BlackSnake Ransomware Emerges from Chaos Ransomware's Shadow

blog.cyble.com/2023/03/09/blacksnake-ransomware-emerges-from-chaos-ransomwares-shadow/

March 9, 2023

New Ransomware Goes Beyond Traditional Tactics with Clipper Integration

Ransomware is a significant threat that can encrypt its victims' files and demand a ransom. Additionally, the Threat Actors (TAs) responsible for these attacks often use a double extortion technique, where they encrypt the files and exfiltrate sensitive data from the victim's device before encryption. These TAs then leverage this stolen data to extort their victims further by threatening to release it on a leaked site unless their demands are met.

The TAs are constantly devising new methods to extort money from their victims. In the previous year, Cyble Research and Intelligence Labs (CRIL) discovered a ransomware variant that not only encrypts victims' files but also steals their Discord tokens.

Recently, CRIL spotted a new strain of malware known as the "BlackSnake" ransomware that is capable of performing clipper operations aimed at cryptocurrency users. This variant was initially identified by a researcher @siri_urz. It was detected in the cybercrime forum in 2022, and the TAs behind it were actively seeking affiliates.

In addition, the TAs claimed they would take a 15% share of the profits generated through the affiliation process, as shown in the figure below.

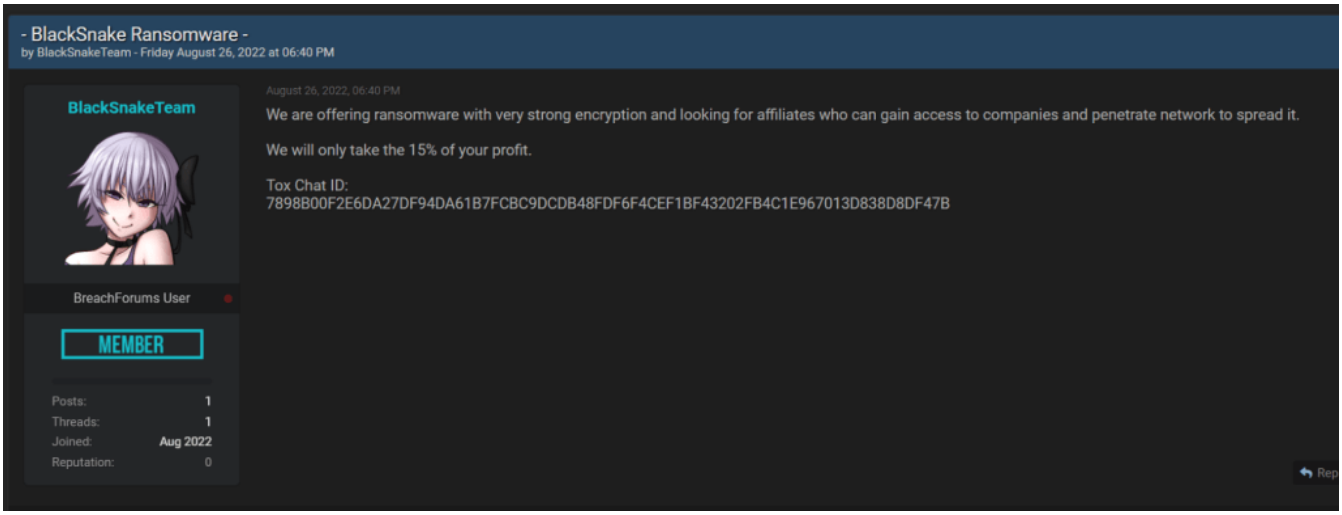


Figure 1 –TAs advertisement in Cyber Crime Forum

Our analysis has uncovered evidence suggesting that BlackSnake Ransomware has been created based on the source of Chaos ransomware. In this blog, we delve into the technical aspects of BlackSnake Ransomware, including its clipper operations.

Technical Analysis

Static analysis of the sample with hash: e4c2e0af462ebf12b716b52c681648d465f6245ec0ac12d92d909ca59662477b shows that the malicious file is a 32-bit PE binary compiled using .NET, as demonstrated in the following figure:

