a004F0 00000000
06a004F4 00000000
06a004F8 00000000
06a004FC 00000000
06a00500 00000000
06a00504 00000000
06a00508 00000000
06a0050C 00000000
06a00510 00000000
06a00514 00000000
06a00518 00000000
06a0051C 00000000
06a00520 00000000
06a00524 00000000
06a00528 00000000
06a0052C 00000000
06a00530 00000000
06a00534 00000000
06a00538 00000000
06a0053C 00000000
06a00540 00000000
06a00544 00000000
06a00548 00000000
06a0054C 00000000
06a00550 00000000
06a00554 00000000
06a00558 00000000
06a0055C 00000000
06a00560 00000000
06a00564 00000000
06a00568 00000000
06a0056C 00000000
06a00570 00000000
06a00574 00000000
06a00578 00000000
06a0057C 00000000
06a00580 00000000
06a00584 00000000
06a00588 00000000
06a0058C 00000000
06a00590 00000000
06a00594 00000000
06a00598 00000000
06a0059C 00000000
06a005A0 00000000
06a005A4 00000000
06a005A8 00000000
06a005AC 00000000
06a005B0 00000000
06a005B4 00000000
06a005B8 00000000
06a005BC 00000000
06a005C0 00000000
06a005C4 00000000
06a005C8 00000000
06a005CC 00000000
06a005D0 00000000
06a005D4 00000000
06a005D8 00000000
06a005DC 00000000
06a005E0 00000000
06a005E4 00000000
06a005E8 00000000
06a005EC 00000000
06a005F0 00000000
06a005F4 00000000
06a005F8 00000000
06a005FC 00000000
06a00600 00000000
06a00604 00000000
06a00608 00000000
06a0060C 00000000
06a00610 00000000
06a00614 00000000
06a00618 00000000
06a0061C 00000000
06a00620 00000000
06a00624 00000000
06a00628 00000000
06a0062C 00000000
06a00630 00000000
06a00634 00000000
06a00638 00000000
06a0063C 00000000
06a00640 00000000
06a00644 00000000
06a00648 00000000
06a0064C 00000000
06a00650 00000000
06a00654 00000000
06a00658 00000000
06a0065C 00000000
06a00660 00000000
06a00664 00000000
06a00668 00000000
06a0066C 00000000
06a00670 00000000
06a00674 00000000
06a00678 00000000
06a0067C 00000000
06a00680 00000000
06a00684 00000000
06a00688 00000000
06a0068C 00000000
06a00690 00000000
06a00694 00000000
06a00698 00000000
06a0069C 00000000
06a006A0 00000000
06a006A4 00000000
06a006A8 00000000
06a006AC 00000000
06a006B0 00000000
06a006B4 00000000
06a006B8 00000000
06a006BC 00000000
06a006C0 00000000
06a006C4 00000000
06a006C8 00000000
06a006CC 00000000
06a006D0 00000000
06a006D4 00000000
06a006D8 00000000
06a006DC 00000000
06a006E0 00000000
06a006E4 00000000
06a006E8 00000000
06a006EC 00000000
06a006F0 00000000
06a006F4 00000000
06a006F8 00000000
06a006FC 00000000
06a00700 00000000
06a00704 00000000
06a00708 00000000
06a0070C 00000000
06a00710 00000000
06a00714 00000000
06a00718 00000000
06a0071C 00000000
06a00720 00000000
06a00724 00000000
06a00728 00000000
06a0072C 00000000
06a00730 00000000
06a00734 00000000
06a00738 00000000
06a0073C 00000000
06a00740 00000000
06a00744 00000000
06a00748 00000000
06a0074C 00000000
06a00750 00000000
06a00754 00000000
06a00758 00000000
06a0075C 00000000
06a00760 00000000
06a00764 00000000
06a00768 00000000
06a0076C 00000000
06a00770 00000000
06a00774 00000000
06a00778 00000000
06a0077C 00000000
06a00780 00000000
06a00784 00000000
06a00788 00000000
06a0078C 00000000
06a00790 00000000
06a00794 00000000
06a00798 00000000
06a0079C 00000000
06a007A0 00000000
06a007A4 00000000
06a007A8 00000000
06a007AC 00000000
06a007B0 00000000
06a007B4 00000000
06a007B8 00000000
06a007BC 00000000
06a007C0 00000000
06a007C4 00000000
06a007C8 00000000
06a007CC 00000000
06a007D0 00000000
06a007D4 00000000
06a007D8 00000000
06a007DC 00000000
06a007E0 00000000
06a007E4 00000000
06a007E8 00000000
06a007EC 00000000
06a007F0 00000000
06a007F4 00000000
06a007F8 00000000
06a007FC 00000000
06a00800 00000000
06a00804 00000000
06a00808 00000000
06a0080C 00000000
06a00810 00000000
06a00814 00000000
06a00818 00000000
06a0081C 00000000
06a00820 00000000
06a00824 00000000
06a00828 00000000
06a0082C 00000000
06a00830 00000000
