What is a Crypter?

Okay before we get into the good stuff, lets first clear up all your questions you have been having by really getting into all the fundamentals of Crypters. Oh and if you have any questions of anything throughout this tutorial, always refer and search on SPAM for answers. If you don't already know, A Crypter is usually used to encrypt files like viruses, rats, and keyloggers usually for the sole purpose of bypassing antivirus detection.

What's the difference between a Crypter and a Packer?

A Crypter Encrypts your files, while a Packer packs your files usually with the intention of making it smaller in size and sometimes for it to be undetectable on virus scans.

What's the difference between a Runtime and Scantime Crypter?

Both can look exactly the same so you better watch out..

-A Runtime Crypter encrypts the specified file and when executed (ran), it is decrypted in memory. This way antiviruses aren’t able to analyse the file before executed and after executed.

-A Scantime Crypter encrypts the specified file so antiviruses aren’t able to analyse the file only before executed but NOT when executed.

How do i know which antiviruses detect my file?

There are many sites with this same purpose of scanning files and giving a report of which antiviruses detect your files. The main issue leading to Crypters becoming detected is because if you or someone who is in posession of your crypted file, scans it on some of these scanner sites, the crypted file will be distributed to the antivirus vendors, thus causing the crypted code overwritten on your file to become detected, which in turn causes your Crypter to turn out detected. I recommend that you scan your files on http://www.novirusthanks.org - But MAKE SURE the "Do not distribute sample" check box is checked

What is EOF and what is it used for?

EOF stands for End Of File. Some files like Bifrost, Medusa, and Cybergate require the end of file data in order to run without corruption. So If Crypters don’t preserve this end of file data, your crypted file will become corrupt.

What is a USG?

A USG is part of a crypter that generates a unique version of the stub (stub is part of a

**crypter used to encrypt and decrypt the specified file). The purpose of this is because FUD crypters don’t last forever, eventually crypters become detected over a period of time. You will understand
this better later on in the tutorial.**

What is a File Binder?

A File Binder is pretty self explanatory. It “binds” or puts 2 files together as one so as a result when someone opens this one file, 2 files will execute. You would usually use a file binder when being even more stealth then just simply a crypted file. The biggest question people have when first learning what a binder is and what it does is, can you bind a .exe with something different? like a .jpg for example? The answer is Yes, BUT.. the output of both binded files will be shown as .exe, so in a way it can defeat the purpose.

What are "Antis" on Crypters?

Anti’s are an extra feature that come with some Crypters. For example anti-vm, anti-debugger, anti-avira...etc. These refer to bypassing or preventing something specified, so anti-debugger meaning it will prevent it from being debugged.

What is a File Pumper?

A File Pumper will “pump” your file - referring to adding more bytes to it making your file larger. The benefit of this is usually not so great but it can be okay to have and may lose a detection or 2.

Types and Forms of Crypters

Crypters can range in many types and forms and it is important to understand these types and forms because it will help you choose a quality crypter to solve your needs or help you realize what options and features you would want to implement in your own Crypter.

The Most IMPORTANT Factors You Should Know

As I'm sure many of you know, finding Crypters and Crypters themselves can be a huge pain. I know when i first started out, I hated the fact that i just couldn’t find a FREE FUD CRYPTER anywhere. I got so pissed, but didn’t give up just yet. I kept on searching and reading a diverse range of forums. Overtime, once I learned enough about them i realized the actual undetection vs antivirus concept. This is the eye opener point which you will all eventually end up and at this point you will then realize why.

The Antivirus vs Crypter Concept

Have you ever wondered how all the virus’s, rats, and bots..etc become detected by

antiviruses? I'm sure you have, and this concept will give you all the answers. Antiviruses can be alot more complex then you would imagine, so learning the ways they are notified of malicious files and how they detect are essential for bypassing them. Okay there are 2 ways antiviruses are notified of malicious files and eventually flag your file as detected.

