
Interested in learning more about security?

# SANS Institute

## Security Consensus Operational Readiness Evaluation



# SCORE Security Checklist

There are a number of schools of thought on how to approach reversing malware. Some people jump right into dynamic analysis in effort to quickly learn what the specimen is doing so they can put rules in place on their network to stop it's functionality or see who else might be infected. Some of these people, will then perform static analysis to see if they may have missed something. Others leave it at the results found from their dynamic analysis.

Other people do static analysis first to fully understand the expected behavior so they know if something is happening with the sample when running it in a dynamic lab other than what is expected. They will then run dynamic analysis on it to see if their findings are correct.

Some people just submit the sample to places like Virus Total, Anubis, or CWSanbox, just to name a few . They then take action based on the results they get back from these tools. Finally there are some that just submit it to their Anti Virus vendor and wait for a signature to be released.

There is nothing wrong with any of these methods. Most are done because of either lack of time, skills or understanding of how to reverse malware. Some may think, why reinvent the wheel? This is all OK.

**Overall process of static analysis:**

The first step in your process should be to start a log or collection of the details you are about to find. Some people do this is a text file, others may just jot things down in a notebook. Others like to use a mind mapping software such as FreeMind. Lenny Zeltser, a SANS instructor and an overall excellent resource for reversing knowledge, has freely released a template for FreeMind specifically for analyzing malicious code. You can download it here.

If you are looking for very good training in this area, Lenny also offers a few courses with SANS that can be taken online or at a conference. He has specifically created, along with others, the Forensics 610: Reverse Engineering Malware , and the Security 569: Combating Malware in the Enterprise courses. I highly encourage you to take these courses if you are interested in malware analysis.

It doesn't matter which method you choose to write your notes down, but it is very important that you do. Documenting this will help you to keep on track and assist you in writing reports of your analysis if you do this professionally. Additionally if you choose a method that allows you to compile all of your findings centrally, you will be able to see trends or recognize similar behaviors of samples that could help in reversing future samples that exhibit similar characteristics.

With that aside, take note of the system you found the malware on. Take notice of the operating system, patch level, applications installed etc. Write down where you found the code (i.e. C:\windows\system32). Add any information that may be relevant on how the code was found (i.e. the system administrator noticed the system was running slowly, or found the system blue screened).

Next take a hash of the file or files found. A hash, in this case, is a mathematical computation on the bits of a specimen. This will help with identification of other copies of the malware even if the name is changed. You can think of this much like doctors and scientists look for specific characteristics of a virus in the human body. That way, other doctors or scientists can identify the same virus in another person.

It is generally accepted to perform a MD5 hash on the file. Some people will also do a SHA1 or other computations as well. There is also a newer method called fuzzy hashing or peicewise hashing that can be done. This actually hashes portions of a file, rather than the whole thing. This method allows for identification of portions of the code which may be useful in catching malware that has changed just a little in order to avoid detection by a person or Anti Virus application for example.

There are many tools out there to do this. WinMD5 is an easy to use tool which is freely available. SSDEEP can be used for fuzzy hashing. What tool you use is up to you. The mathematical computations such as MD5 or SHA are standard equations. All of the applications that perform these actions then should produce the same result. Some operating systems such as *nix varieties have these tools built right in on the command line such as md5 or shasum.

The next step that is good to take is to compare the hash values that you found with sites on the Internet to see if there are any matches or similar hashes found by others. This can go a long way in saving you time and effort if it is a known specimen. You can copy and paste your hash value into sites such as Virus Total, or Threat Expert to name a few. You can also run this through an internal database you might have to see if there are any hits. The purpose of this is to save yourself time and energy if this sample has already been analyzed and identified by someone else.

Next you will want to classify your specimen. This classification will be the file type, format, target architecture, language used, and compiler used. These characteristics will let you know what systems may be vulnerable and which may not. For example, if you find a PE file format, you can assume that *nix systems will most likely not be vulnerable to the sample. This is because the PE format is used on Microsoft Windows systems.

TrIDNet with the latest definition files is a good place to start this classification. Minidumper is another free tool which works well. One thing you will notice is that most analysts generally run files though similar tools and compare the output. Sometimes one tool works better than other and will give you more information which may be helpful. Many have also seen malware samples that are coded in such a way to recognize tools and change their behavior if they realize they are being analyzed.

GT2 is another good tool to run the sample through in this stage as well. This application attempts to identify the compiler, DLLs and other requirements for the code being analyzed.

You will want to scan the file next in attempt to see if it hits any known signatures. You can do this with multiple Anti Virus applications if you have multiple on your lab systems. You can also take this time to submit it to online sources which run the sample through a number of anti virus applications in effort to see if it's known. Depending on your organization, you may not want to submit the samples to these sources as they share their information freely with the public and anti virus vendors. If this sample is targeted in any way, this can blow your cover that you have found and are analyzing the sample.

If that is not a concern for you, some good examples of these services are Virus Total, Virscan, or Jotti. Again, this is just to name a few, there are many other good choices out there. Pick the ones you like and are comfortable with.