06a00834 00000000
06a00838 00000000
06a0083C 00000000
06a00840 00000000
06a00844 00000000
06a00848 00000000
06a0084C 00000000
06a00850 00000000
06a00854 00000000
06a00858 00000000
06a0085C 00000000
06a00860 00000000
06a00864 00000000
06a00868 00000000
06a0086C 00000000
06a00870 00000000
06a00874 00000000
06a00878 00000000
06a0087C 00000000
06a00880 00000000
06a00884 00000000
06a00888 00000000
06a0088C 00000000
06a00890 00000000
06a00894 00000000
06a00898 00000000
06a0089C 00000000
06a008A0 00000000
06a008A4 00000000
06a008A8 00000000
06a008AC 00000000
06a008B0 00000000
06a008B4 00000000
06a008B8 00000000
06a008BC 00000000
06a008C0 00000000
06a008C4 00000000
06a008C8 00000000
06a008CC 00000000
06a008D0 00000000
06a008D4 00000000
06a008D8 00000000
06a008DC 00000000
06a008E0 00000000
06a008E4 00000000
06a008E8 00000000
06a008EC 00000000
06a008F0 00000000
06a008F4 00000000
06a008F8 00000000
06a008FC 00000000
06a00900 00000000
06a00904 00000000
06a00908 00000000
06a0090C 00000000
06a00910 00000000
06a00914 00000000
06a00918 00000000
06a0091C 00000000
06a00920 00000000
06a00924 00000000
06a00928 00000000
06a0092C 00000000
06a00930 00000000
06a00934 00000000
06a00938 00000000
06a0093C 00000000
06a00940 00000000
06a00944 00000000
06a00948 00000000
06a0094C 00000000
06a00950 00000000
06a00954 00000000
06a00958 00000000
06a0095C 00000000
06a00960 00000000
06a00964 00000000
06a00968 00000000
06a0096C 00000000
06a00970 00000000
06a00974 00000000
06a00978 00000000
06a0097C 00000000
06a00980 00000000
06a00984 00000000
06a00988 00000000
06a0098C 00000000
06a00990 00000000
06a00994 00000000
06a00998 00000000
06a0099C 00000000
06a009A0 00000000
06a009A4 00000000
06a009A8 00000000
06a009AC 00000000
06a009B0 00000000
06a009B4 00000000
06a009B8 00000000
06a009BC 00000000
06a009C0 00000000
06a009C4 00000000
06a009C8 00000000
06a009CC 00000000
06a009D0 00000000
06a009D4 00000000
06a009D8 00000000
06a009DC 00000000
06a009E0 00000000
06a009E4 00000000
06a009E8 00
```

Want to learn more? The InfoSec Institute [Ethical Hacking course](#) goes in-depth into the techniques used by malicious, black hat hackers with attention getting lectures and hands-on lab exercises. You leave with the ability to quantitatively assess and measure threats to information assets; and discover where your organization is most vulnerable to black hat hackers. Some features of this course include:

- Dual Certification - CEH and [CPT](#)
- 5 days of Intensive Hands-On Labs
- CTF exercises in the evening

FIRST NAME *	LAST NAME *
<input type="text"/>	<input type="text"/>
COMPANY	EMAIL *
<input type="text"/>	<input type="text"/>
PHONE *	JOB TITLE *
<input type="text"/>	<input type="text"/>
WHO WILL FUND YOUR TRAINING? *	
<input type="text"/>	
<input type="button" value="FIND PRICING FOR THIS COURSE"/>	

It is good to know how to fetch an IAT of a PE image since we can use this output to detect any sort of IAT Hooks. We will look into IAT hooks later which is a technique used by rootkits to take control of the functions in a DLL by overwriting the function pointers in the IAT.

Now let's discuss different structure of IAT. The first member **OriginalFirstThunk**, which is a DWORD union, may at one time have been a set of flags. However, Microsoft changed its meaning and never bothered to update WINNT.H. This field really contains the RVA of an array of **IMAGE\_THUNK\_DATA** structures.

The **TimeDateStamp** member is set to zero unless the executable is bound when it contains -1 (see below). The **ForwarderChain** member was used for old-style binding and will not be considered here. The last member **FirstThunk**, also contains the RVA of an array of DWORD-sized IMAGE\_THUNK\_DATA structures – **a duplicate of the first array.**