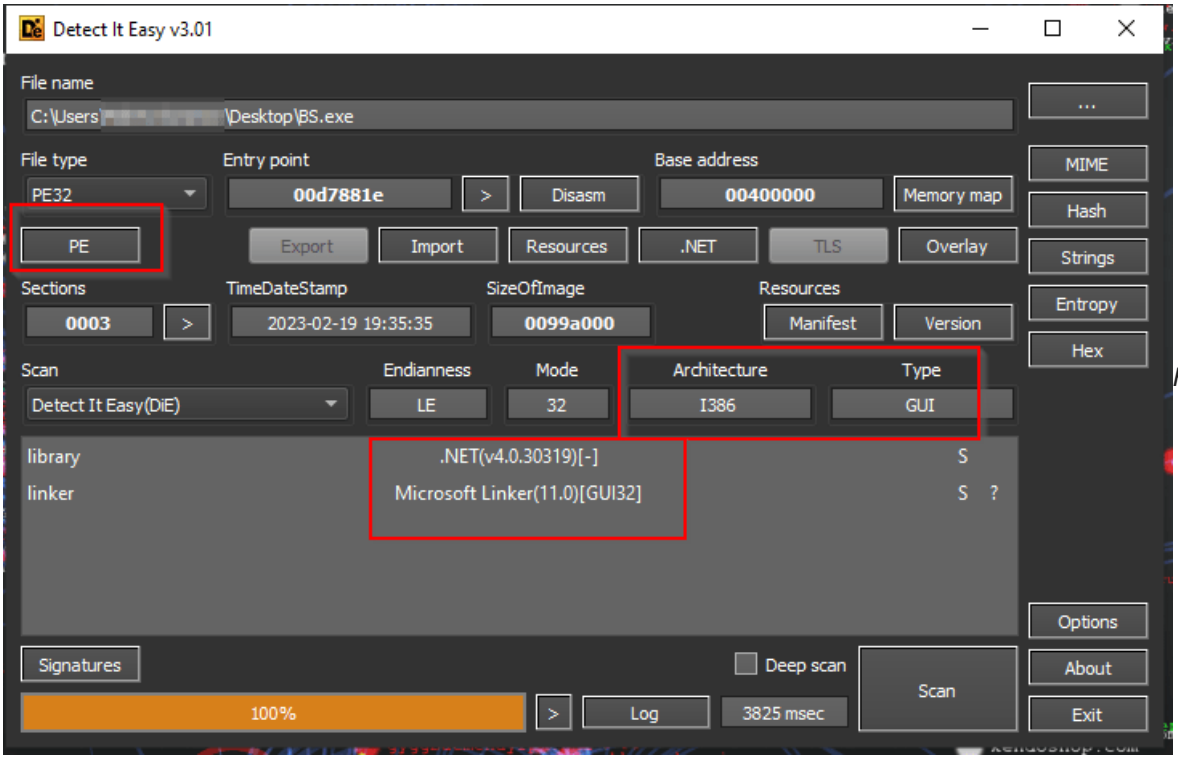


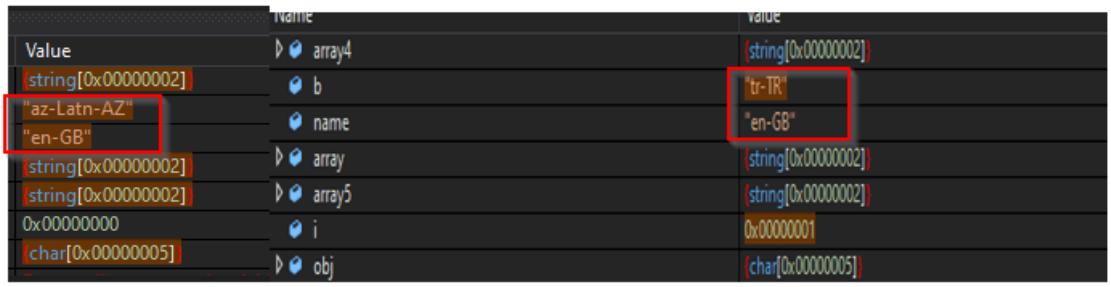
Figure 2 –File Information

Upon execution, the BlackSnake Ransomware performs an initial check to verify if the current input language of the system matches the language codes “az-Latn-AZ” or “tr-TR”.

If a match is found, the ransomware immediately terminates itself, indicating that the TAs of BlackSnake ransomware intend to exclude systems located in Azerbaijan or Turkey from being infected, as shown below.

```
foreach (string b in array4)
{
    try
    {
        string name = InputLanguage.CurrentInputLanguage.Culture.Name;
        if (name == b)
        {
            return true;
        }
    }
    catch
    {
    }
}
return false;
}
```

Figure 3 –



Locale Check

After confirming the user's location, the BlackSnake Ransomware creates a registry entry, as shown below.

HKEY_CCURRENT_USER\SOFTWARE\oAnWieozQPsRK7Bj83r4

The BlackSnake ransomware has a method of detecting whether it has already infected a system. It does this by checking the location of the executing assembly with the path "C:\Users[user-name]\AppData\Roaming\svchost.exe". If this path matches, the ransomware continues to search for the file named "UNLOCK_MY_FILES.txt" in the %appdata% directory. Once the file is found, the ransomware will terminate itself. This behavior suggests that the ransomware is designed to avoid infecting a system more than once, and it may be an attempt to limit the impact of the ransomware.

