G DATA
SIMPLY
SECURE

G Data Whitepaper 11/07/2017

Analysis of Rurktar

MSIL.Backdoor.Rurktar.A

Analysis by: Nathan Stern

G DATA Software AG | Application Development | July 2017

Contents

Rurktar

1. Introduction

There is a new malware called Rurktar[2]. It's a trojan spy which is installed as service called RCSU. The service connects back to the attacker machine and waits for commands which will be given by the attacker. The file size of the malware is mostly around ~50Kb, as you can see from the list of sample hashes at the end of this report. Currently, the trojan spy is still in development and is not spotted in-the-wild yet. This could change once the trojan spy has fully developed.

2. General structure

The malware contains 7 namespaces with one or more classes inside.

"Conn" -> Core functionalities like connecting back to the attacker by using a list of multiple IP's and listening for commands and executing them.

"Ini" -> Used to read and write information from and to the .ini configuration file for the malware.

"Prefs" -> The default configuration settings of the malware are hardcoded here.

"RCS" -> With administrator privileges, the malware installs itself as a windows service called

"RCSU" -> Inside is the main class, which gets executed first and calls all the other classes.

"RCSUConn" -> A helper class for the "Conn" namespace with a method to read incoming commands.

"Trinet.Networking" -> Used to enumerate local and remote network shares. The code used from https://www.codeproject.com/Articles/2939/Network-Shares-and-UNC-paths

3. Debug path

When viewing the strings of the malware, the debug path becomes visible and reveals some interesting information like the username "Alex" and also, that the user stores the malware in his Dropbox folder.

Full debug path: "c:\Users\Alex\Dropbox\Projects\RCS\RCSU\obj\Release\RCSU.pdb"

4. Configuration

The configuration file "C:\WINDOWS\system32\R_C_S.ini" is loaded and stored into the attribute this.prefs, if the file "C:\WINDOWS\system32\RCS.ini" doesn't exist.

If "C:\WINDOWS\system32\RCS.ini" does exist, it will read the configuration details and store them into "this.prefs". After that, Rurktar deletes the "RCS.ini" file. All configuration details are listed below.

Configuration	Functionality
Debug	If enabled, a logfile “RCS.log” gets written to the hard drive.
Port	The port which the malware connects to
IP	The IP which the malware connects to
FriendlyID	Default return value which is being used if no UUID could be enumerated.
CaptureMode	Not implemented yet
CaptureStart	Not implemented yet
CaptureMonikerString	Not implemented yet
ACaptureMonikerString	Not implemented yet
VideoCap	Not implemented yet
SkipFrames	Not implemented yet
SkipDetectionFrames	Not implemented yet
SkipTakeFrames	Not implemented yet
DetectionPreBuffer	Not implemented yet
NetworkImageQ	Sets the quality of the to be delivered image
CaptureDirectory	Checks whether a directory exists or not
DefPass	Not implemented yet
CaptureStopProcess1	Not implemented yet
CaptureStopProcess2	Not implemented yet
DetectPorog	Not implemented yet
MaxCaptureFrames	Not implemented yet
WatchFiles	Not implemented yet
AutoSendPreviews	Not implemented yet
SendOriginPreviews	Not implemented yet

CopyOriginsToCaptureDir	Not implemented yet
ControlExt	Not implemented yet
MaxCaptureFolderSize	Not implemented yet
WatchProc	Not implemented yet
ScreenshotAutoCapture	Not implemented yet
ScreenshotAutoStartProcess	Not implemented yet
ScreenshotExt	Sets the extension type for all screenshots
ScreenshotPause	Not implemented yet
ProxyEnabled	Not implemented yet

5. Persistence

The malware isn't placing a registry key into "HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run" like most of the malware out there does. Instead, it installs a new service called RCSU. This service will be started on a reboot just like any other service.

```
private void InitializeComponent()
{
    this.serviceProcessInstaller1 = new ServiceProcessInstaller();
    this.serviceInstaller1 = new ServiceInstaller();
    this.serviceProcessInstaller1.Account = ServiceAccount.LocalSystem;
    this.serviceProcessInstaller1.Password = null;
    this.serviceProcessInstaller1.Username = null;
    this.serviceInstaller1.Description = "RCSU";
    this.serviceInstaller1.DisplayName = "RCSU";
    this.serviceInstaller1.ServiceName = "RCSU";
    this.serviceInstaller1.StartType = ServiceStartMode.Automatic;
    base.Installers.AddRange(new Installer[]
    {
        this.serviceProcessInstaller1,
        this.serviceInstaller1
    });
}
```

Figure 1. Rurktar service installation

6. Commands

Below you can see a list of all possible commands the attacker might use, to execute actions at the victim computer.

