Project in Ransomware “MindLost”

Daniel Ohayon
danielohayon444@gmail.com

1 Table Of Content

Project in Ransomware "MindLost" 1

1 Table Of Content 2
2 Abstract 4
3 Introduction 5
3.1 Background 5
3.1 motivation 6
4 Theoretical Background and Algorithms 7
4.1 AES 128 bit 7
4.2 Obfuscation 8
4.3 Executable Compression 9
4.4 Parallel Computing 10
5 High Level construction Of MindLost 11
6 Encryptor 12
6.1 Step1: Hiding The Console 12
6.2 Step2: Writing to the registry 12
6.3 Step3: Check The Environment Is Not A Virtual Machine 12
6.4 Step4: Smart Interval Wait 12
6.4 Step4: Checking for insurance 12
6.5 Step5: Creating AES Encryption Key 12
6.7 Step 7: File Encryption 13
6.8 Step 8: Uploading Encryption Key 14
6.9 Step 9: Deleting The Original Files 15
6.10 Step 10: Displaying The Hidden Encrypted Files 15
6.11 Step 11: Notifying The User And Giving Instructions 15
6.12 Step 12: Obfuscate And Compress 15
7 Decryptor 16
8 Command and Control Server 16
9 Payment Website - http://mindlost.azurewebsites.net 17
9.1 Website's Backend Structure 17
9.2 Website's main page and logic 18
9.3 About page 19
10 Testing MindLost Against Anti Viruses 20

10.1 MindLost Vs AVG's Active Realtime defender 20
10.2 Virus total 22
10.3 Metascan 22
10.4 VirSCAN 22
11 Compiling Instructions _____23
11.1 Installation 23
11.2 Workflow 25
11.3 Website 25
12 Summary _____26
13 Bibliography _____27

2 Abstract

In this paper we will discuss the background of ransomware malware and the process of creating a ransomware. We will also discuss various methods I used to avoid the detection anti virus scanners such as code obfuscation, code compression, anti debugging methods, detects if it's running on a virtual machine and if so aborts, being 100% transparent to the user as well, low CPU usage to avoid suspicion but uses parallel computation to achieve an average rate of 100(Mb/s). Automatically writes it's self to the registry so it can withstand a shutdown during the encryption process and when the computer reboots it will keep encrypting while still being completely transparent to the user. In addition the ransomware communicates with a command and control server which stores the keys of all of the victims in an encrypted manner. And a fully functional payment website for the victims.

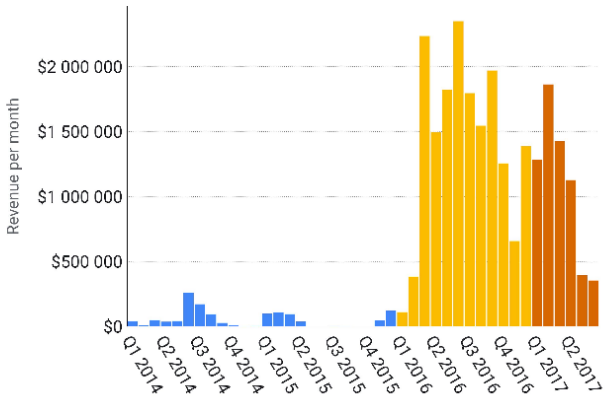
3 Introduction

3.1 Background

Computers have evolved during the past years to become an integral part of out life. We trust them to keep our most important information safe like our beloved family photos, our important work documents and records and the projects we are working on. But what happens if you suddenly take from someone the capability to use their valuable information? How much would they pay to get it back?

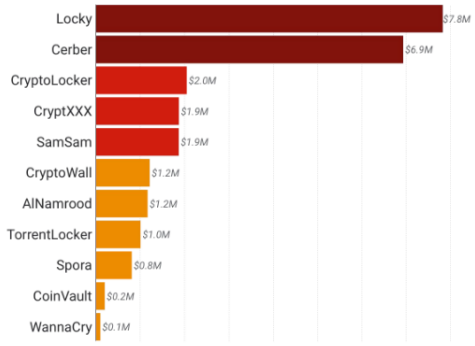
Turns out that a lot of money. In recent years a phenomenon of a new type of cyber attack has exponentially risen called Ransomware.
Ransomware is a type of malicious software that encrypts the victim's data and by doing so blocks access to it unless a ransom is paid.

This new business is extremely profitable since as we mentioned before people are wiling to pay up to 600$ to get their information back.



In 2016 ransomware became a multi-million $ business

And so in 2016 Ransomware became a multi-million business. With a lot of players entering the market with various forms of ransomware. Some of the most successful ones have been Locky and Cerber as you can see below:



The ecosystem is dominated by a few kingpins

3.1 motivation

Since many describe the cyber security world as a game of cat and mouse I set out to make the next mouse for the cats to chase and maybe by doing so help the Anti Virus companies learn how to defend from it. Meaning I set out to create another sophisticated type of ransomware with the purpose of giving it to Anti Virus companies to see if they can detect it and if not then it would help them create new techniques of defending from a ransomware.
And of course Im also very interested in this so I wanted to learn more about this world by experimenting with building a ransomware of my own.

4 Theoretical Background and Algorithms

In this part we will discuss all of the elements that are implemented into this project.

4.1 AES 128 bit

in this ransomware I use the AES 128 bit encryption. AES is based on a design principle known as a substitution-permutation network, a combination of both substitution and permutation, and is vert fast. I leaned about this type of encryption in a course called Computer security (Network Security) so when I had to decide what kind of encryption I should use in the project I immediately thought of AES since it is fast and extremely hard to break.

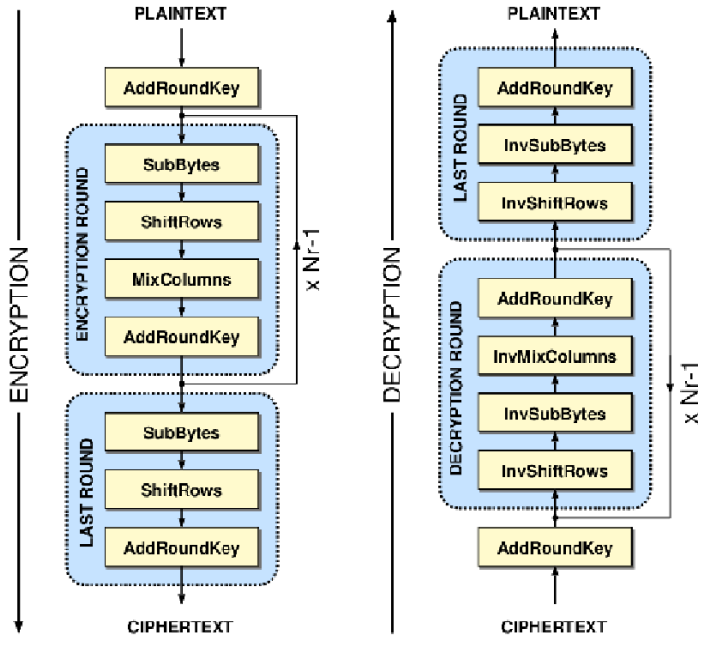
A high level description of this Encryption is as follows:

1 KeyExpansions—round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.

2 InitialRound
   1 AddRoundKey—each byte of the state is combined with a block of the round key using bitwise xor.

3 Rounds
   1 SubBytes—a non-linear substitution step where each byte is replaced with another according to a lookup table.
   2 ShiftRows—a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
   3 MixColumns—a mixing operation which operates on the columns of the state, combining the four bytes in each column.
   4 AddRoundKey