Another step to take is then to start to dig a little deeper. You will want to start looking for executable type, dll's called, exports, imports, strings etc. This information will start to reveal characteristics of the sample which may lead you to get a better understanding of what it does. For example, if you run strings on the file you may reveal IP addresses used for call back and download components, you may find information that lets you know the file is packed or encrypted in some way, you may find other files that are needed or used in the infection. etc.

Some of the tools that you can use to see this information are BinText, dumpbinGUI, or DependencyWalker to name a few. Again, there are many more out there. These are just some ideas of tools that some analysts use. For example if you are running on a *nix server, the command strings will work on most distributions right out of the box. Just open a terminal and run strings filename (where filename is the name of the file you are looking at) . You can also use the file command. Type file filename (again where filename is the name of the file you are looking at) to reveal the file type you are working with.

The next stage is to look for packing or encryption. This is often where the process becomes difficult. Some packers are easy to defeat. Others are not. Some encryption are easily seen and reversed. Others are custom and difficult. This is the time when a lot of analysts will enter into dynamic analysis. The reason for this is that they may not be able to unpack or decrypt the file by hand. In order for the file to run on a system, it needs to be unpacked or decrypted in memory to function.

Some tools to use to look for this type of information are PEiD, PE Detective, or Mandiant's Red Curtain. These tools are capable of detecting known packers, encryption or behaviors that are common to malicious files to make them difficult to analyze. You may have found this information out already from the output of previous tools.

Here are the steps with screen shots. For the sample, I just went to Malware Domain List and grabbed a suspected malware sample from that page.

I will warn you now as always. You should only be downloading and executing the following sample in a lab environment. If you don't know how to do that, please reference material that can instruct you in this. You can check http://internetopenurla.blogspot.com for examples on how to do this. A lot of this document is also available there along with other step by step instructions.

First I'm going to boot my lab environment and pull the sample into it. Then I'm going to run a hash application to get the digital fingerprint of the binary.

I just installed the Malcode Analyst Pack. This will give us a handy right click menu to obtain the MD5 Hash and Strings. So I right click the file and choose MD5 Hash. Here is the result:

As you can see, the hash value of our sample is 121340AA444B4D4153510C0BE58D4D61. We will jot this down in our notes.

Next we will take our fuzzy hash with SSDEEP. In this example we will use the SSDEEP Front End. This is a nice little GUI to make things easy. When we first open the application, we need to choose Create or Append Hash Value. We will choose Single File as our Input. Click the Choose Input button and navigate to where you have your sample, as seen below:

Next, we click the Choose Output button. This will open an Explorer window, where you can choose where you would like your output to go. Here we are going to choose to put it on the Desktop so we can find it easy. I generally name the file: <filename>_exe (or dll if it's a dll). This is just my method to know what the name is and what that sample it goes with. Click the Open button. You then need to click the Execute button.

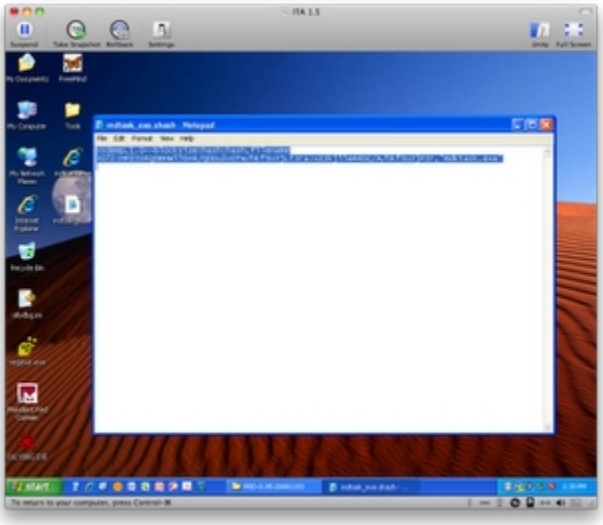
A dialog box will pop up telling you that you are about to run a batch file, and that a DOS box is about to pop up. Click the yes button to continue.

A DOS box will pop up for a second and go away. You will be left back at the main screen of the SSDEEP Front End. You can choose Exit at this time.

You now have the file on your desktop. Double Click to open it. Windows will ask you what application you would like to use to open the file. Choose to select a program from a list and click OK. I normally use the Notepad application for these, but you can use any text editor or viewer that you want. I generally also choose to always use the selected program to open this type of file. That way I don't need to choose every time.

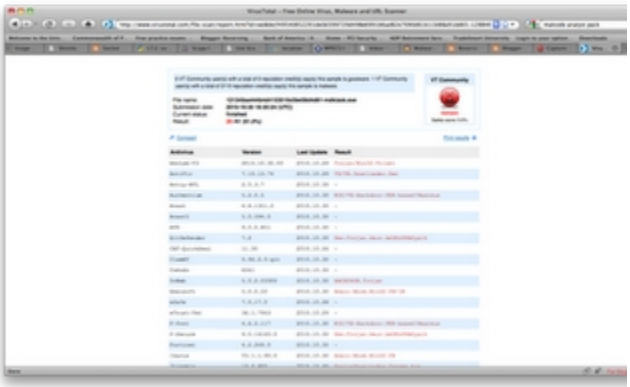
As you can see from the screen shot below. Our fuzzy hash is

3072:HPZJsRgSmHWii6X4/QDDu3vDTw/hkfSUJjLTJra:vZJkjiie44DC/A/hkfSUJjPJr,"mdktask.exe"



The next thing we are going to do is take our Hash value and search Virus Total to see if anyone has submitted this sample before. So we navigate to http://www.virustotal.com. We will click the Search link and paste our Hash value in and hit search.