The below figure shows the code snippet used by the malware for validation.

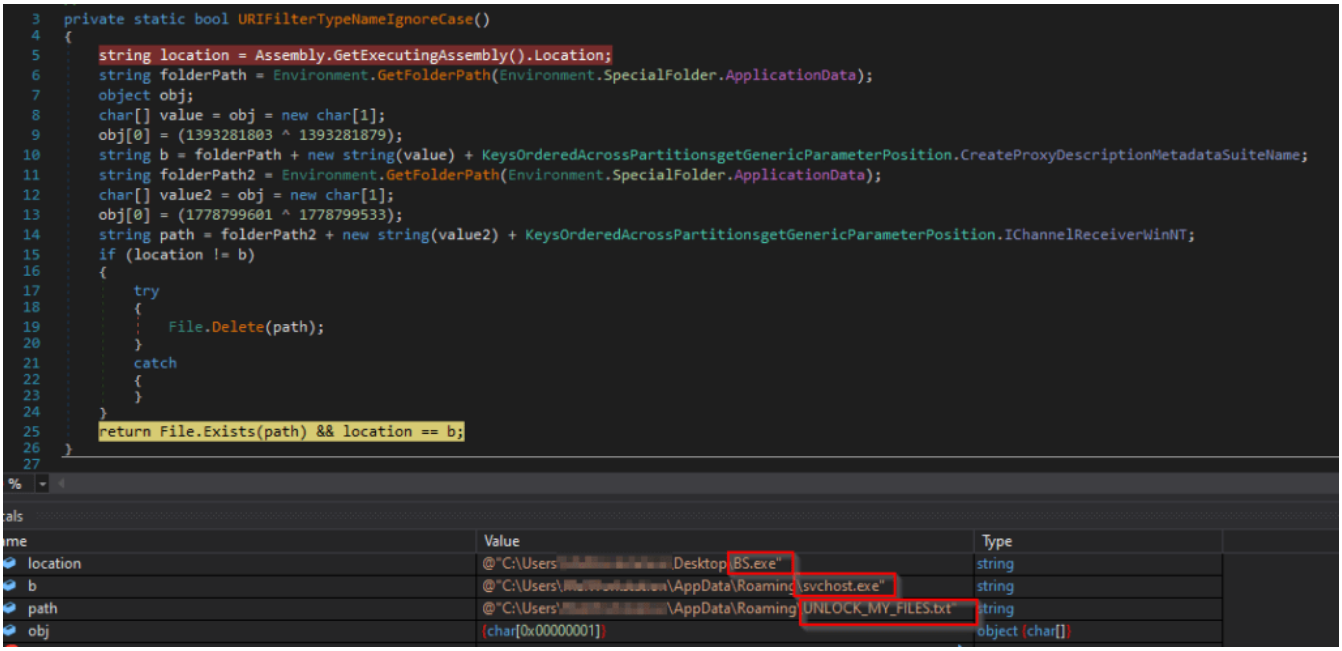


Figure 4 – File Path Validation

To prevent multiple instances of the malware from running concurrently, the malware enumerates the names of all currently running processes, retrieves the filename of the current executing assembly, and compares it with the filenames of the running processes. If there is a match, the malware then compares the Process ID of the current process with that of the target process. If there is a difference in the IDs, the malware identifies itself as a duplicate instance and terminates itself to avoid running multiple copies at the same time.

The below figure shows the code snippet used by the ransomware for checking the malware instance.

Figure 5 –Check for Duplicate Instances of Malware

After confirming that there is no existing infection of itself, the ransomware creates a copy of itself in the %appdata% directory with the file name “svchost.exe” and executes the newly created process as shown below.

```
private static void NAMEsetRNG(string processName)
{
    string friendlyName = AppDomain.CurrentDomain.FriendlyName;
    string location = Assembly.GetExecutingAssembly().Location;
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData);
    object obj;
    char[] value = obj = new char[1];
    obj[0] = (622753875 ^ 622753807);
    string text = folderPath + new string(value);
    string text2 = text + processName;
    if (friendlyName != processName || location != text2)
    {
        byte[] bytes = File.ReadAllBytes(location);
        if (!File.Exists(text2))
        {
            File.WriteAllBytes(text2, bytes);
            ProcessStartInfo processStartInfo = new ProcessStartInfo(text2);
            processStartInfo.WorkingDirectory = text;
            if (new Process
            {
                StartInfo = processStartInfo
            }.Start())
            {
                Environment.Exit(1);
                return;
            }
        }
```

Figure 6 – Creates New Process and Executes

The ransomware now creates a new thread for executing the clipper module, which includes functions such as GetText(), PatternMatch(), and SetText(). These functions allow the clipper module to perform its intended task of intercepting and modifying clipboard data as needed.