Command	Functionality
Shares	Enumerates IPv4/IPv6 address, subnets and the default gateway.
Share	Enumerates all network shares available on the system.
Find	Enumerates files with the handover parameters directory, recursive search, filetype and filesize.
User	Enumerates Username, UserDomainName, MachineName and the OSVersion.
Ping	Determines whether a computer is accessible
Prefs	Gets the current preferences the malware is actively using.
Type	Not implemented
Receive	Saves a file to the disk
id	Enumerates the UUID
Ls	List hard disks and information about them (Name,type, label, format, available free space, total free space and the total size)
Get	Saves a large file to disk
Getfolder	Not implemented
Setparam	Sets a specific parameter to the configuration
Cmd	Executes a command via the command prompt
Proc	List current running processes on the computer
Kill	Terminates a running process by ID
Killname	Terminates a running process by name
Lsdev	Returns the DeviceName
Screenshot	Makes a screenshot from the current screen
Preview	Creates a preview image (500x500) of the current screen

Delete	Deletes a file
Delfolder	Deletes a folder

7. Communication

Below is a table of the communication format from the commands.

Command	Server/Client communication
Shares	Returns Mac address, IPAddr, IPv6addr, IPSubnet, IPSubnet mask, DefaultIPGateway, DNSServerSearchOrder as String
Share	Returns the error message “Share Cant resolve”
Find	Returns “find Cant find” + error message, if the file is not found.
User	Returns user?user*USERNAME*DOMAIN*USERINTERACTIVE*MACHINENAME*OSVERSION
Ping	Returns “ping Ping start error” if the command fails and “ping PingStarted” if the command was successful
Prefs	Should return the preferences, returns nothing at all instead.
Type	Returns “type?type*abstract”
Receive	Server: sends “file fileiwanttohave.txt” Client: returns “recieveCannot save file” if there was an error. Returns “recieve FILE SAVED”+ FILENAME if it was successful.
id	Server: sends “id” Client: returns “id?id*”+UUID”?FriendlyID*” + Prefs.FriendlyID

Ls	Server: sends “ls;root” Client: returns “ls;root?disc*”+LABEL+”*”+DRIVEFORMAT+”*”+AVAILABLEFREESPACE+”*”+TOTALFREESPACE+”*”+TOTALSIZE+”?disc… If it was successful. If there is an error “ls;root Cannot read drives” +ERRORMESSAGE is shown. Server: sends “ls;C:\” or “ls;D:\” etc Client returns with all files and folders in the format “ls;C:\?dir*”+SOMEFOLDER+”*”+”file*”+SOMEFILE…
Get	Server: sends “get;”+FILENAME Client: returns “get;”+FILENAME+”Cannot read file” if the file wasn’t found. Client: sends the file, if it was found.
Getfolder	Server: sends “getfolder” Client: returns “getfolderCannot read file” (Method not implemented)
Setparam	Server: sends “setparam” Client: sends “setparamCannot set param” Server: sends “setparam;Debug;true” Client: sends “setparam;Debug;true OK”
Cmd	Server: sends “cmd” Client: sends “cmdCannot start cmd” Server: sends “cmd;calc.exe” Client: sends “cmd?cmd*out cmd?cmd*err cmd;calc.exe Ok”
Proc	Server: sends “proc” Client: sends “proc?proc*”+PROCESSNAME+”*”+PROCESSID+”?proc….
Kill	Server: sends “kill;”+PID Client: sends “kill;”+PID+PROCESSNAME”+”killed” Server: sends “kill;” Client: sends “killCannot get list of processes”

Killname	Server: sends “killname;”+PROCESSNAME Client: sends “killname;”+PROCESSNAME+PROCESSNAME+” killed” Server: sends “killname;” Client: sends “killname; process not finded”
Lsdev	Enumarates the DeviceName
Screenshot	Makes a screenshot from the current screen
Preview	Creates a preview image (500x500) of the current screen
Delete	Server:sends “delete;”+FILENAME Client:sends “delete;”+FILENAME+ “DELETED” Server:sends “delete;” Client: sends “delete; Cannot delete”
Delfolder	Server: sends “delfolder;” Client: sends “delfolder; Cannot delete” Server: sends “delfolder;”+FOLDERNAME Client:sends “delfolder;”+FOLDERNAME+” DELETED”

8. Evolution of Rurktar

The first submission of Rurktar[1] was on the 2017-02-13 20:36:11, while the first submission of Rurktar[2] was on the 2017-06-11 19:12:52 UTC. Both samples were submitted on VirusTotal. Now this doesn’t specifically say, that the Rurktar malware is evolving, but when decompiling the projects with Dnspy and comparing the two project structures with WinMerge, it becomes clear, that parts of the code were changed. The older submission had 3 useless commands(fuu, fu2, fu3) which were a copy of the “delfolder” command. Those cases were taken out of the switch-case statement, as well as their functionality.