4 Final Round (no MixColumns)
   1 SubBytes
   2 ShiftRows
   3 AddRoundKey.

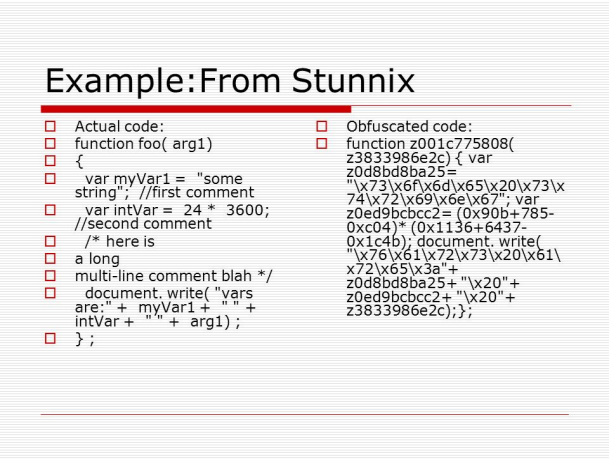


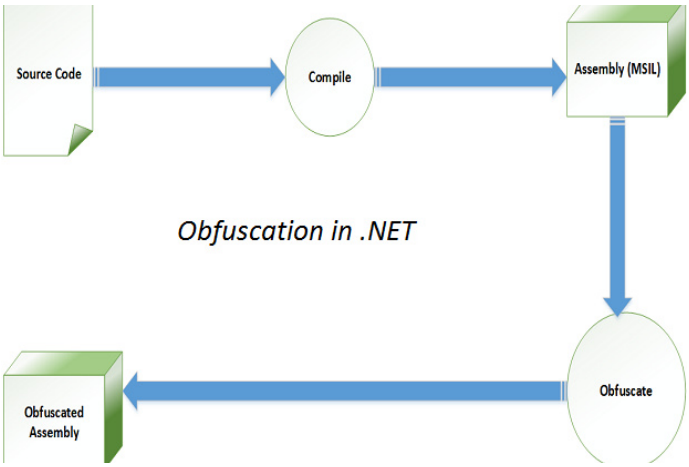
4.2 Obfuscation

In software development, obfuscation is the deliberate act of creating source or machine code that is difficult for humans to understand. deliberately obfuscate code to conceal its purpose (security through obscurity) or its logic or implicit values embedded in it, primarily, in order to prevent tampering, deter reverse engineering.
In addition another advantage of obfuscated code it that in .net a decompiler can reverse-engineer source code from an executable or library easily and if the code is not obfuscated it will be very easy to understand what is the purpose of the code (which we want to conceal).

Obfuscating my code includes:

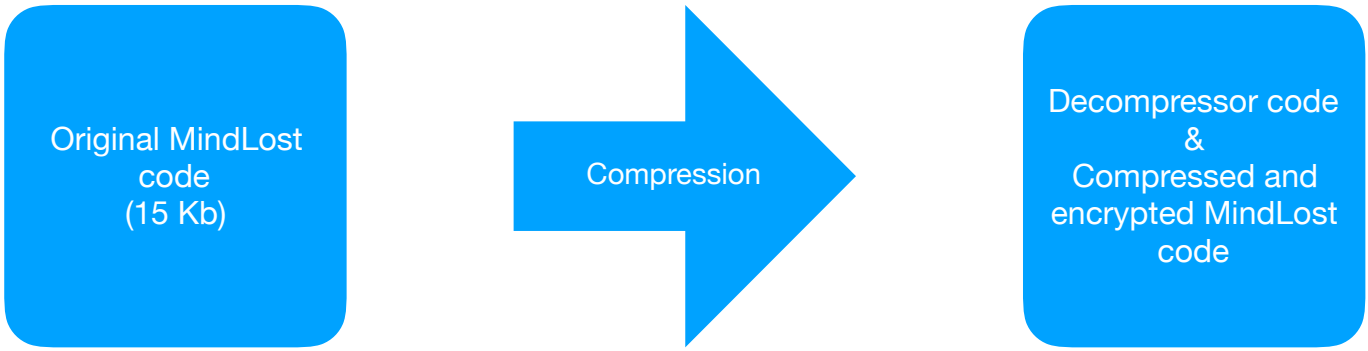
1) string encryption - substituting the strings in my code with calls of special encryption functions, which render my strings perfectly at runtime, leaving them unreadable to someone else.
2) Code control flow obfuscation - code gets replaced with functionally equivalent, but different instructions. Decompilers often crash on such code. “spaghetti code”.
3) inlining functions - We as humans always look for patterns like function which make it easy to understand the purpose of every chunk of code. But with code obfuscation we remove the structure of functions by having the entire function embedded in the code each time we call it making the code just a one very long function that is extremely hard to understand.
4) Symbols renaming - When we develop a program, we often present the most valuable information in form of symbol names: names of functions, variables, classes. so in obfuscation we will just make the names of all the symbols unreadable.
5) indirect addressing - indirectly addressing cells in the virtual memory.





4.3 Executable Compression

Executable compression is any means of compressing an executable file and combining the compressed data with decompression code into a single executable. When this compressed executable is executed, the decompression code decompresses the original code from the compressed part before executing it. In most cases this happens transparently so the compressed executable can be used in exactly the same way as the original.
Executable compression is also frequently used to deter reverse engineering or to obfuscate the contents of the executable (for example, to hide the presence of ransomware in our case from antivirus scanners) by proprietary methods of compression and/or added encryption. Executable compression can be used to prevent direct disassembly, mask string literals and modify signatures. Although this does not eliminate the chance of reverse engineering, it can make the process very hard. By compressing the code and thus making the code available only during run time when the decompressed part decompresses the compressed part an anti virus program can't read or search for code signatures statically.
Another method an anti virus can use is look for a dynamic code signature but that requires the anti virus to run the suspicious code in a virtual environment. But our ransomware can detect it's running in a virtual machine and shuts down to avoid suspension.
Or the anti virus can decompress the code to look for a code signature but to do so it has to know the method it was compressed with which if you use a costume compression is not possible for it. Plus the anti virus cant just generate a code signature for the malware from the uncompressed code because there are a lot of legitimate programs that use compression and that would lead to a lot of false positives.

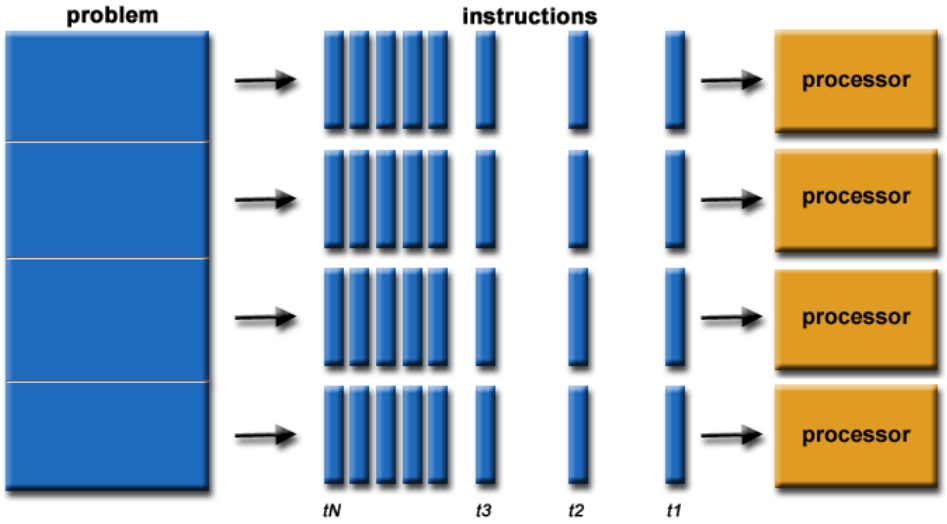


Original MindLost.exe (15 KB)

compressed MindLost.exe (5 KB)