The below figure shows the clipper module.

```
protected override void WndProc(ref Message m)
{
    if (m.Msg == 797)
    {
        driveNotification.NotificationForm.ReflectionPermissionIBindCtx = driveNotification.NotificationForm.GetText();
        string reflectionPermissionIBindCtx = driveNotification.NotificationForm.ReflectionPermissionIBindCtx;
        object obj;
        char[] value = obj = new char[3];
        obj[0] = (2091169696 ^ 2091169730);
        obj[1] = (418241642 ^ 418241545);
        obj[2] = (1917655345 ^ 1917655296);
        if (reflectionPermissionIBindCtx.StartsWith(new string(value)))
        {
            if (this.getApplicationManifestBytesgetILOffset(KeysOrderedAcrossPartitionsgetGenericParameterPosition.PointerCOREISOSTORE) && !
              driveNotification.NotificationForm.ReflectionPermissionIBindCtx.Contains
              (KeysOrderedAcrossPartitionsgetGenericParameterPosition.TokenMandatoryPolicyArraySegmentEnumerator))
            {
                string text = KeysOrderedAcrossPartitionsgetGenericParameterPosition.PointerCOREISOSTORE.Replace(driveNotification.NotificationForm.ReflectionPermissionIBindCtx,
                  KeysOrderedAcrossPartitionsgetGenericParameterPosition.TokenMandatoryPolicyArraySegmentEnumerator);
                driveNotification.NotificationForm.SetText(text);
            }
        }
        else if (this.getApplicationManifestBytesgetILOffset(KeysOrderedAcrossPartitionsgetGenericParameterPosition.PointerCOREISOSTORE) && !
          driveNotification.NotificationForm.ReflectionPermissionIBindCtx.Contains(KeysOrderedAcrossPartitionsgetGenericParameterPosition.IsStaticgetWrapper))
        {
            string text2 = KeysOrderedAcrossPartitionsgetGenericParameterPosition.PointerCOREISOSTORE.Replace(driveNotification.NotificationForm.ReflectionPermissionIBindCtx,
              KeysOrderedAcrossPartitionsgetGenericParameterPosition.IsStaticgetWrapper);
            driveNotification.NotificationForm.SetText(text2);
        }
    }
    base.WndProc(ref m);
}
```

Figure 7 – Clipper Module

By constantly monitoring the user's clipboard activity, the BlackSnake malware can check whether any cryptocurrency addresses are present by utilizing a hardcoded regular expression pattern for validation, as shown below.

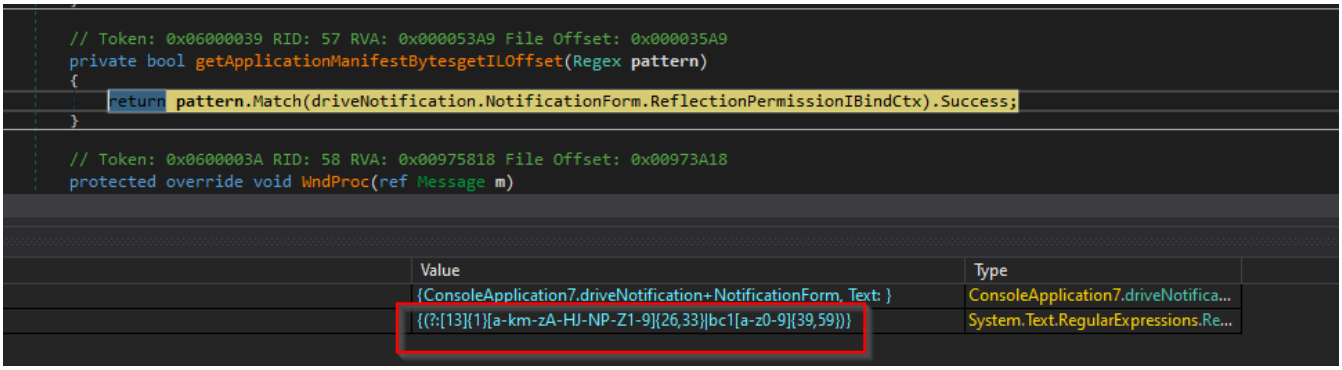


Figure 8 – regex pattern Match

The BlackSnake clipper module appears to specifically target Bitcoin wallet addresses, as indicated by the pattern used for identification.

When a matching wallet address is found in the clipboard data, the malware utilizes the SetText() method to replace it with a hardcoded Bitcoin wallet address belonging to the attacker, as shown in the figure below.