1.The First One is:
From online file scanner sites where people upload files they think might be suspicious looking, and want to know if its actually a virus or not. They upload their files to one of these sites to check which antiviruses detect it and flag it as a virus. Once the files are uploaded, based on certain elements they are then distributed to the antivirus vendors labs. On some online scanners there is an option available for you to check for no distribution. I am not aware if this actually does what we all think because i heard they will still distribute, but with a price to the av vendors. Even though this may be true or false, it is still always a good idea to scan on these sites that have this option available.

2.The Second One is:
From the antiviruses themselves. You may be thinking, oh really? Yes.. and to tell you the truth, hardly anyone even knows about this. It's sad isn't it? This is essential information that everyone must know when using or making Crypters. Most of the time, the antivirus will automatically send the files out when any certain file becomes detected. Antivirus owners also have the option to send off a file to the vendor with a click of a button through their desktop antivirus.

What can you do about this?

You can change the settings on your antivirus! The setting usually come in slightly different forms, sometimes you are also asked during setup, and sometimes you just have to go into the settings or options manually to change them. All of what you just read is essential to keep in mind when making an FUD Crypter. The sole reason behind why public Crypters always become detected, and usually fast is because the majority of people do not know the antivirus vs Crypter concept. Therefore they either blindly upload there crypted files to one of the scanner sites that distribute, Also, the antiviruses themselves are uploading there crypted files without them even noticing. Even people who make there own Crypters arent aware of this, which is why they are always wondering why there crypted files always become detected so fast.

What do AntiViruses look for in a file?

First off, you will need some basic understanding of how anti-viruses actually work. .Exe files are simply lines of instruction, and each line is called an offset. Anti-virus's have databases of these lines that are known to be associated with malicious files. They use that database to check against your file to see if it matches. If it does, then it is marked as infected. They do use other methods of detection, but this is the one you will learn how to avoid.

What will the program need to do?

Your Crypter is going to take the contents of an infected file, encrypt them,and place it at the bottom of a seemingly virus-free file called your “stub”. Your stub file will then extract the encrypted data from itself, decrypt it, then extract and run it. So just imagine if this stub file that is joined together with the crypted infected file is detected? Well, then all the files you crypt will also show up as detected since this stub is used with all the crypted files. This may sound like a complicated and confusing process, but it isn’t and I will explain more about it later on.

Programming and Vb6 Fundamentals

Okay now.. your either one of 3 people.
- **someone who has no idea how to make/code a program**
- **someone who knows and can code a program**
- **someone who can code but not in visual basic 6**

This section is intended for all these people. If you can code and you think you wont benefit from it, you can either just scim through it or just read it all and refresh your memory. First we must download Visual Basic 6 of course. If you aren’t aware of what torrents are and how to download them, then follow this:
step 1 if you do know about torrents and already have torrent downloading software go to step 2
1. download and install u torrent here **http://www.utorrent.com/**

2. now we must download the vb6 torrent here **http://btjunkie.org** **search?q=visual+basic+6 (just download the first one or something)**

3. it should then open with utorrent, and just press ok to download.
now to get a quick picture of the interface in vb6 and understand how most of it works, (you can just scroll down to the pictures on this site) **http://www.profsr.com/vb/vbless01.htm**

Intro to programming with vb6

You must be aware that in order to make your own FUD Crypter, you must at least know the basics of programming. So if you don’t, this is a very important section to read! So read through it all and if there's something you don’t understand I encourage you to do some google searching about it or read around/ask through this forum (SPAM). Without getting so in depth and complicated, I am going to first have you learn the basic concepts of programming in order for you to just understand enough to be able to first understand the most essential parts of what a program is doing so you will be able to understand other sources when you read them and modify them. You most likely will have questions that I will not be able to answer, so if you're unsure about some of these basic concepts, search vb6 tutorial or visual basic tutorials on **http://youtube.com**. **This way always seems to be best because it seems like people learn alot easier over video tutorials rather than text tutorials. If you have a more specific question search google. Okay so, from searching for a long time. I came to the conclusion that this site teaches vb6 in the best most understandable/appealing way in my opinion.** **http://www.vbtutor.net/vbtutor.html**. **Please try and go up until lesson 18 and**

ignore all the ads on the sides and in between. You obviously don't have to do this all in one day just take your time and make sure you firmly grasp what your learning.