4.4 Parallel Computing

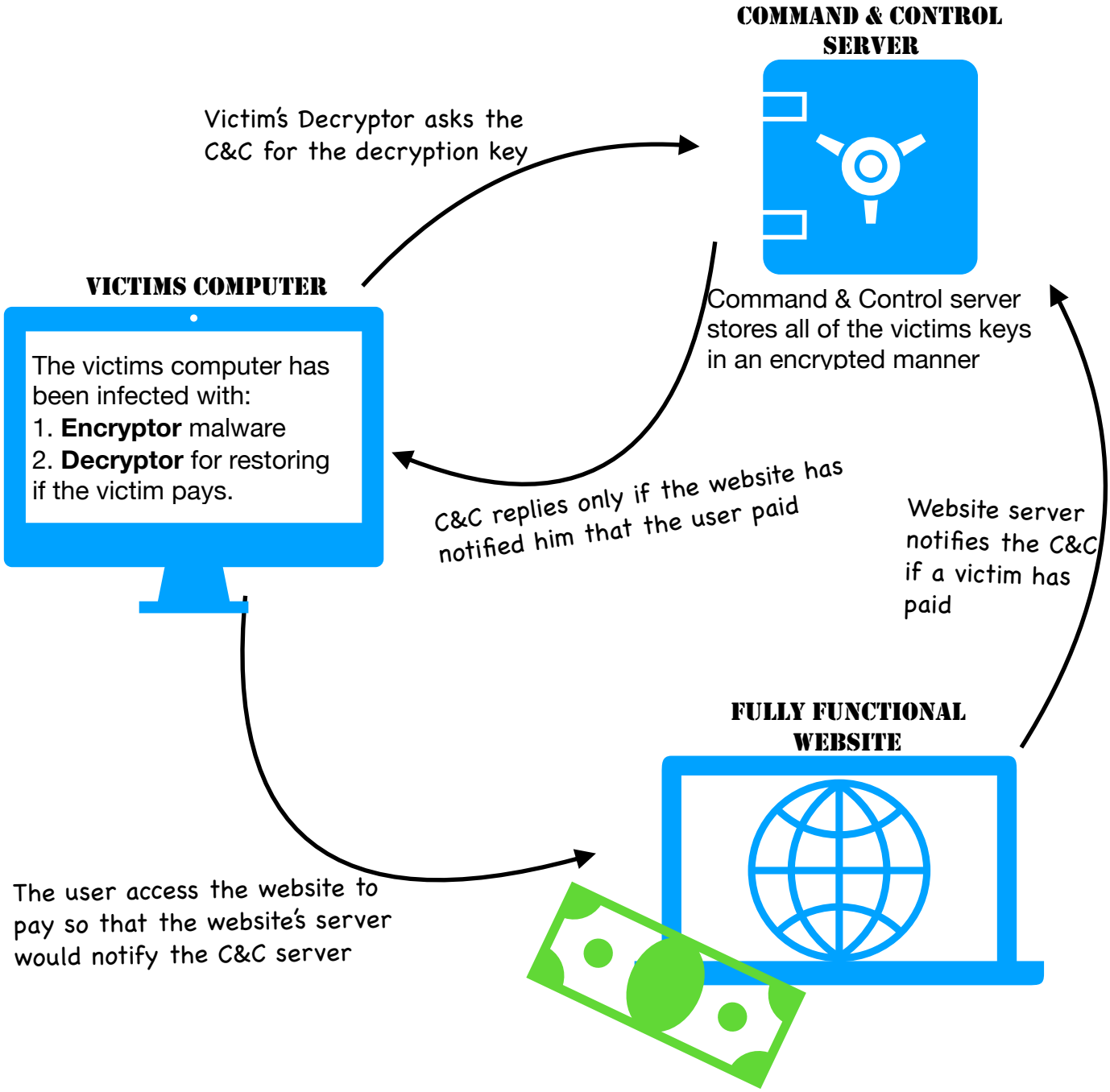
Parallel computing is a type of computation in which many calculations or the execution of processes are carried out simultaneously.
Modern computers usually have more then one core. So in order to optimise the encryption process of the files of our victim we take advantage of all of the victim's cores and use them all to encrypt his files in a parallel manner.



5 High Level construction Of MindLost

our ransomware is called “MindLost” because the first line in the message the user sees after his computer has been encrypted is *“Don’t loose your mind but all of your files have been encrypted”.*

We will now give a high level description of whole MindLost operation before going into details abut each one in the next section.



Fully functional website currently running at http://mindlost.azurewebsites.net
That allows the victims to pay and if a victim pays it notifies the C&C that he paid.

6 Encryptor

This is the payload which can be masked as a jpg or called from a word macro. Once it has been executed it does the following:

6.1 Step1: Hiding The Console

The first thing we do is make sure to stay hidden from the user as well by not opening a consul.

6.2 Step2: Writing to the registry

Writes it's self to the registry.

Advantage:

Incase the computer will shut down before the encryption process is complete.

6.3 Step3: Check The Environment Is Not A Virtual Machine

Checks if its currently running in a virtual environment and if it detects it is running in a virtual environment it aborts to avoid detection.

Advantage:

This is done in the case that an anti virus tries to run MindLost in a virtual environment before letting the user run it to see if it does damage to the virtual environment which doesn't store anything important. Then the anti virus would report to the user that MindLost is harmless and when the user would run it in his actual computer he would get infected.

6.4 Step4: Smart Interval Wait

Waits for a specific number of minutes before it runs. The number can be changed by changing the constant called MINUTES_TO_WAIT. It does so by waiting for 5 seconds and doing so (MINUTES_TO_WAIT/12) times.

Advantage:

This is done in the case that an anti virus tries to run MindLost in a virtual environment to see if it does damage before allowing the user to run it. And if the anti virus has managed to fool MindLost into thinking it's not running in a virtual environment.

6.4 Step4: Checking for insurance

Checks the command and control server to see if the victim has purchased insurance. And if so MindLost won't encrypt his computer again.

Advantage:

An important part of any successful ransomware is being trust worthy. Because only if it is known to be trust worthy then victims will trust that they will get their files back if they decide to pay. And for an additional sum of money the victim can purchase insurance to make sure his computer won't be encrypted again by MindLost.

6.5 Step5: Creating AES Encryption Key

Creates an AES Encryption key. Then it will iterate over all of the user's files and add to a list the paths of the files that can be encrypted which includes all of the text files like .txt, .c, .h, .py, .java and more. In addition all of the PDF files or image files like jpg or png, any video files and any music files or any word or excel files. If need be more files extensions can always be added by changing the list "targetExtensions" in the code.

Advantage: storing the paths of the files we are going to encrypt in a list, which is not a

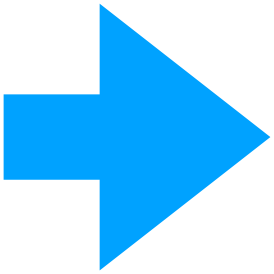
suspicious activity so we can take the time to do it, before actually encrypting them allows us to make the encryption part which is almost (we will explain why almost shortly) the most suspicious part faster. That is because we don't have to waist time with iterating over folders and files that we are not going to encrypt in between every file encryption. In addition it also makes more suitable for parallelism because each time a core has finished encrypting a file it can simply choose the next file that needs to be encrypted and that no other core is currently encrypting from the list.