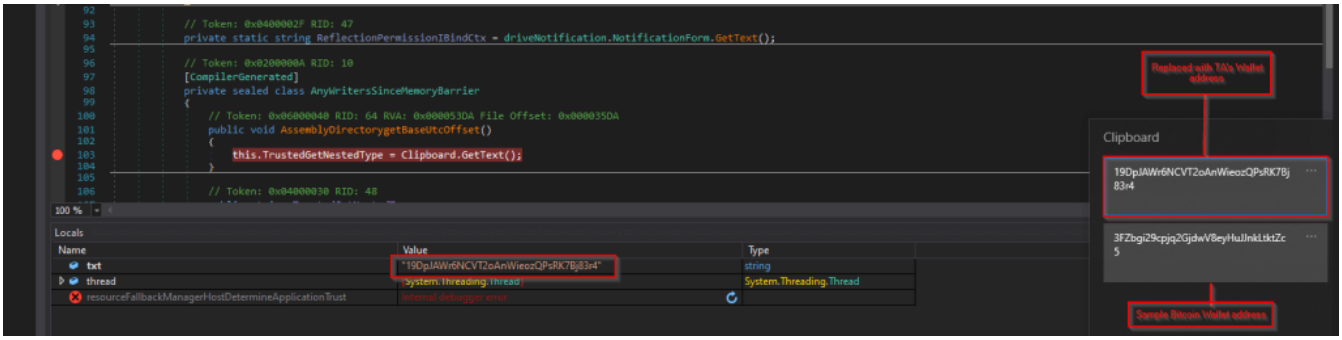


Figure 9 – Replacing Clipboard value with TA's wallet address

Once the clipper module is executed, the BlackSnake ransomware jumps to the encrypting modules. The malware creates a below registry entry that automatically launches whenever the system starts to ensure it remains active and persistent on the infected system.

HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

"C:\Users\ [user-name]\AppData\Roaming\svchost.exe"

Before encrypting files, the ransomware identifies the list of directories to be enumerated and excludes a few folders from its encryption process. The below figure shows the folders excluded by the ransomware.

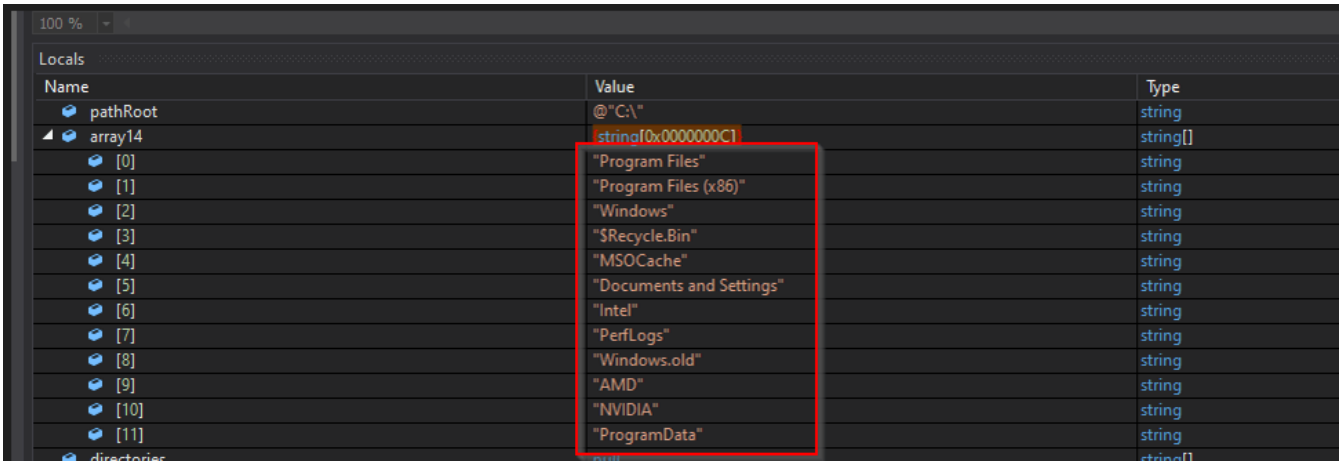


Figure 10 – Folder excluded by BlackSnake Ransomware

Once the relevant directories are identified, the malware enumerates all the files. During this stage, the ransomware checks the file path against a pre-defined list of strings, as mentioned in Figure 11. Any file path that matches these strings is then excluded from the encryption process.