Basic Vb6 Outline for Creating a Crypter

Crypters in Vb6 consist of two parts:

- **The Crypter Client which is the actual user interface that the user uses for specifying the file to encrypt, the settings...etc**

- **The Stub file, which is part of the Crypter but it is not used by the user, it is simply just there, in the same directory as the Crypter client, because it is being used by it.**

So programming a Crypter comes in these 2 parts and are made seperately in 2 different projects. They only interact with each other when compiled into finished .exe’s. You might be wondering, well what project gets detected so I will know which to modify? The Stub project is only what you have to always undetect and, re-undetect. The crypted files become detected BECAUSE the stub file is what is actually injected into all the crypted output files. So common sense being, when eventually, for example someone that you infected runs the crypted file and maybe uploads it to virustotal (which distributes) or the antivirus itself distributes, the crypted file has your stub code in it aswell as the crypted malicious code. Therefore the antivirus will then detect and put signatures causing the stub code to become detected. Basically this stub code is injected into all crypted files so obviously all the crypted files will then also become detected since it caries these detected signatures.

vb6 and undetection - what to do and what not to do

When Making a Crypter, first always keep in mind to have the project placed directly in the C:/ location on your drive because if it is for example in your documents folder like (“C:/user/john/crypter/stub”) this whole string of text will be shown and easily read by antiviruses and cause your crypted files to become detected or provide an easy target for antiviruses to develop a signiture for. Now this is only one factor to keep in mind but it is definitely something you should know. Okay now that you have your whole Crypter/stub projects on your C:/ drive, Lets open up the project and do some of the main essential tasks for preventing antiviruses from detecting these sources.

Changing Assembly information

First we are going to change the compilation settings for the .exe, like the file version, description, etc These files settings are one of the first things antiviruses check and is something you should always do when picking up and modifying new sources without even thinking about it. Just make this a habit. Open the Stub Project and Right click in the project space on the top right and click project Properties. Once your there, you should see few options like project name, startup object, if you want to change any of that then do it. So now go into the next tab called “Make”. Here you should see the version info, title of application, icon, and in the middle you will see “version Information” with comments, version, company name, file description, etc. All these

options should be changed to anything random. Especially when starting from someone else’s source.

The Antivirus Signatures concept

Whats going to be explained here, you should always keep in mind when undetecting. Read every bit of this section, some things you may know already but there are definitely things you do not know which are very important. To my experience there are 2 types of signatures, which i like to call:
Specific Signatures
Broad Signatures
Throughout making FUD Crypters you will come to realize that overtime all Crypters, private or public, will eventually become detected. Now the reason for this is because not only do the people you spread the crypted files to have antiviruses that automatically distribute, etc. But also, antiviruses in cases where they get alot of similar files distributed, try to create signatures for the most unique parts of the code that all these malicious files have in common. Now what I mean by that is for example, Avira antivirus will detect a certain set of api’s that’s being used in a certain variation of ways, corresponding to, and interacting with other certain parts of code. This is a broad type of signature. Unlike specific signatures that just detect a certain string of text in a certain part of the code, this broad signature will then cause all the Crypters using this api related to this situation to become DETECTED. This is the very disadvantage of programming in the most popular languages where Crypters are most popular to program with. So now if you think about it, a stub can also only go so far in being unique because antiviruses are always updating and populated their databases with not only specific signatures but, these broad signatures which eventually overtime will cause your Crypter to become detected. No matter how unique your stub is, a part of this code in relation to broad signatures will become detected. Even if you do nothing with it. Now it may be more unlikely depending on how unique, but the point is that. Even if your doing nothing with your stub and never crypt files, eventually it will become detected, all will.