As you can see from the following screen shot someone has submitted this sample and 25 our of 41 anti virus applications say it's a virus.

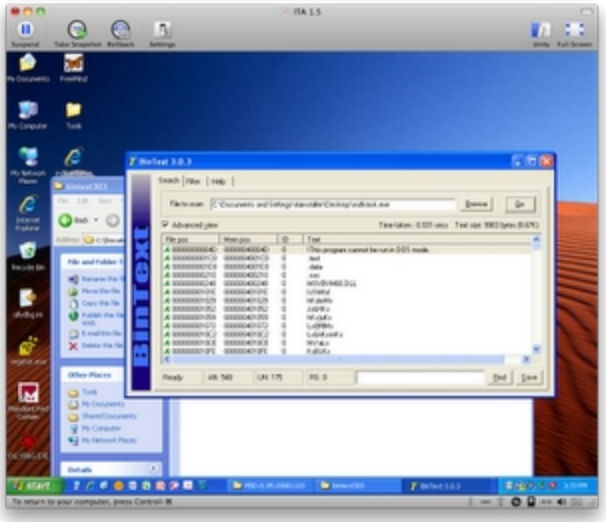


Next, we want to classify the file. To do this, we are going to use TRiD. Ensure you download the latest definition files from here. Once you run TriD, you will need to point to your definition xml file. Choose the Browse button on the bottom right of the application. Navigate to your definition files and choose the first listed xml file in the directory. I generally put the xml files in a folder called Defs in the TRiD application folder.

You will want to choose Browse on the top to choose our sample we want to analyze. After choosing the file, we click the Analyze button and get the results. As you can see below we have a match of 86% of a Windows 32 Executable file, potentially written in Visual Basic 6.

Normally at this point, we would run the sample through a few other similar applications to see if anything new is found. We have cut this out for brevity. I'm also not going to upload the sample to any sites to be scanned for known virus signatures. Mainly because I've seen by our search on Virus Total that we can be pretty sure it is known.

The next step we will do is to open the file with BinText. Choose the Browse button and choose our sample. After that we hit the go button. This will show us some of the strings available in the binary file. In some cases you may not see what you like here due to packing or encryption. We will go into that more later. In the case of this sample we are able to see a good bit of detail as seen below:



Below are some of the things I saw that should be added to our notes as items of interest:

Form1. This tells us that potentially there is a visual form

Timer1. This lets us know there is a timer or countdown of some sort.

Winsock.

WinsockAPI. These two tell us there is some sort of network component.

modSMTP. This would let us know there could be an email component which starts to corroborate what we have found from our search on Virus Total earlier.

mod_Variaveis. A quick Google of this word looks like it translates to variables from Portuguese. We now have an idea of where it may have come from. Maybe from Brazil or somewhere like that.

getpeername. This function will retrieve a name of a socket that was created. This starts to show there are more facts to prove network connectivity.

I could go through each, but I'm going to shorten this to only pull out the remainder of very interesting points that I see.

C:\Arquivos de programas\Messenger\msmsgs.exe\3 (Microsoft Messenger...interesting)

DownloadFile. Looks like we are getting more malware.

Crypt. Looks like some hashing or encryption going on.

GetWinPlatform. Looking for a specific Windows version maybe?

strEmailTO

strEmailTO1

strEmailTO2

strEmailTO3. Now we see some email capabilities which shows us why VirusTotal called this an email worm possibly

D:\Programas Daniel\infect\infect Interlig - rato\Project1.vbp. Humm is this guy called Daniel? gatta love when people don't clean up :)

GOD DAMNIT, the internet doesn't work... Little bit of error checking?

catia@oticasopcao.com. Source Email maybe?

showbol2010.log. Log for us to see how things went maybe?

http://www.youtube.com/. Maybe a link used in the email?

www.youtube.com. A link to more malware sent in the email possibly.

InternetBanking.exe. The next executable file name to download if you click the link or when the executable file executes. InternetBanking is interesting. Maybe we have something trying to steal banking credentials?

atualizahook.cfg. A config file on how to set things up?

Subject:

From:  more email functions

EHLO

AUTH LOGIN

MAIL FROM:<

RCPT TO:<. Email server communications.

Norton AntiVirus. Possibly to look like it's been scanned in transit?

lhe enviou o link do video no youtube. sent the video link on youtube in Portuguese.

This program cannot be run in DOS mode. We know this shows up at the beginning of PE files, but this is in the middle. Maybe this is our worm sending a copy of itself. Going through the rest of that section takes use through what we just went through previously. We may assume this is the case until we find out more.

Everything we have seen, and believe me there is a lot more in there, shows us that our results at Virus Total were pretty close so far. We could finish by looking at the PE format

with PEiD or attempt to look for packing or encryption, but I think you can see we really don't need to at this time.