6.7 Step 7: File Encryption

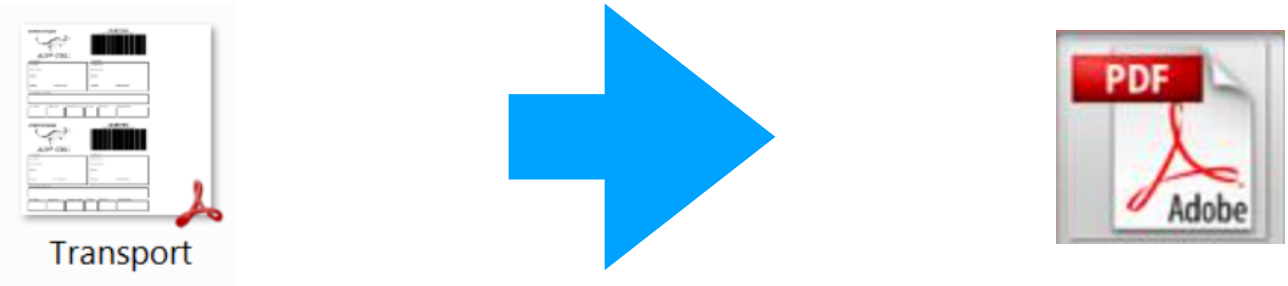
Iterates over the target files list and encrypts all of the files there in a parallel manner using all of the cores. As we just described in the previous section having a list with the paths of all of the files that we are going to encrypt allows us to make this part which is suspicious faster. For every path in the list an available core will create another file in the same place with the same name plus an extension of .enc and write the encrypted version of the original file into the new file. The new file will be specified to the operating system as a hidden file which means that the user won't be able to see it thus making this part which is the longest and most important part completely transparent to the user. In between each file decryption we give the cpu a break to go back and proceed with the user's process for a while in order to keep our cpu percentage low.

Advantages:
First: Deleting a file is an extremely suspicious action and if an anti virus will see a process starting to delete a lot of files it might stop it.
Second: keeping the cpu usage low also helps us avoid anti virus detection. We can allow ourselves to have this process last a little longer because we are completely transparent to the user and if we user a high percentage of the cpu the user would immediately notice because his computer would slow down and we run a higher risk of an anti virus detecting us.
Third: If I were to limitedly erase the original file once i finished creating an encrypted version of it the user would Defiantly notice something is wrong (even if I would keep the same extension and save a list of all of the files I encrypted somewhere on the computer for the decryption process) because of three key indications:

1. first the file icons would start flashing one after the other since they are being deleted and then replaced by another file even if it has the same name.
2. In addition he would his pictures for example become form the first icon to the second which is very suspicious seeing all of you pictures in a file making this transition one after another.



And in addition also PDF files have a preview that will change:



3. Keep in mind that for an average user has at least around 100Gb of files and pictures that we need to encrypt and even at a rate of 100 (Mb/S) it will still take **17 minutes** which in that time he will most likely try to open a file and discover it has been encrypted and realise that he is currently under attack.

So to sum up if I were to immediately delete the original file when I finish creating the encrypted version I run a much higher risk of being detected by both anti virus scanners and the user himself. And if the user Finds out he can unplug the computer thus saving all of the files we didn't manage to encrypt.

Disadvantages:
The Only disadvantage is that we do need extra space in the victims's computer. But that's not a big problem if I had more time I would implement a course of action in which we use up all of the free space the user has and when he runs out of space we "go lowed" meaning we start deleting the original files right after creating the encrypted version. By doing so we bought ourselves as much time as possible for remaining stealthy.

6.8 Step 8: Uploading Encryption Key

When we finish the process of creating all of the encrypted version of the original files we upload the encryption key used to the command and control server so we can restore the victims files when he pays. The C&C stores the key in an encrypted way.
In addition this is done before deleting the original files. To identify a user we compute a UUID - Universally Unique Identifier. This is a is a 128-bit number used to identify information in computer systems. While the probability that a UUID will be duplicated is not zero, it is extremely close to zero to be negligible. The number of random version 4 UUIDs which need to be generated in order to have a 50% probability of at least one collision is **2.71 quintillion**. The UUID is calculated by taking into consideration different parameters including for example the MAC address.
Advantage:
This is done to make sure we are a reliable trustworthy ransomware operation. We don't want to delete all of the user's files and then that an anti virus or a user's shut down will stop us from sending the key to our server. Making it impossible for the victim to pay even if wanted to.
Disadvantage:
if someone would successfully reverse my code even after all of the obfuscation and compression. He could realise how the uuid is being generated and thus hook some of the

operating system's calls to make sure his uuid is for example the same as someone who has bought insurance.
Fix: If I had more time I would come up with a better way of assigning IDs by incorporating random elements into the ID's calculation.

6.9 Step 9: Deleting The Original Files

Now that we know we can restore the victims files need be, we can proceed to the most suspicious part of the process. Deleting a lot of files is very suspicious so we have to do it fast, because in this case it won't help us doing this slowly. So we go over the list of files we have encrypted and in a parallel manner using all of the cores with no rest in between we delete all of the files.
Disadvantage:
It is possible that the access to a remote server is suspicious enough to make an anti virus shut us down before we get a change to delete the original files thus finishing the attack.

6.10 Step 10: Displaying The Hidden Encrypted Files

After finishing the file deletion part we are now in the clear and it is time to let the user know we are here. Now we can display the encrypted files that have been hidden until now.

6.11 Step 11: Notifying The User And Giving Instructions

Now after finishing the encryption process and all of the victims files have been encrypted we also need to inform him of how he can pay us. First we create a file on his desktop called ID.txt which is where we write his UUID so he wouldn't have to calculate it himself. In addition we change the victim's Desktop background so he couldn't miss our message. In it we give him instruction to go to

6.12 Step 12: Obfuscate And Compress

After finishing writing our encryptor it's time to obfuscate and compress for the reasons we discussed in the **"Theoretical Background and Algorithms".** The tool we use to do all of the features we discussed in the **"Theoretical Background and Algorithms"** is called agile.Net. After extensive research this is the best platform to use for .net code and offers all of the feature of obfuscation and compression we discussed.

7 Decryptor

In addition to the Encryptor program we also created a Decryptor program. The decryptor as you can see in the illustration in Chapter 5: **"High Level construction Of MindLost's"** sends a request to the C&C server and only if the user has paid the decryptor will get the key. If the user tries to run the decryptor without paying an error message appears letting him know he can only decrypt his files if he pays. If the user did pay through the our website and then runs the decryptor it'll get the decryption key from the C&C and iterate over all of the files in the same way that the Encryptor did and will decrypt them.

8 Command and Control Server

This is a server I created in Azure but of course if this was a real ransomware this server would be sitting in a third world country where no government has access to is. It is a sql database server that saves all of the victims UUID's with their keys and status which is -1 if the victim hasn't paid yet, 1 if he paid but didn't purchased insurance and 2 if he paid and purchased insurance.

For example a row of a victim who hasn't paid yet would look like this:

UUID	KEY	STATUS
42134D56-77C6-5245-E253-29ACEF95CC9E	52-79-E8-DA-19-FC-15-06-CB-27-16-80-2F-A5-D6-C7-74-80-D7-31-42-7C-DA-78-07-8E-10-16-29-3C-94-38	-1

And example a row of a victim who has paid but didn't purchase insurance would look like this:

UUID	KEY	STATUS
42134D56-77C6-5245-E253-29ACEF95CC9E	52-79-E8-DA-19-FC-15-06-CB-27-16-80-2F-A5-D6-C7-74-80-D7-31-42-7C-DA-78-07-8E-10-16-29-3C-94-38	1