So to clarify, the fact that from all the other Crypters being distributed that for example that use a specific method of execution using a specific api which has slight relation to how your Crypter was made, will cause your crypter to also become detected. Now with all this in mind, i want to make sure your not getting the impression that all vb6 Crypters suck and they will all get detected easily, because this is not completely true. As long as you use the right techniques and have your own unique and creative way of doing things, the longer the Crypter will last. And just to let you know, when a crypted file is distributed, its not like it will become detected right away. It takes about a week to a few weeks for a signature to made on the file and updated into the database. So a point i want to also get across while you understand this concept is that, The most honest true approach you will learn in this tutorial, is the fact that no matter what undetection technique or method you use, there is no one technique that will last forever, they all eventually become detected, which means that there’s no guarantee for giving you a technique to easily just copy and paste to make your fud crypter and live happily ever after, that would be a lie. What this tutorial will give you, is a layout of the universal, proven techniques that you can keep in mind so you can learn how they work, improve upon them, and make variations of them to successfully make your own FUD Crypters.

Finding and pinpointing Whats causing detection

To accomplish the process of finding and pinpointing detection it is required that you understand the different parts of code and know what most of it does because you will be literally taking apart the code when finding the cause of detection. I find that alot of people try undetecting there sources blindly by just throwing a whole series of undetection methods at the code. This is fine if your first starting out from scratch on a fully Detected source, but when there are only a few antivirus's detecting the source, you must start finding exactly whats causing detection. This will save you tons of heart ache and make the whole undetection process a whole lot easier.

Aright This is where all the learning happens, you will realize and learn alot of how what code certain antivirus's will detect. You will then be able to easily mitigate certain antivirus's and find that some antivirus's are easier then others to undetect from. You will then not only have a good set of knowledge from experience of finding what causes detections for certain av's, but you will also easily gain a set of skills and new and improved techniques that build upon the previous ones for undetecting against certain av's. Finding whats causing detection can be very easy or somewhat difficult depending on if its a broad signature or a specific signature. The majority of the time they will be specific signatures. I will be giving you an example of both. Basically you will be pulling apart your code deleting them one by one, then drilling down further deleting more specific, smaller bits of code until you end up at whats exactly causing detection. When going through this process it can seem like its time consuming but its actually not if you have the specific antivirus which is detecting your file downloaded and installed on your system so you can instantly scan each compiled stub with certain bits taken out. A specific signature detection in this example will be a small string in the RunPE module, so we will be taking apart the code and to do this, there are a few steps involved

1.The first logical step to take would be to first check if the detection is caused somewhere in the first start of the program execution so delete everything in the sub main() ..then compile the stub and scan it. no detection found so we move on and put the code back.

2. Delete all the code in each module one after another until the detection doesn't come up anymore. We then find out once we take out the runpe module the detection goes away.

3. Now that we know the detection is coming from the RunPE module, we will put the code back and drill down by first deleting each sub and function in the module. We then find out the detection is coming from the CallAPIbyname function.

4. Now that we know which function is detected, we will then drill down further by deleting each line of code. (depending on the size of the func, just delete each segment and drill down from there, you can do the same for the modules, for example you can delete the first half and second half of the function first)

5. Then once you found the string in the function that's causing detection the whole undetection process comes into play. You can basically just recode the portion of code that's causing detection in a very different or even slightly different way

and combine this what you will learn in the next section, or simply only use whats in the next section alone.