I will end this example here. As you can see we didn't need to know any programming to find out what this thing does so far. For the APIs we may not have known, a quick search on Google or MSDN gave use the capability of those pieces of code. Not all code is this simple and to be honest I'm happy I choose one so simple randomly as it made our day a bit easier.

Next you might want to take this sample into a debugger or disassembler. Some example applications which are probably most popular are IDA Pro and OllyDBG. These are 2 of the more popular tools in this field. Our next section will get into these applications.

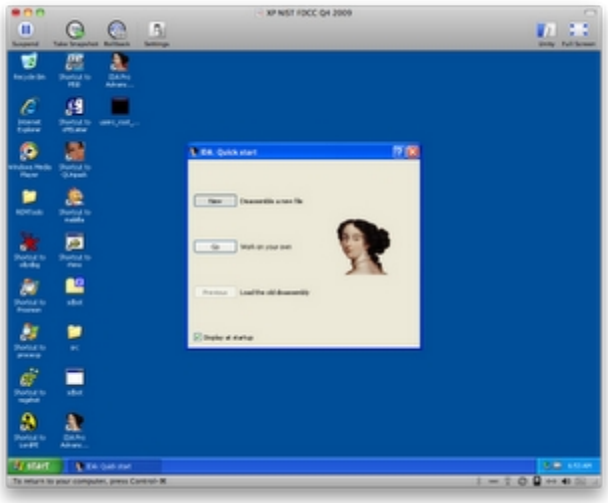
IDA Pro is a disassembler application that is commonly used in the reverse engineering field. There are many other applications like this, but if you plan to do this as a job, it would be a good idea to at least learn this interface in my opinion. Next week we will look at another good and similar tool called OllyDBG.

In my example I actually am running the latest version of IDA Pro and it's sister product called Hex-Rays. Hex-Rays is a decompiler application which adds a nice feature set to IDA. You can download a free trial of IDA Pro to see what the newer version offers you. Alternately, they offer a free version which is a few features behind. You can download that here. You can do most of what we will go over on this paper with the free version. If you are serious about reverse engineering, or do it for a living I would highly recommend getting the Hex-Rays add on. It really breaks out code in a nice readable format, especially for someone that may not be as strong in assembly programming. This is just my opinion, so take it for what it's worth. Overall your really looking at under $4,000 bucks for a set of tools that is going to save you a ton of time once you become familiar with them.

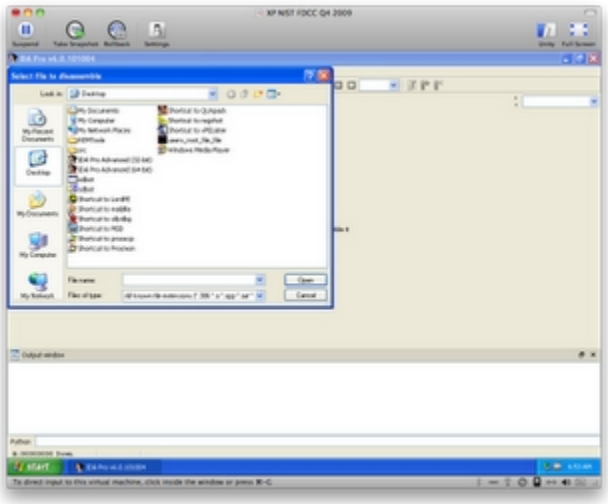
So on to the analysis. As always these links direct you to known malicious software. We hold no responsibility on your machine getting infected to a point where you can not recover or credentials that may be stolen due to improper handling. Please only analyze this and all samples in a secured lab environment.

I went out to grab a new file from Malware Domain List. The malware I downloaded for this example can be found here. (note: by the time you read this paper, this sample may not be available any longer. This is due to the fact that servers which are infected generally get taken down or cleaned up. We cannot get this sample for you if you are unable to obtain it. Sorry.)

Let's open this in IDA Pro. You can do this in a few different ways. You can drag the file onto the IDA Desktop shortcut or you can open IDA Pro, Choose to Dissamble a new file by clicking the New Button.



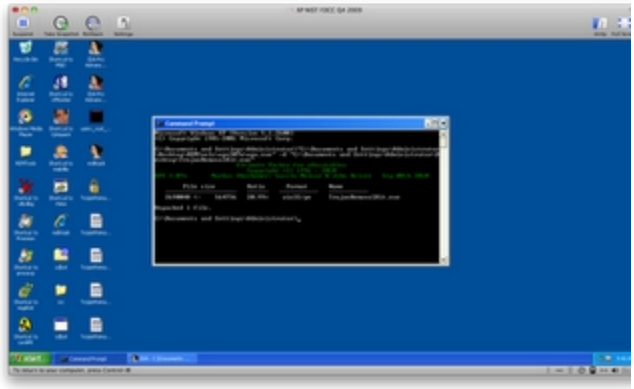
Navigate to the file you want to disassemble and choose open:



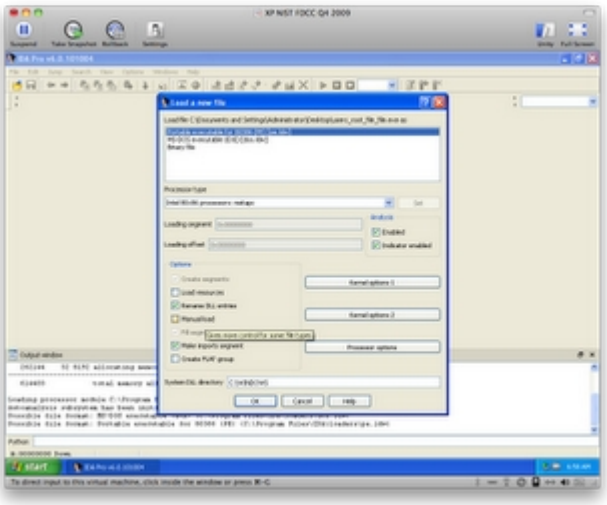
In most cases IDA will automatically recognize the processor type and options needed to open the file. You can modify these if you know that this is not correct from earlier inspection. A quick note is that normally I move on to IDA Pro or something similar after doing the steps we outlined previously. Therefore I may already know some things about the file such as what architecture the file is created for or if it is packed or not.

If the malware is packed or encrypted, which a lot of malware today is, there are many more steps which you may need to do before you can open the file in IDA and analyze it fully. This paper does not go into these details. We may add details on beating obfuscation at another time.

This specimen did appear to be packed with UPX. So we just unpacked it with the following command:

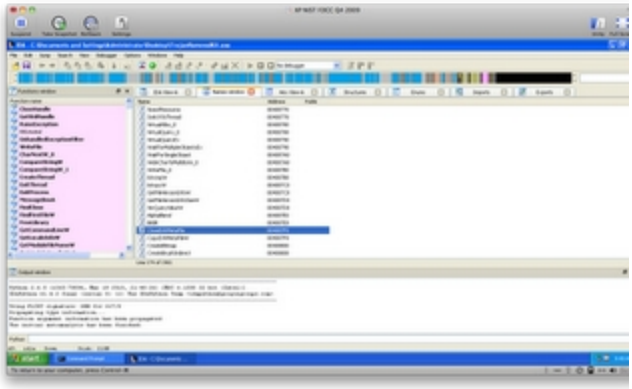


In this case, we will keep the default options and click OK:



Depending on how big the malware is, this may be quick or it may take some time. The first couple things I do once this is done is to take a look around at a few screens. This time taken helps me understand how much work may be ahead of me by how much IDA has recognized automatically. I will let you know that my screen shots will be a little different than yours if you are using the Freeware version. Version 6 looks a bit different even though all of the windows and options are still there. You may just need to hunt around to see where your version is displaying the same information.

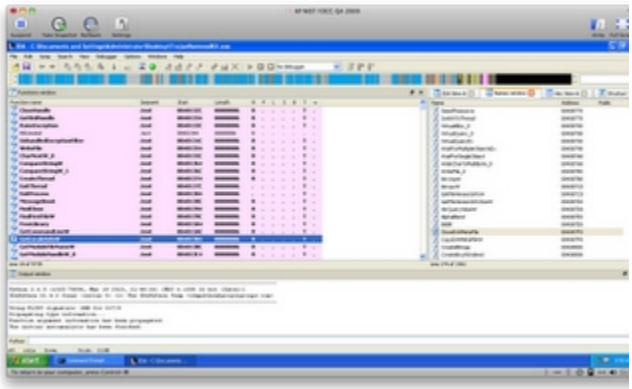
One of the first boxes I look at is the Names window. If you don't see the Names window, you can go to the view menu, click open subviews and choose names. Alternately you can hit Shift + F4. This Names window is going to show you names of APIs or type libraries it was able to recognize automatically. These may not be written with names that immediately tell their function, but sometimes they are. If they are pretty descriptive, then that may be less work that we need to do. Here is a screen shot of what this sample looks like when we first open it:



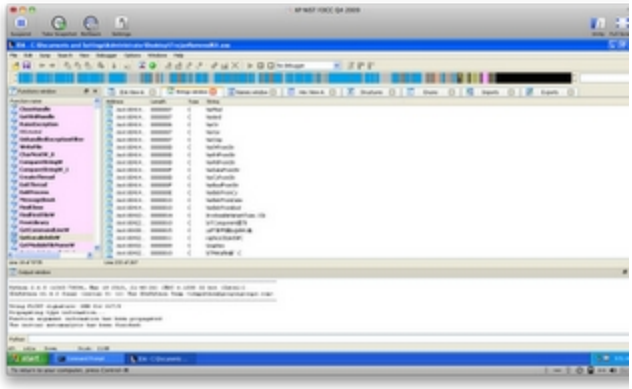
As you can see in the Names view, some of the names are pretty easy to distinguish, some are not. So you cannot always judge a function by that.

The next thing I generally look at is the Function subview. The functions window shows us the subroutines available in the sample. In some cases you will see names of these functions and in some cases, they will have generic names such as sub_xxxx where xxxx is the memory location of the routine. The reason IDA will show these names is if it matched a type library that IDA knows. The more named functions you have, the easier disassembly generally is.