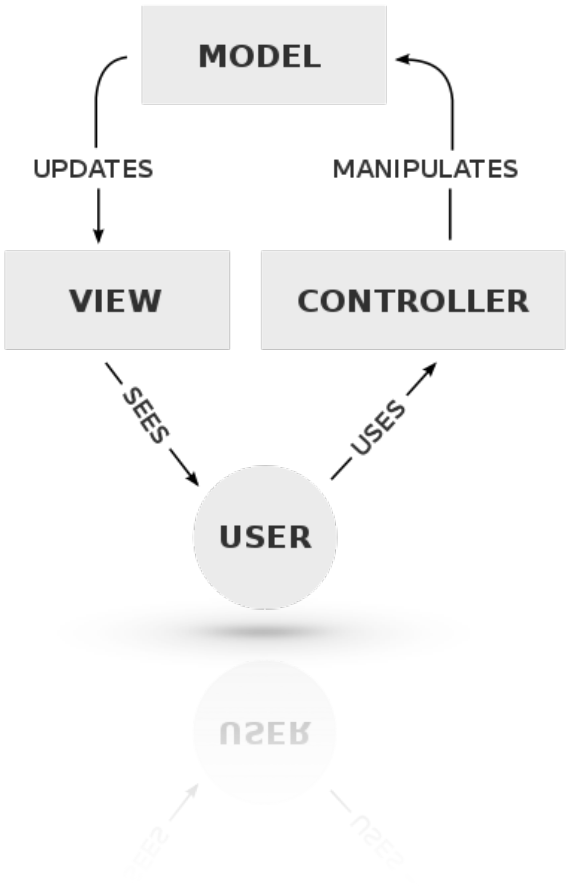
And example a row of a victim who has paid and has purchase insurance would look like this:

UUID	KEY	STATUS
42134D56-77C6-5245-E253-29ACEF95CC9E	52-79-E8-DA-19-FC-15-06-CB-27-16-80-2F-A5-D6-C7-74-80-D7-31-42-7C-DA-78-07-8E-10-16-29-3C-94-38	2

9 Payment Website - http://mindlost.azurewebsites.net

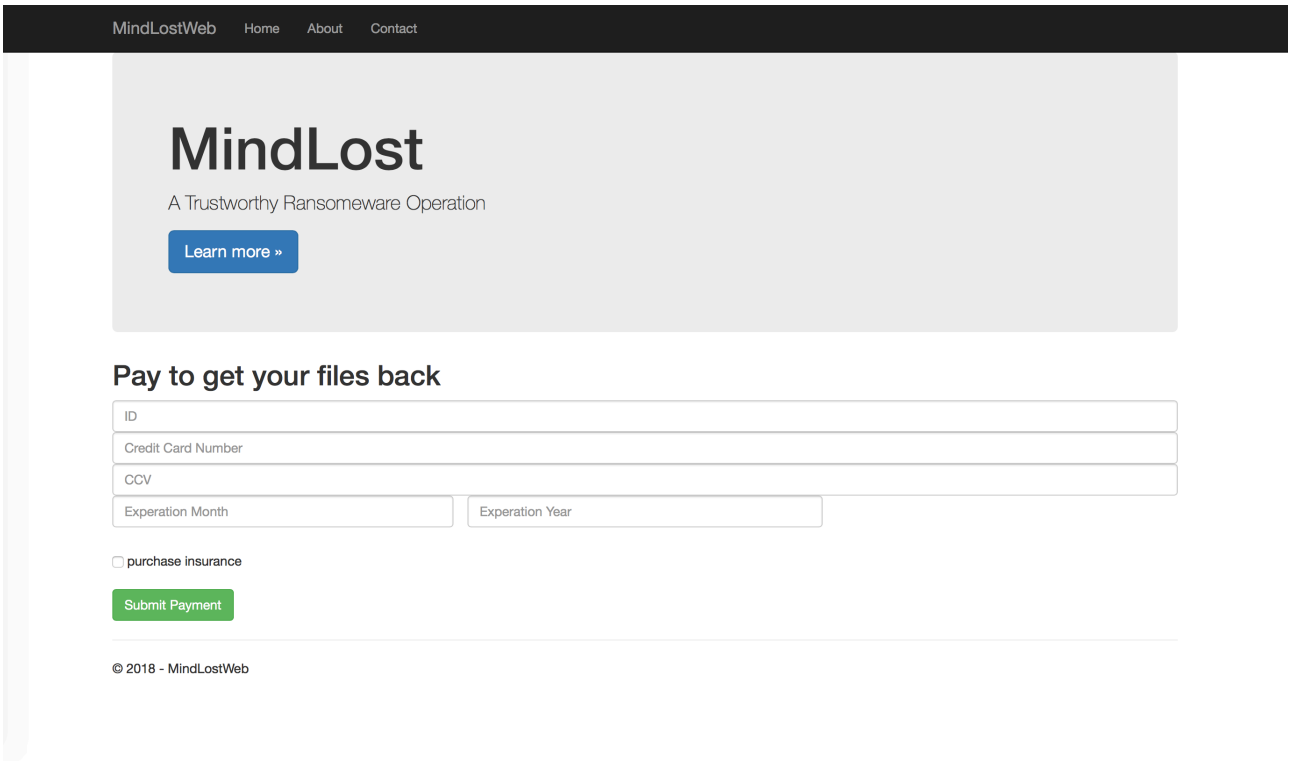
9.1 Website's Backend Structure

we have to simulate a payment method in some way so I decided to build an entire functional website which is up and running on an Azure server. The website was built in Visual Studio and is an ASP.Net core web MVC application.
It was written in C# and cshtml with MVC model of Model–view–controller.



9.2 Website's main page and logic

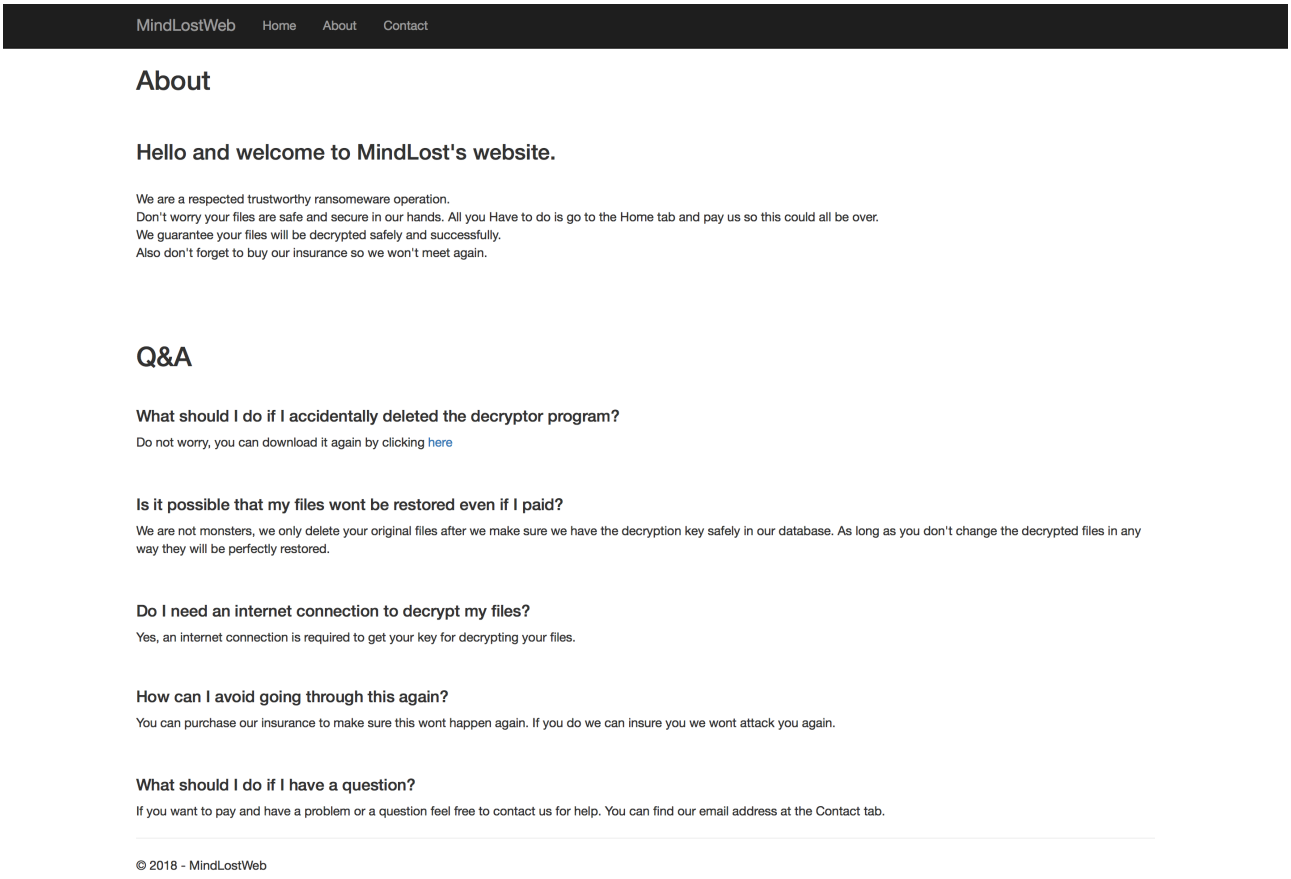
The website looks like this when you enter it:



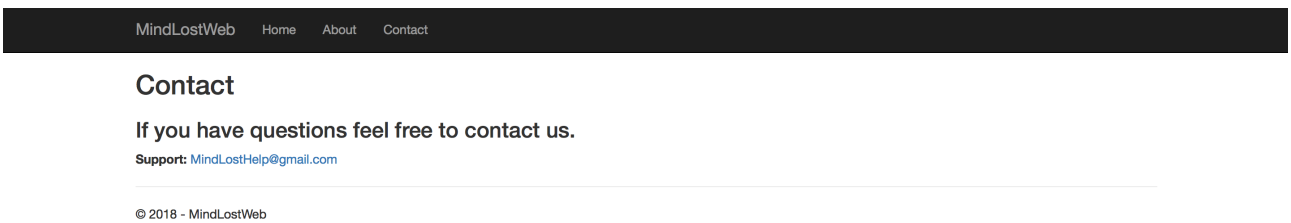
A victim then enters his UUID which he can find in the file we wrote for him called ID.txt on this desktop. Then he enters his credit card information and can decide if to purchase insurance or not. Our website checks that the the ID that was entered is valid and exits in our Command and Control server. And it also checks that the user has filled the credit card information tabs but doesn't charge anything obviously. If the victim checked the insurance box which is supposed to cost extra money the website notifies the C&C to update that user's status to 2 and if he didn't check the insurernce checkbox but still did fill in his credit card information his status will be changed to 1 meaning that if he ever runs MindLost agin all of his files will be encrypted again.

9.3 About page

I also implemented an About page in our website with a short description of MindLost and then frequently asked questions and answers.
Such as what should I do in case I accidentally deleted the decryptor file. In which case for example I provide a link do download it again. Because I want to help the victims get their files back in as easily as possible so the would pay. And many more questions and answers.



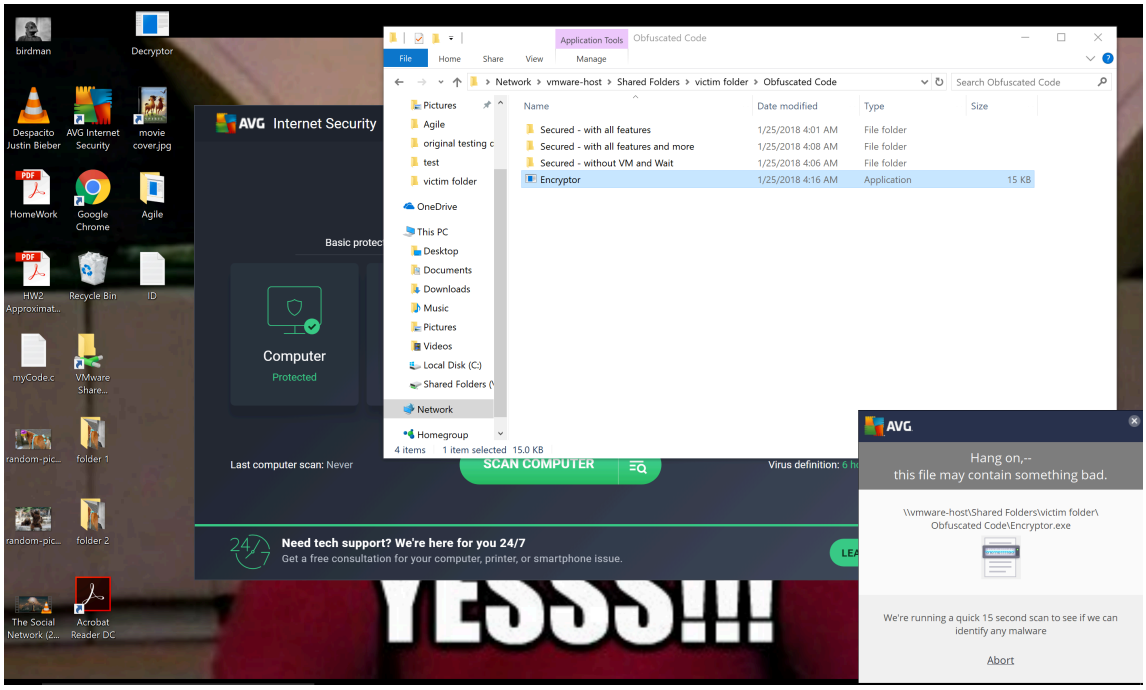
Plus a Contact Page where a victim can find an email address (which I actually registered) to send questions to. Again in order to make the process as easy as possible.



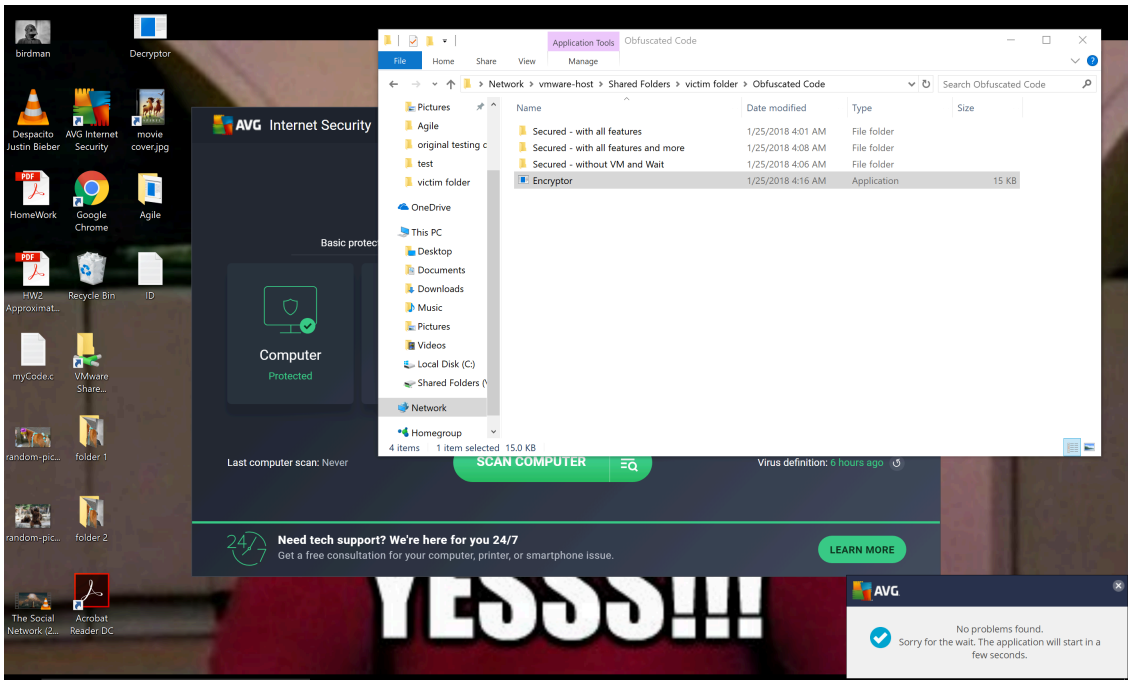
10 Testing MindLost Against Anti Viruses