Broad Signatures

For detecting broad signatures, its pretty much the same process, The only difference is that you have to be aware of a few more things throughout the process. I will show you some examples of a broad signature in this situation, Lets say the "RtlMoveMemory" api is causing detection. Now if we are taking apart the code using the process i just showed, you will realize that the detection is coming from the module but you wouldn't realize what is being detected inside the module by doing the standard, remove each sub/func at a time. The reason for this is that this api is used in multiple places throughout the module. Sometimes you will even come across situations where variations of the same piece of code is used throughout the module.

The Universal Undetection Process

Like I mentioned earlier, The most honest true approach you will learn in this ttorial, is the fact that no matter what undetection technique or method you use, there is no one technique that will last forever, which means that there's no guarantee for giving you a technique to easily just copy and paste to make your fud crypter and live happily ever after, that would be a lie. What this tutorial will give you, is a layout of the universal, proven techniques that you can keep in mind so you can learn how they work, improve upon them, and make variations of them to successfully make your own FUD Crypters.

Okay so basically lets say you found a certain specific or broad portion of your code that's causing detection, THIS is when the whole undetection process comes into play.So you have some options at this point depending on if you're a beginner or experienced programmer. (More programming knowledge and how crypters are made will give you a huge advantage when undetecting code) So you can either just recode the portion of code that's causing detection in a very different or even slightly different way and combine this with the examples I am about to show you or you can only use the examples alone, using your own variations, of course, and extensive amounts of them. Sometimes though, you will eventually realize that no matter what undetection techniques and how much you use them based, you have to actually end up recoding, or using a different variation of that same code which do the same overall task, And this is very simple for someone who has well rounded programming knowledge so this is why I say, you will have a big advantage if you do too.

Here's the basic outline of the whole universal undetection process summed up in the most brief way

- **Adding junk code for modifying execution flow and various other reasons**
- **Changing the order of all code aspects.**
- **Changing variable names**
- **String manipulation.**

● Change Assembly information
● Add or change icon

Just remember, no one way will last forever. So be creative and try new things to distract and confuse av's.

About Unique Stubs and USG’s

Okay so you know how a USG comes with some Crypters right? Well these USG’s also known as stub generators, generate unique versions of the stub for that Crypter. How all of these USG’s generate unique stubs are from using all these methods of undetection but in the click of a button. How?
A set of techniques and methods are implemented into the USG using variations of the same undetection method/techinque by randomizing the strings, variables, and the order within these undetection techniques (like variations of junk code). Also giving the user the ability to choose specific undetection options/methods to use thus creating a “unique” version of the stub, This way, when someone’s stub becomes detected there is a high chance another persons stub, using the same Crypter, won’t get detected. Since the majority of the stub might have a different variation and layout of the code from all the undetection options/methods used in the usg, there is a high chance the signiture that causes the other stub to be detected will not be shown, or in the same place in this other unique version of it because it might be 90% different. So basically USG’s ultimately give an advantage for how long the stub will last undetected. If you don't fully understand this, its fine because you will better understand it once you actually start learning and applying these actual methods and techniques. Lets start with,

Adding junk code

Okay here’s pretty much all the types of junk code:
● junk subs/functions
● fake calls
● fake variables
● junk strings of text
● fake loops
● fake if/else

Basically all junk code is, is randomized portions of regular code which you spread across your program that can either just be in between and/or throughout your programs code, it can deceive or confuse execution but never actually interferes too much with the process of execution to the point where it will corrupt.

Changing variable names

Changing variable names is highly important and must be done. Press ctrl + H and you will see a small replace form popup. It is very important that you don’t mess up the code, so always make sure you use the right options when changing a certain variable or set of variables in your code. For example
you could be changing a public variable which is used throughout your whole project and without noticing, only selecting the “current module” option, causing only the variables to be changed in that current module. So always keep these things in mind

when changing variables.