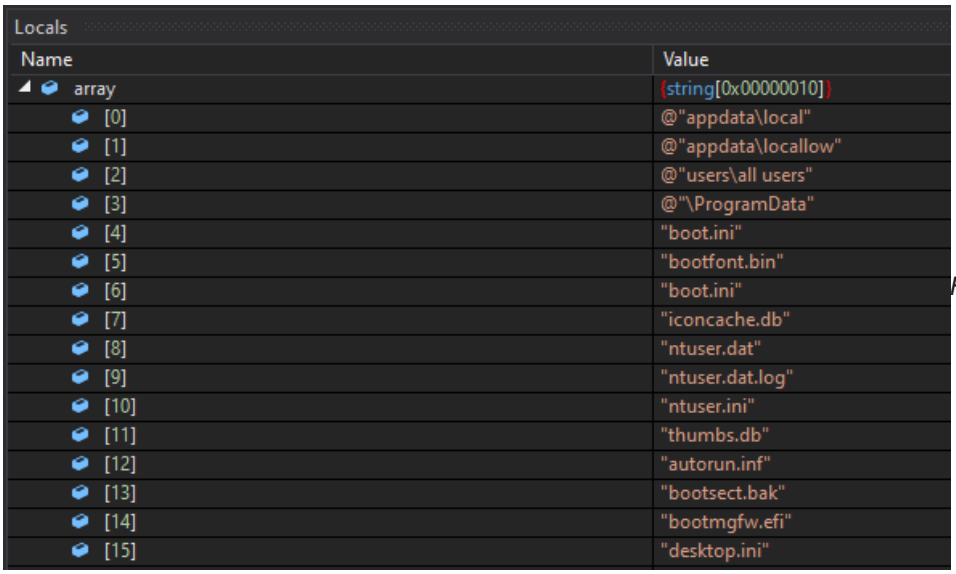


Figure 11 – Exclusion List

The ransomware specifically focuses on encrypting files that have the below file extensions.

.txt	.jar	.dat	.contact	.settings	.doc	.docx	.xls
.xlsx	.ppt	.pptx	.odt	.jpg	.mka	.mhtml	.oqy
.png	.csv	.sql	.mdb	.php	.asp	.aspx	.html
.htm	.xml	.psd	.pdf	.xla	.cub	.dae	.indd
.mp3	.mp4	.dwg	.zip	.rar	.mov	.rtf	.bmp
.mkv	.avi	.apk	.lnk	.dib	.dic	.dif	.divx
.iso	.7zip	.ace	.arj	.bz2	.cab	.gzip	.lzh
.tar	.jpeg	.mpeg	.torrent	.mpg	.core	.pdb	.ico
.pas	.wmv	.swf	.cer	.bak	.backup	.accdb	.bay
.p7c	.exif	.vss	.raw	.m4a	.wma	.flv	.sie
.sum	.ibank	.wallet	.css	.crt	.xlsm	.xlsb	.cpp
.java	.jpe	.ini	.blob	.wps	.docm	.wav	.3gp
.webm	.m4v	.amv	.m4p	.svg	.ods	.vdi	.vmdk
.onepkg	.accde	.jsp	.json	.gif	.log	.config	.m1v
.sln	.pst	.obj	.xlam	.djvu	.inc	.cvs	.dbf
.tbi	.wpd	.dot	.dotx	.xltx	.pptm	.potx	.potm
.pot	.xlw	.xps	.xsd	.xsf	.xsl	.kmz	.accdr
.stm	.accdt	.ppam	.pps	.ppsm	.1cd	.3ds	.3fr
.3g2	.accda	.accdc	.accdw	.adp	.ai3	.ai4	.ai5
.ai6	.ai7	.ai8	.arw	.ascx	.asm	.asmx	.avs

.bin	.cfm	.dbx	.dcm	.dcr	.pict	.rgbe	.dwt
.f4v	.exr	.kwm	.max	.mda	.mde	.mdf	.mdw
.mht	.mpv	.msg	.myi	.nef	.odc	.geo	.swift
.odm	.odp	.oft	.orf	.pfx	.p12	.pls	.safe
.tab	.vbs	.xlk	.xlm	.xlt	.xltm	.svgz	.slk
.tar.gz	.dmg	.psb	.tif	.rss	.key	.vob	.epsp
.dc3	.iff	.onepkg	.onetoc2	.opt	.p7b	.pam	.r3d
.pse	.webp						

The BlackSnake ransomware encryption process consists of several stages. In the first step, the malware employs a string_Builder() function to generate a 40-byte random string. Next, it retrieves a pre-defined RSA public key that is hard-coded within the malware file. This key encrypts the previously generated random string, producing a key suitable for AES encryption.

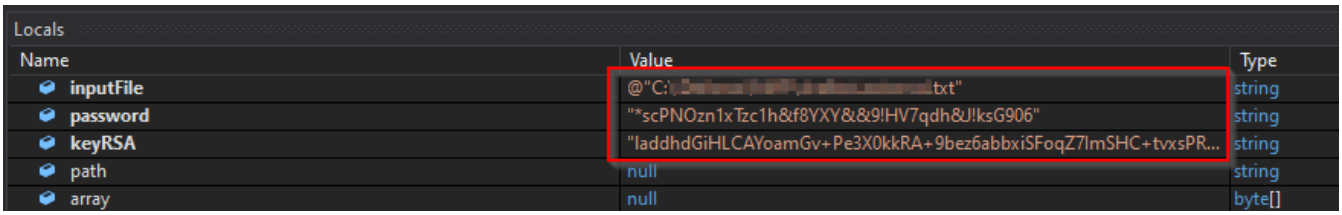


Figure 12 – parameters passed to the encryption function

Once the malware gets the key, it encrypts all the identified files from the directory using the AES algorithm and appends the generated key (base64 encoded) to the end of the encrypted file.