In our case, IDA named a number of functions. This will help us significantly ahead because we don't need to figure out what they do necessarily. Here is the screen shot of the Functions window of our sample:

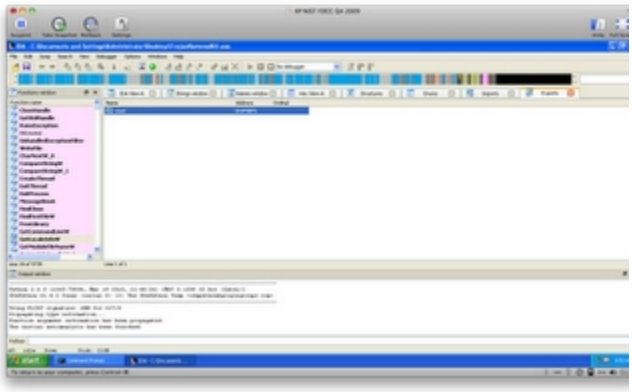


The next thing we can look at is the strings window. This will be a listing of all of the strings that IDA was able to recognize in the binary. We used strings in a previous section, so this may not be new news to you. If you do not see the stings window, you can go to view, click open sub views and choose Strings. Alternately you can hit Shift + F12. Here is a screen shot of the strings of our current sample:



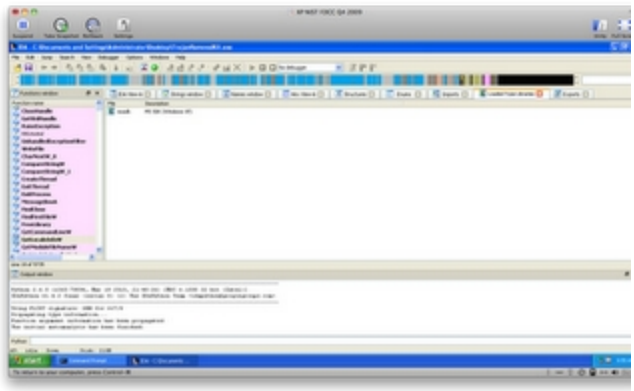
As we learned previously, we see some interesting things right away.  In this sample there are some strings which can help us understand some of the functions, but unlike the last sample, there are only a few which immediately make sense. We can learn a lot from the strings in a file, but a word of warning is that some malware authors will also put Red Herring strings in a file to throw you off.

If you look across the top of the view window, but under the command menus, you see a multi colored line. This is your binary time line so to speak. This will show you where you are in the binary at the current time. You will notice a little yellow arrow. This arrow shows you exactly where you are at the moment. As you can see in the following screen shot, IDA dumped us off at a function called Start. This is because IDA recognized this function as a potential entry point into the binary. You can view the Exports sub view to see other possible entry points. In our case we only have one Export listed at this time.



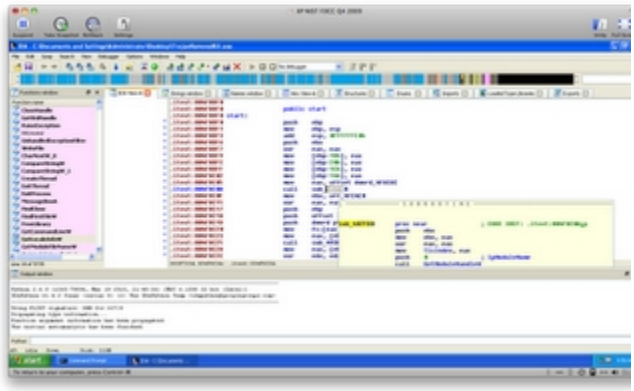
The imports sub view shows us all of the functions or APIs that the binary is using. In our example, it looks like all of the imports are coming from the standard Microsoft libraries.

If we navigate to the Type Library sub view, we will see what IDA thinks was the compiler used to create the sample. In our case, it says MS SDK(Windows XP). This let's us know the binary file was created with the standard Microsoft SDK platform which is used to compile applications for operating systems such as Windows XP, Windows Server 2003 etc.

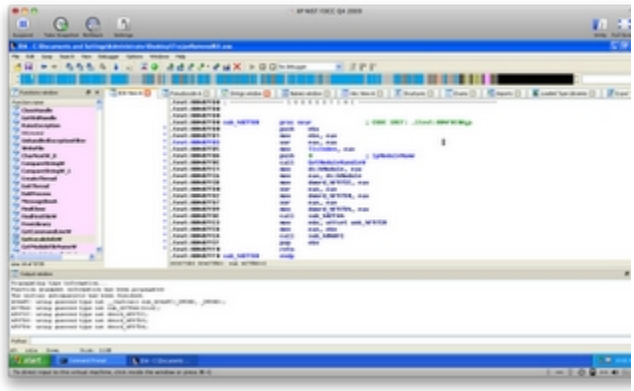


Now on to do job at hand. We will navigate to the IDA View and start to figure out what this thing is doing. I'm only going to show a few pieces in this paper.