```
typedef struct _IMAGE_THUNK_DATA32 {  
  
    union {  
  
        DWORD ForwarderString; // PBYTE  
  
        DWORD Function; // PDWORD  
  
        DWORD Ordinal;  
  
        DWORD AddressOfData; // PIMAGE_IMPORT_BY_NAME  
  
    } u1;  
};
```

Each **IMAGE\_THUNK\_DATA** is a DWORD union that effectively only has one of 2 values. In the file on disk it either contains the ordinal of the imported function or an RVA to an **IMAGE\_IMPORT\_BY\_NAME** structure. Once loaded the ones pointed at by FirstThunk are overwritten with the addresses of imported functions – this becomes the Import Address Table.

Each IMAGE\_IMPORT\_BY\_NAME structure is defined as follows:

```
typedef struct _IMAGE_IMPORT_BY_NAME {  
  
    WORD Hint;  
  
    BYTE Name[1];  
  
};
```

**Hint** – contains the index into the Export Address Table of the DLL the function resides in. This field is for use by the PE loader so it can look up the function in the DLL's Export Address Table quickly.

**Name1** – contains the name of the imported function. The name is a null-terminated ASCII string.

### Functions Exported by Ordinal Only

As we discussed in the export section, some functions are exported by ordinal only. In this case, there will be no IMAGE\_IMPORT\_BY\_NAME structure for that function in the caller's module. Instead, the IMAGE\_THUNK\_DATA for that function contains the ordinal of the function.

### Bound Imports

When the loader loads a PE file into memory, it examines the import table and loads the required DLLs into the process address space. Then it walks the array pointed at by FirstThunk and replaces the IMAGE\_THUNK\_DATAs with the real addresses of the import functions. This step takes time. If somehow the programmer can predict the addresses of the functions correctly, the PE loader doesn't have to fix the IMAGE\_THUNK\_DATAs each time the PE file is run as the correct address is already there. Binding is the product of that idea.

### The Bound Import Directory

The information the loader uses to determine if bound addresses are valid is kept in a IMAGE\_BOUND\_IMPORT\_DESCRIPTOR structure.

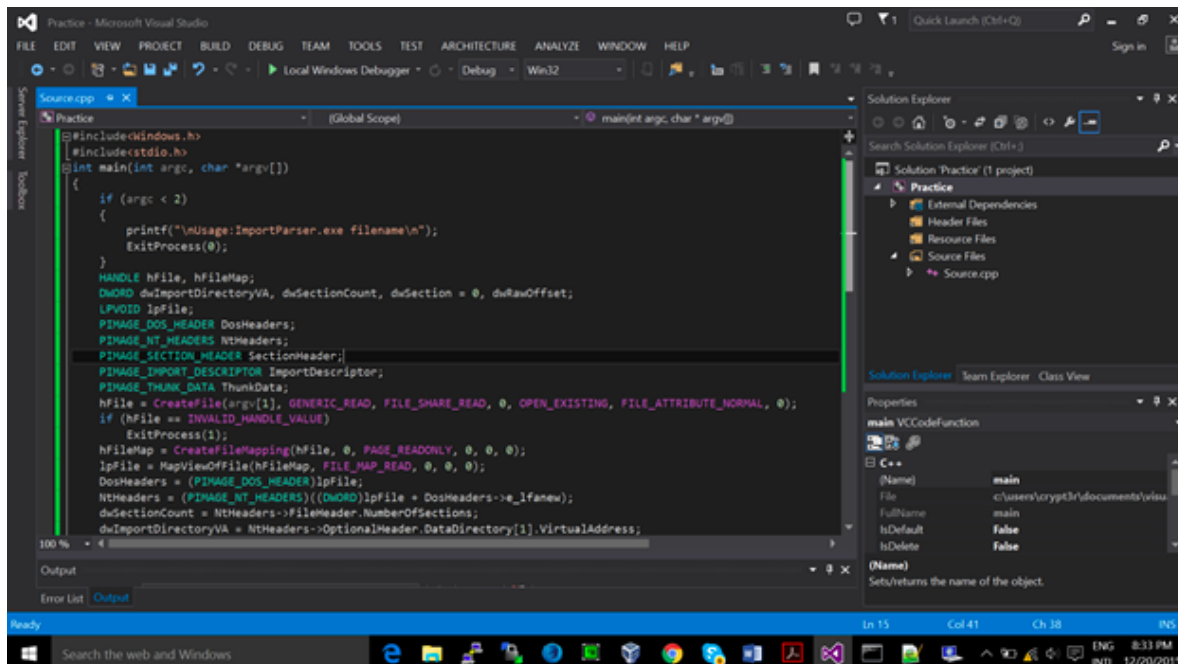
### The Loader

When an executable is run, the windows loader first create virtual address space for the process and maps the executable from disk into process's address space. it tries to load the image at the preferred base address but relocates it if that address is already occupied. The loader goes through section table and maps each section at the address calculated by adding RVA of the section to its base address.

The Import table is then checked and any other required DLL are mapped into process address Space. After all the dll are located and mapped. it will check each DLL's Export section and IAT is fixed to point to the actual imported function address. Once all loaded modules are loaded, execution passed to the app's entry point