The below figure shows the key appended to the encrypted file.

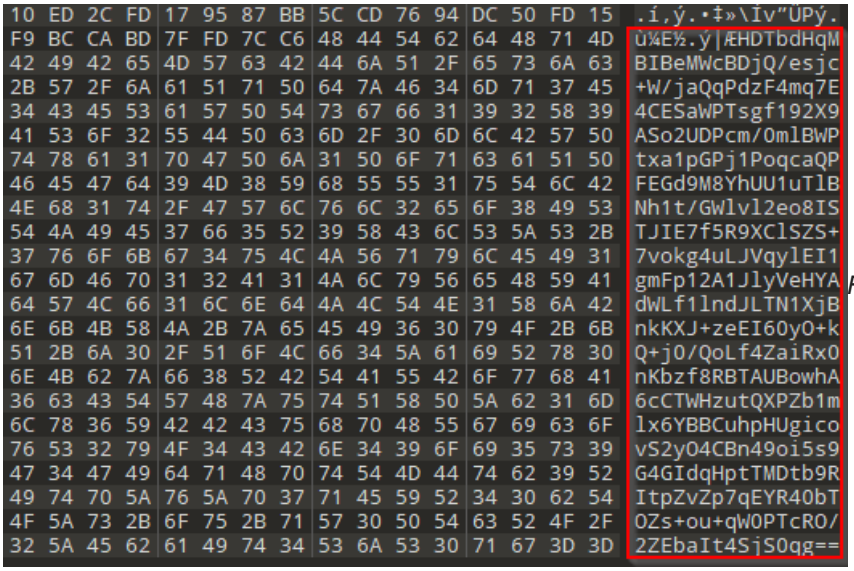


Figure 13 – RSA Encrypted Key

On successful encryption, it appends the “pay2unlock” extension to the encrypted files and drops a ransom note in that folder.

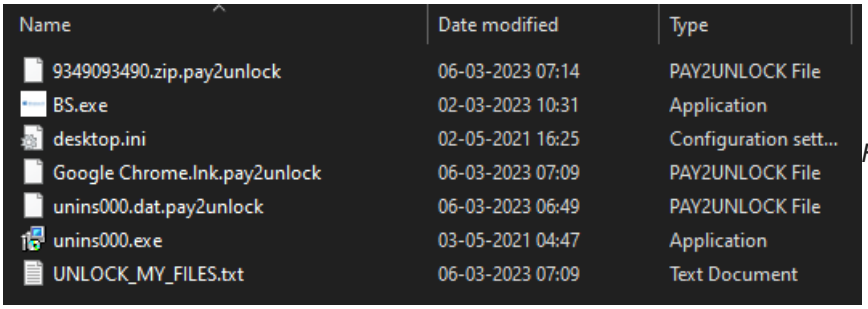


Figure 14 – Encrypted files

Finally, the victims are presented with a ransom note, "*UNLOCK_MYFiles.txt*" that directs them to contact the attackers via their TOX_ID if they wish to recover their encrypted files, as shown below.