The start function appears to be setting up the stack with a number of variables. It then calls a sub routine at 004F8C0A. The call is to sub_407FB0. If you hover over the sub routine, you will see a small information box appear which will show you the details of the function:

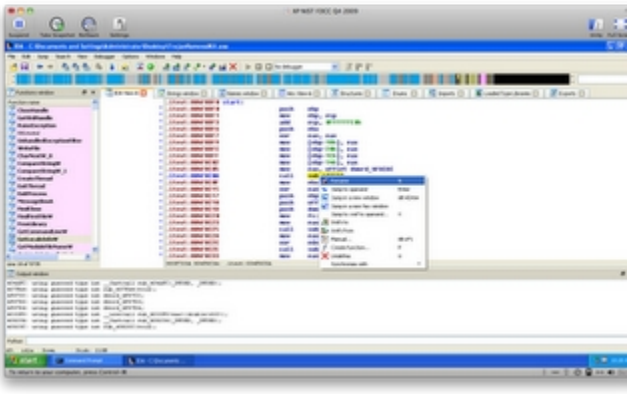


Alternately you can navigate to the sub routine by double clicking the value. Which we will do here.

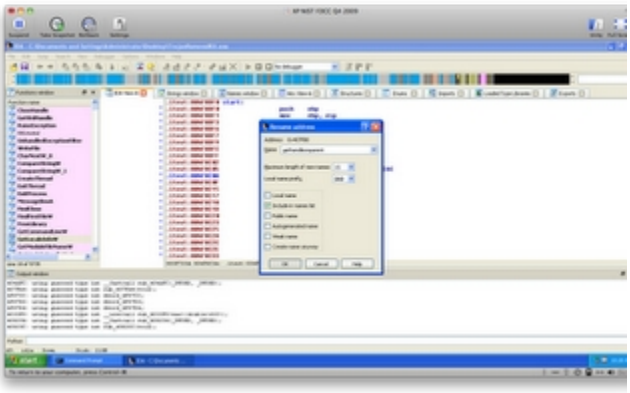


This sub routine appears to get a handle on an already loaded module. This module needs to be loaded before this call. The lpModuleName should contain the module we are trying to get a handle on. Here it appears to be 0. This appears to tell us that it returns the handle to the file used to create the process per the MSDN documentation on GetModuleHandleW.

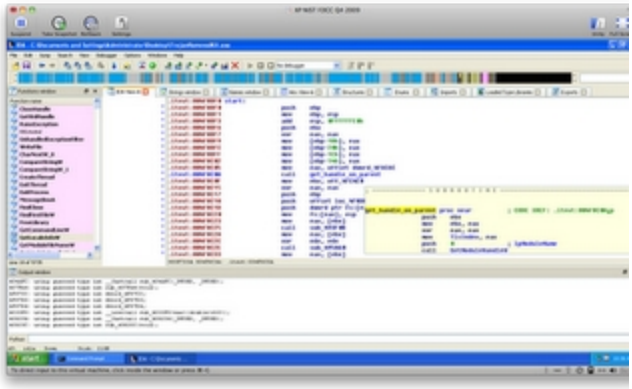
Let's rename this sub routine get_handle_on_parent. We do this by right clicking on the sub routine name and choose rename.



We then rename the routine to what we want.

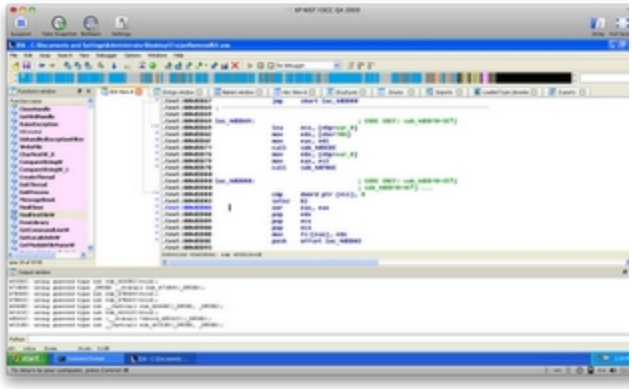


After clicking OK, you may get a warning that the name length exceeds the limit (15). Do you want to increase the limit. Click Yes. You will now see the more meaningful name in the code. Anywhere this sub routine is referenced will be changed automatically for you as well.