10.1 MindLost Vs AVG's Active Realtime defender

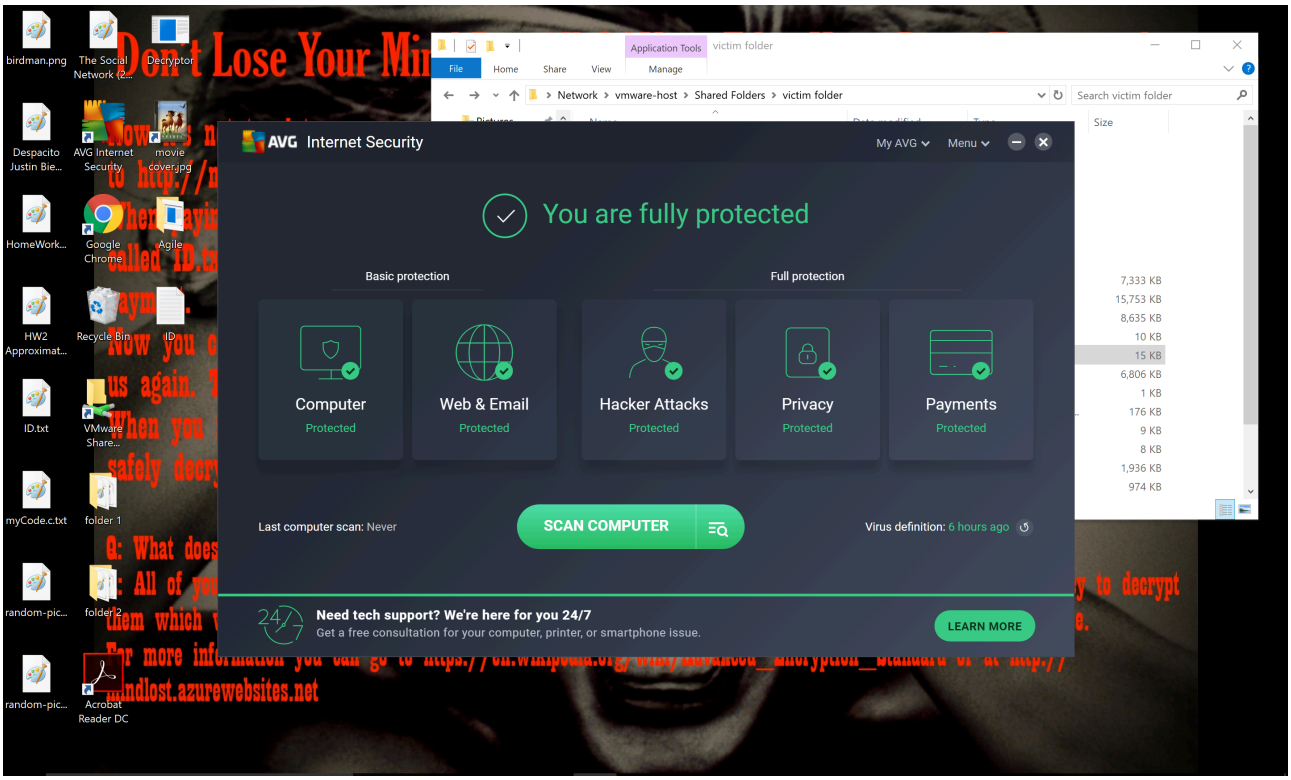
I installed AVG's Active protector. then I ran MindLost to see if the anti virus can dynamically detect that this is melodious code. The moment I ran the encryptor a morning message appeared in AVG saying AGV is not sure if this file is suspicious or not so it needs 15 seconds to test it:



After those 15 seconds it said the file is ok and that it is going to continue running it for us.



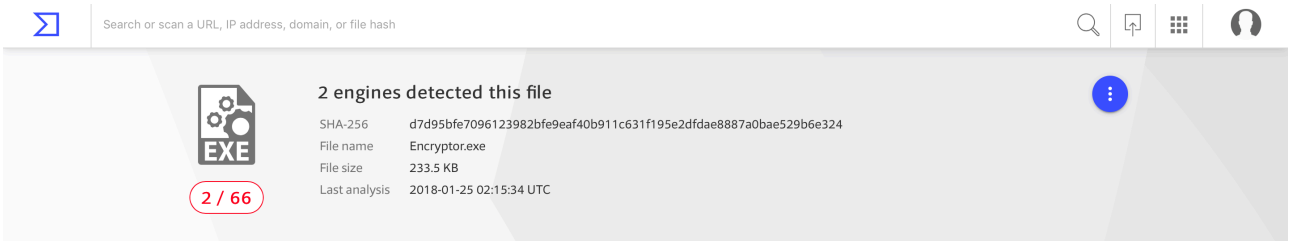
Then MindLost finished the job successfully and encrypted all of the users files without AVG objecting. As you can see in the image below the computer has been encrypted even



while AVG's active realtime protector was running with it only suspecting it for a few seconds at first. After running it again AVG didn't even suspect it and just let it encrypt the entire computer. Great Success.
In addition it is also really interesting that even if I purposely block the process through AVG during the encryption process MindLost will still finish encrypting all of the files because I open multiple threads during the encryption process.

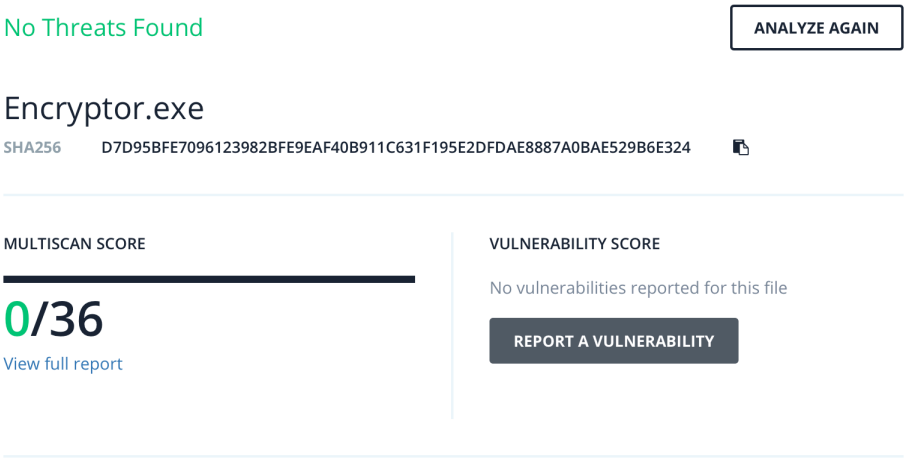
10.2 Virus total

A 0.3% detection rate



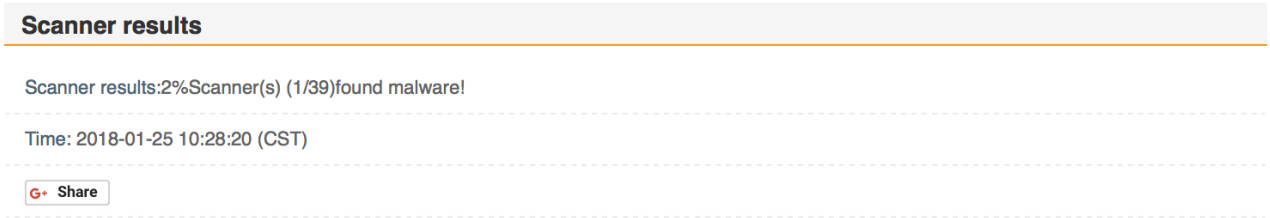
10.3 Metascan

A 0% detection rate



10.4 VirSCAN

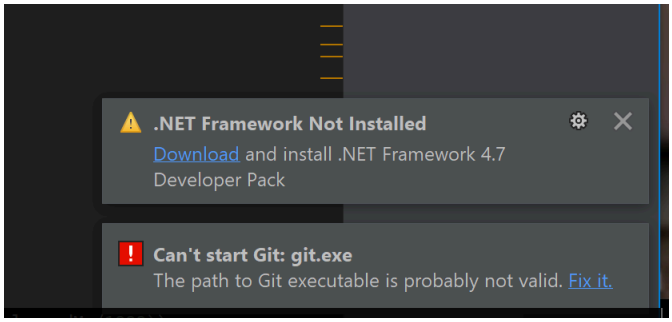
A 2% detection rate (1/39 Anti viruses)