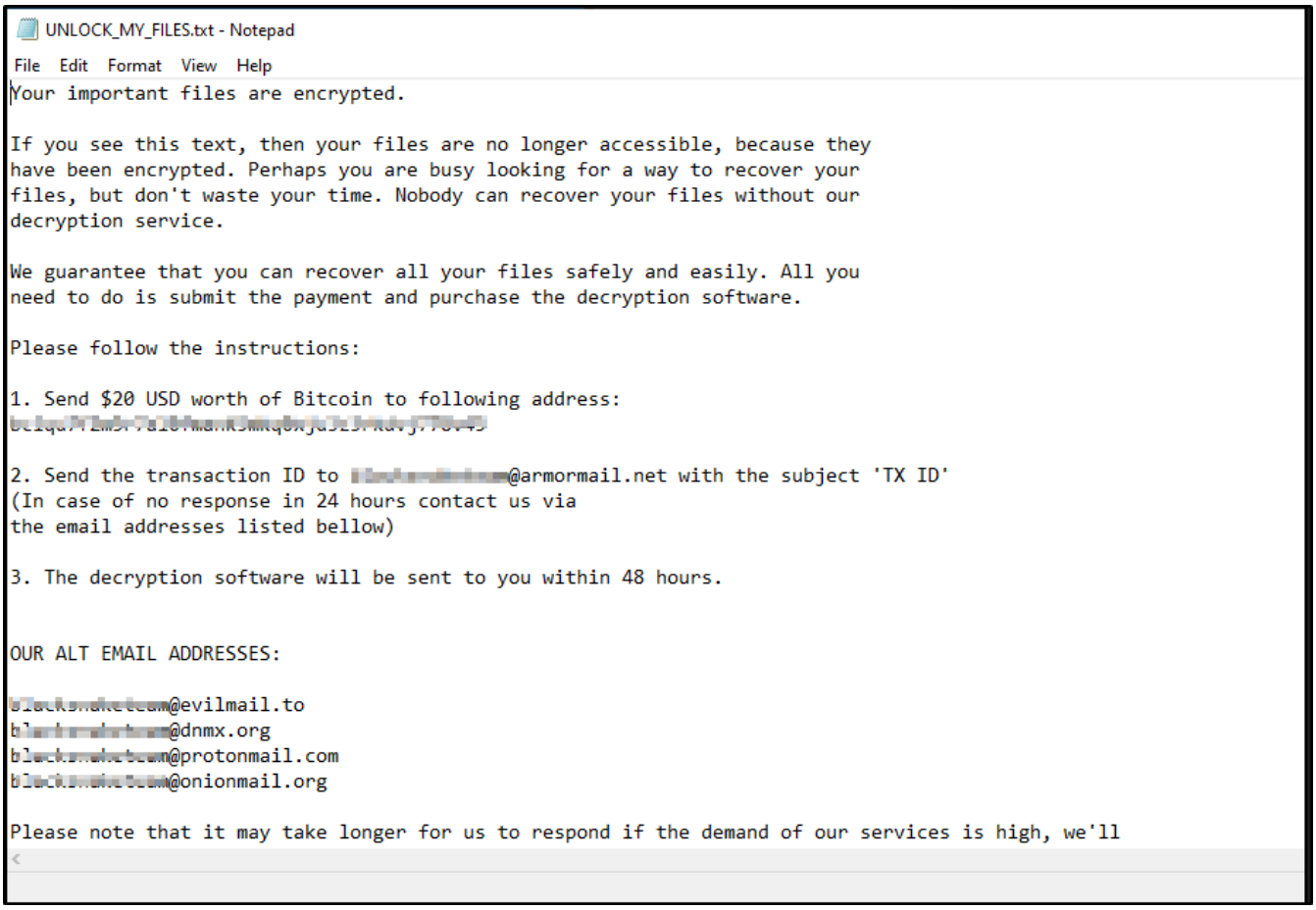


Figure 15 – Ransom note

Conclusion

It is convenient and straightforward for TAs to use pre-existing ransomware codes as a basis for developing new ransomware families. Onyx and Yashma ransomware families were already linked to the Chaos ransomware family, and the BlackSnake ransomware is another family now associated with Chaos ransomware. The Threat Actor has tweaked the Chaos ransomware source code and added a clipper module directly into the file, which is different from the usual approach of having a separate file for the clipper.

Cyble Research & Intelligence Labs continuously monitors all ransomware campaigns and will keep updating our readers with the latest information as and when we find it.

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 best practices given below:

- Back up data on different locations and implement Business Continuity Planning (BCP). Keep the Backup Servers isolated from the infrastructure, which helps fast data recovery.
- Frequent Audits, Vulnerability Assessments, and Penetration Testing of organizational assets, including network and software.
- Enforcement of VPN to safeguard endpoints.
- Conduct frequent training on security awareness for the company's employees to inform them about emerging threats.
- Implementation of technology to understand the behavior of the ransomware-malware families and variants to block malicious payloads and counter potential attacks.
- The users should carefully check their wallet addresses before making any cryptocurrency transaction to ensure there is no change when copying and pasting the actual wallet addresses.
- The seeds for wallets should be stored safely and encrypted on any devices.

MITRE ATT&CK® Techniques

Tactic	**Technique ID**	**Technique Name**
Execution	T1204	User Execution
Impact	T1486 T1490	Data encrypted for impact Inhibit System Recovery
Discovery	T1082 T1083 T1057	System Information Discovery File and Directory Discovery Process Discovery
Defense Evasion	T1140	Deobfuscate/Decode Files or Information
Persistence	T1547	Registry Run Keys / Startup Folder

Indicators of Compromise (IOCs)

Indicators	Indicator Type	Description
e4c2e0af462ebf12b716b52c681648d465f6245ec0ac12d92d909ca59662477b	Sha256	BlackSnake Ransomware
afa9d7c88c28e9b8cca140413cfb32e4	MD5	BlackSnake Ransomware
6936af81c974d6c9e2e6eaedd4026a37135369bc	SHA-1	BlackSnake Ransomware