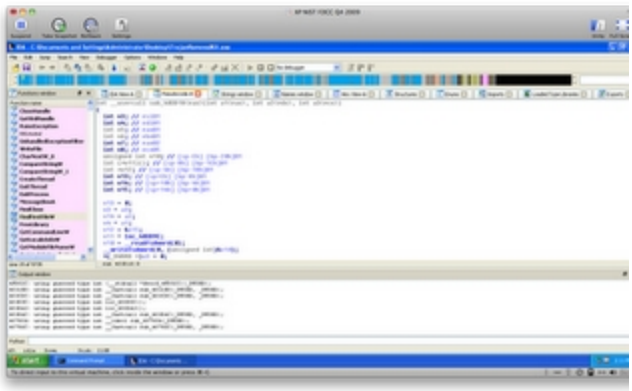


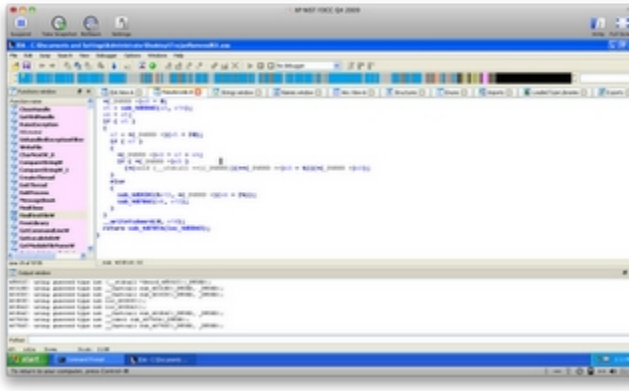
You will now run through the remaining sub routines and name them appropriately. This will help make more sense of the code. You may not get the sense of all of the routines. One word of advise that I would give is to spend a few minutes trying to figure out what it does. If you don't get it, don't sweat it. Move on to another and identify all of the routines you can. Maybe then, others will start to make sense and you will be able to figure them out.

I'm not going to go into detail on this sample. I was really using this as an introduction to IDA Pro. In the next section, I will do the same thing but with OllyDBG. One last thing I will show is how the Hex-Rays decompiler helps make understanding functions a little easier. I have highlighted the sub routine that I want to understand in the following screen shot located at 00403D69:



I will now navigate to the view menu, open sub views and then choose Pseudocode. Alternately you can press F5. This will open a new window called Pseudocode-A for the first window and subsequently Pseudocode-B, Pseudocode-C etc. Below is what that window looks like for this routine:





As you can see, that looks a lot like standard C like programming which you may be more comfortable with than Assembly as I am. The more you clean up your functions and variable names, the easier this will be to read. If you know programming at all though, you can get the jest of it without cleaning much.

Again, you may be eager to learn what this sample does, but alas I am not going to fill that void for you in this paper. I just wanted to show some general options that are available in IDA Pro to help you understand what a binary does via static analysis.

Last Updated: March 15th, 2016

# Upcoming SANS Training

**Click Here for a full list of all Upcoming SANS Events by Location**

| | | | |
|---|---|---|---|
| SANS Secure Singapore 2016 | Singapore, SG | Mar 28, 2016 - Apr 09, 2016 | Live Event |
| SANS Northern Virginia - Reston 2016 | Reston, VAUS | Apr 04, 2016 - Apr 09, 2016 | Live Event |
| SANS Atlanta 2016 | Atlanta, GAUS | Apr 04, 2016 - Apr 09, 2016 | Live Event |
| SANS Secure Europe 2016 | Amsterdam, NL | Apr 04, 2016 - Apr 16, 2016 | Live Event |
| Threat Hunting and Incident Response Summit | New Orleans, LAUS | Apr 12, 2016 - Apr 19, 2016 | Live Event |
| SANS Secure Canberra 2016 | Canberra, AU | Apr 18, 2016 - Apr 23, 2016 | Live Event |
| SANS Pen Test Austin | Austin, TXUS | Apr 18, 2016 - Apr 23, 2016 | Live Event |
| SANS SEC301 London '16 | London, GB | Apr 18, 2016 - Apr 22, 2016 | Live Event |
| ICS Amsterdam 2016 | Amsterdam, NL | Apr 18, 2016 - Apr 23, 2016 | Live Event |
| SANS Copenhagen 2016 | Copenhagen, DK | Apr 25, 2016 - Apr 30, 2016 | Live Event |
| SANS Security West 2016 | San Diego, CAUS | Apr 29, 2016 - May 06, 2016 | Live Event |
| SANS SEC542 Budapest | Budapest, HU | May 02, 2016 - May 07, 2016 | Live Event |
| SANS FOR508 Hamburg in German | Hamburg, DE | May 09, 2016 - May 14, 2016 | Live Event |
| SANS Baltimore Spring 2016 | Baltimore, MDUS | May 09, 2016 - May 14, 2016 | Live Event |
| SANS Houston 2016 | Houston, TXUS | May 09, 2016 - May 14, 2016 | Live Event |
| SANS Prague 2016 | Prague, CZ | May 09, 2016 - May 14, 2016 | Live Event |
| SANS Stockholm 2016 | Stockholm, SE | May 09, 2016 - May 14, 2016 | Live Event |
| SANS Melbourne 2016 | Melbourne, AU | May 16, 2016 - May 21, 2016 | Live Event |
| Beta 2 Cincinnati - ICS456 | Covington, KYUS | May 16, 2016 - May 20, 2016 | Live Event |
| Security Operations Center Summit & Training | Crystal City, VAUS | May 19, 2016 - May 26, 2016 | Live Event |
| SANS SEC401 Luxembourg en francais | Luxembourg, LU | May 30, 2016 - Jun 04, 2016 | Live Event |
| SANSFIRE 2016 | Washington, DCUS | Jun 11, 2016 - Jun 18, 2016 | Live Event |
| NetWars @ NSM Conference '16 | OnlineNO | Mar 16, 2016 - Mar 17, 2016 | Live Event |
| SANS OnDemand | Books & MP3s OnlyUS | Anytime | Self Paced |