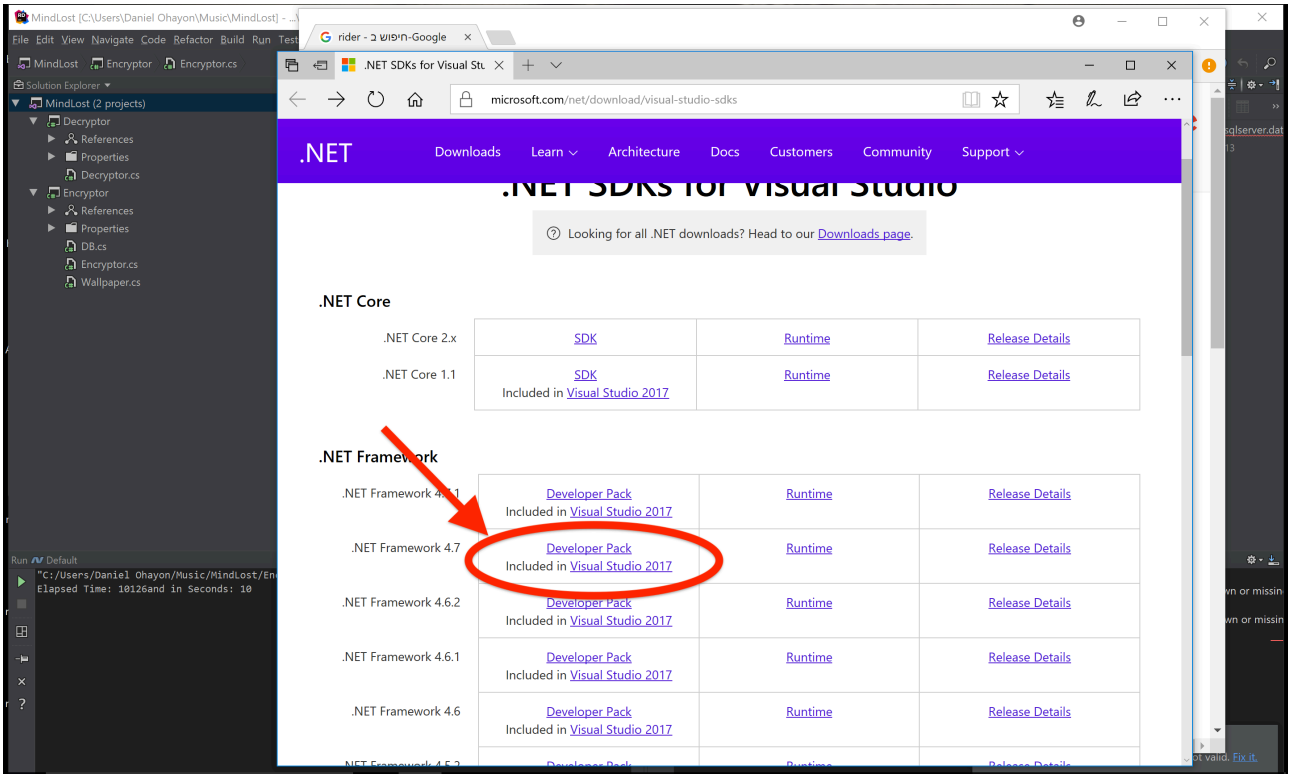
11 Compiling Instructions

11.1 Installation

For The Encryptor and Decryptor Go to the MindLost folder. Then open the project in Rider which is a Jetbrains c# IDE. To do so simply open the .sln file with Rider (right click on the sun file and choose open with) and the project will load. If you don't have rider installed on your computer you can literally install in a few short minutes and then use your campus mail to activate an educational account. Also Make sure you have .NET framework 4.7 installed. If you don't like in this case:



Go ahead and install it from https://www.microsoft.com/net/download/visual-studio-sdks



After you do that go ahead and press build the project.

IMPORTANT - Make sure you are running this on a virtual machine. Do not risk your own system.

Since you are going to run this on a virtual machine The version of the code I submitted has the line that calls the function that detects if MindLost is running on a VM and if so aborts commented out. And also the line calling the smart interval waiting before encryption. This is done so that you could run the ransomware on your virtual machine. As you can see in the image below.

```
public static void Main(string[] args)
{
    hideConsole();
    writeToRegistry();
    //checkSysteInfoForVM();
    //waitGivenMinutes(MINUTES_TO_WAIT);
    checkDBStatus();

    AesCryptoServiceProvider aes = createCipher();
    encryptAllFiles(aes);

    notifyUser();
}
```

I made sure that this process works by installing Rider on my Virtual Machine and making sure i download the .NET 4.7 Framework from the website above. By simply opening the .sln file with rider it will first display a warning about .git but then it should compile properly.

11.2 Workflow

You can play with the code. Once you compile it in order to run the encryptor go to Encryptor/bin/Debug/ and run Encryptor.exe. In order to run the decryptor go to Decryptor/bin/Debug/ and run Decryptor.exe.

After running the encryptor if you want to decrypt the files, go to the website http://mindlost.azurewebsites.net and enter the ID from your desktop. then fill in the credit card details and click submit. Afterwards you can run the Decryptor. If you try to run it before it just won'y let you.
Notice that if you purchase an insurance then the next time you want to encrypt the computer you must uncomment a specific command and then don't forget to comment it again. The line is the one between the stars ****:

```
public static void Main(string[] args)
{
    hideConsole();

    //**************************************************
    /*For Debugging purpose only:
      In case you gave the user an insurance you wont be able to attack him again
      unless you delete him from the DB. Do do so uncomment the next command.
    */
    //DB.deleteVictimsRow();
    //**************************************************

    writeToRegistry();
    //checkSysteInfoForVM();
    //waitGivenMinutes(MINUTES_TO_WAIT);
    checkDBStatus();

    AesCryptoServiceProvider aes = createCipher();
    encryptAllFiles(aes);

    notifyUser();
}
```

I also added three executables. One is just the original encryptor without the anti VM and waning. then I added two more executables which I had obfuscated and compressed with agile.NET.

11.3 Website

If you also want to run the website on a local host on you computer you can go to the MindLostWeb folder and in there you will find the visual basic project that was used to create the website. But you can also always just find it at http://mindlost.azurewebsites.net

12 Summary

This project is defiantly the busiest project I have ever done partly because I ended up doing it alone but also because I really enjoyed working on it and just spent hours researching and reading about different ways to achieve what I wanted. This was also the project I enjoyed most working on so far. I learned so much by researching different anti virus evasion techniques and by doing so learning more about the way the operating system itself is working and the different kinds of anti viruses or different kinds of malware. This is defiantly a fascinating field for me and I'm just happy I decided to take this project. I defiantly

13 Bibliography

1. AES - Wikipedia https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
2. Obfuscation - https://en.wikipedia.org/wiki/Obfuscation_(software)
3. Code Compression - https://en.wikipedia.org/wiki/Universal_code_(data_compression)
4. Parallel programming - https://en.wikipedia.org/wiki/Parallel_programming_model