String Manipluation

Just like changing variables, changing strings can mess up your code if you aren’t too cautious.. Especially when encrypting strings and api’s. Encrypting strings and api’s are very powerful and is a must when it comes to successfully creating a fully undetectable Crypter. Some examples of string manipulation:

- **Encrypt Strings**
- **Reverse Strings**
- **String conversion**

There are many types of encryption algorithms to encrypt strings with for example the most popular are xor, rc4, Rot, string to hex. A big issue most people aren't aware of is the fact that sometimes when encrypting strings with some RunPE modules.. bad things happen, files become corrupt, the Crypter itself can become corrupt, etc. So always be cautious of your string manipulation. There are some important strings to always make sure are changed or encrypted in your Crypter. The first to take note of is, The Key Split which is, in the example below:

**meEncPass = “thepassword”. Change the string to something like: “aksefialUEHF@q#)*!qJFIAUEHFIwqNEOGq)#”
and remember, this string has to be the same key split in both the stub project and the crypter project or the crypter will not work and give you a “subscript 9 out of range” error when running the crypted file. The second to take note of is, all the strings in the RunPE module. 99% of the time these have to be encrypted no matter which runPE module you use. So always remember to encrypt these.**

Add or change icon

Adding or changing an icon isn’t too good of an undetection technique but it can undetect from 1 or 2 av’s in some situations. Also changing an icon can corrupt files aswell but its actually pretty rare. The reason this happens is most likely because the icon size is different then the size the file can handle. This is very easy to do. Google it for step by step. I don't mean to insult anybody's intelligence but I feel like this is common sense.

Tools and automation

The beauty of undetection is that there are tools which automate the universal undetection process using variations of techniques and methods. These tools are usually referred to as, Undetectors, or src undector, etc. If you think about it, undetectors are very similar to USG’s. The only difference between a USG and an undetector is that a USG, choosing and setting certain paramters, will undetect and create a unique stub based on the scrambling/randomizing set of undetection techniques in only a click of a button. The USG’s comes with a certain Crypter which it will only create unique stubs for. On the other hand, a undetector is for undetecting actual source codes using a set of

options and techniques. These options and techniques tend to be alot more flexible compared to USG's.

What you will learn from Undetecting Crypters

This is highly likely to be the most important sections in this tutorial, so I advise you to read it all. There is no doubt you will learn alot from making and undetecting Crypters. Everything you learn will help you along the way making things alot easier. If some things discourage you right now, do NOT let it. Just take immediate action and start making your own right now. Use all of what i taught you and constantly improve and add to everything. Combine methods and create your own. Once you gain momentum and practice, the experience will pay off big time. Things will become alot easier and you will have your own fully undetectable Crypters in no time. Okay now i want to let you in on some important things to keep in mind which will help you benefit from the practice and experience you will gain. Antivirses are alot more complex then you think, so the main focus from undetecting your crypters should be to always, on a consistent basis, learn and understand more and more of how antiviruses detect and what they detect. With this focus, you will have many realizations and insights for easily creating new and improved techniques for bypassing certain av's. The more you learn about the antiviruses, the easier it will be for you to undetect parts of your code that are detected by that certain antivirus and creating new and improved techniques that will not only just bypass the detection but also keep the code undetected for long periods of time from that av by creatively coming up with unique techniques that deceive or distract antiviruses in ways that are harder for that av to detect it. Another piece of knowledge you will come to realize is the kinds of things certain av's will detect and the usual kind of techniques that will bypass those certain av's .Here's a simple example alot of people can relate to, All the people that have successfully created FUD Crypters i think, know that the av, Avira antivir usually will detect api's in your code, and certain techniques that will bypass this is for example, adding the callapibyname function to your code and calling this function everytime you use the detected api. What you should know though is that techniques that may bypass a certain av will not ALWAYS bypass it in a month from now. Antivirus's are constantly being updated, But just because they are constantly being updated, don't let that discourage you because usually a simple tweek to the method that worked a month ago can work now and easily bypass the new detection, This is why you must always be creative and learn from how the certain av works, what they detect, and how they detect because it will help you GREATLY.