### Automation

First we will develop a IAT parser using WINDOWS API via c/c++. we will extract relevant info about a file



ImportParser.exe abc.exe

DLL Name : KERNEL32.DLL

Function : LoadLibraryA

Function : GetProcAddress

Function : VirtualProtect

Function : VirtualAlloc

Function : VirtualFree

Function : ExitProcess

DLL Name : GDI32.dll

Function : BitBlt

Please find the code here.



Remember: If you are compiling the code in 32 bit, then you have to run in 32 bit system only with DEP, ASLR disabled

Also we can print same info by using

python(<http://stackoverflow.com/questions/19325402/getting-iat-and-eat-from-pe>)

So this is our end of this part. We will discuss more interesting topics like the .tls virus, PE packer development in the next series.....



9

52



49



AUTHOR

Revers3r

Revers3r is a Information Security Researcher with considerable experience in Web Application Security, Vulnerability Assessment, Penetration Testing. He is also well-versed in Reverse Engineering, Malware Analysis. He's been a contributor to international magazines like Hakin9, Pentest, and E-Forensics. In his free time, he's contributed to the Response Disclosure Program. website: [www.vulnerableghost.com](http://www.vulnerableghost.com)

#### EDITORS CHOICE

- Malware Researcher's Handbook
- Deep Packet Inspection in Cloud Containers
- Exploiting CVE-2015-8562 (A New Joomla! RCE)

#### RELATED BOOT CAMPS

- Information Security
- Security Awareness
- CCNA

- ⊕ PMP
- ⊕ Microsoft
- ⊕ Incident Response
- ⊕ Information Assurance
- ⊕ 8570

#### MORE POSTS BY AUTHOR



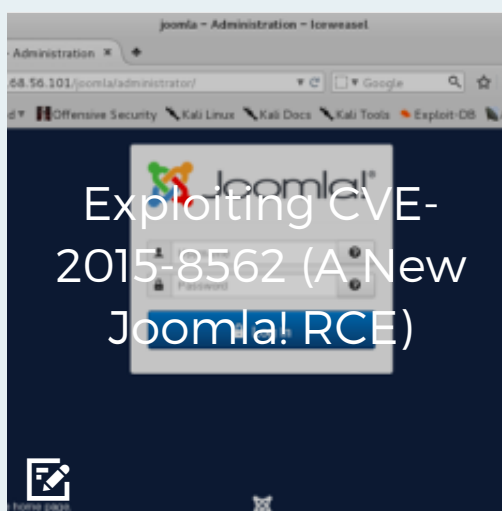
Malware Researcher's Handbook (Demystifying PE File)



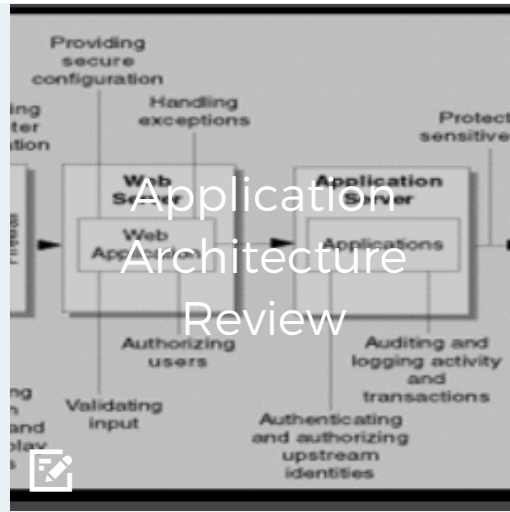
Malware Researcher's Handbook: Introduction



Fingerprinting: Identifying Applications





**0 Comments****InfoSec Institute Resources** **Login** ▾ **Recommend** **Share****Sort by Best** ▾

Be the first to comment.

 **Subscribe** **Add Disqus to your site** Add Disqus Add **Privacy****DISQUS**

## About InfoSec

InfoSec Institute is the best source for high quality [information security training](#). We have been training Information Security and IT Professionals since 1998 with a diverse lineup of relevant training courses. In the past 16 years, over 50,000 individuals have trusted InfoSec Institute for their

## Connect with us

Stay up to date with [InfoSec Institute](#) and [Intense School](#) - at [info@infosecinstitute.com](mailto:info@infosecinstitute.com)

 **Like** 567 **Follow @infosecedu**

## Join our newsletter

Get the latest news, updates & offers straight to your inbox.

**SUBSCRIBE**

professional development  
needs!

© INFOSEC RESOURCES 2015