Figure 2. Rurktar file comparison

Snow

The malicious application called snow[4] is a wrapper to the Rurktar[3] malware. It checks if the current user has admin privileges or not. Then it stores the Rurktar malware on the disk and uses a persistence technique, so that it will get started when the computer is rebooted in future.

9. Evolution of Rurktar within Snow

Snow has stored Rurktar in its resources. When grabbing Rurktar from snow, it becomes clear, that the Rurktar malware has further “developed”. An additional case “fuck” has been added. The main difference of Case 25 (fuck) and the “delfolder” command, are the error messages – the functionality stays the same.

```
default:
```

```
case 25:
    try
    {
        Directory.Delete(array[1], true);
        text += " DELEfghfgjTED";
    }
    catch (Exception ex23)
    {
        text = text + " Cannot deletefghgfhfgh " + ex23.Message
    }
    this.SendAnswer(text, new byte[0]);
    goto IL_2006;
default:
```

Figure 3. Extracted Rurktar from Snow(Case25)

10. Getting administrator privileges

So Snow.exe checks if the current application is run with administration privileges. If it is, it will execute the main part of dropping Rurktar. Otherwise, it will try to execute a new process of itself which is asking the user to execute the application as admin.

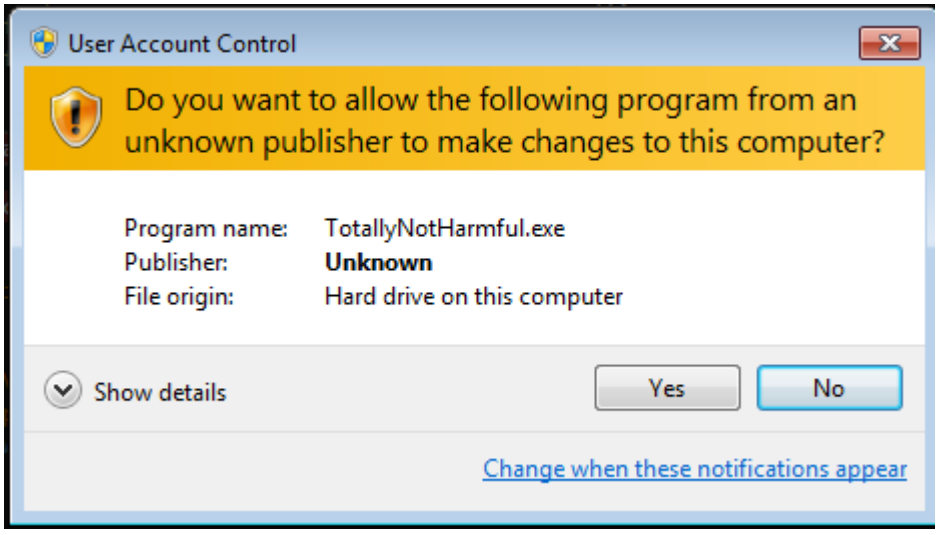

Figure 4. Snow.exe asks for admin privileges

11. Persistence

At first, Snow checks whether the registry key “SOFTWARE\Microsoft\NET Framework Setup\NDP\v3.5” exists. If it doesn’t, the directory string is set to “C:\Windows\Microsoft.NET\Framework\v3.5\”. If the key does exist, the “InstallPath” value from the “SOFTWARE\Microsoft\NET Framework Setup\NDP\v3.5” key gets used as an installation directory for the malware.

In both cases, the string “RCS\\” will get appended to the directory string.

Depending on the previous directory string, snow will now either use the “InstallPath” value + “RCS\\” as directory or use “C:\Windows\Microsoft.NET\Framework\v3.5\RCS” for further actions.

Case 1 (folder exists):

net.exe gets started with the arguments “stop rcsu”, to stop the current rcsu service.

If net.exe has finished, sc.exe gets started with the arguments “delete rcsu”, to delete the service.

If sc.exe has finished, the file RCSU.exe and iu.exe(InstallUtil) in the folder are getting deleted.

After the cleanup phase, the new RCSU.exe and iu.exe are written from the Snow file resources to the folder. Once this process has finished, iu.exe installs RCSU.exe as a service and net.exe gets executed with the parameters “start rcsu”, to run the service.

Case 2 (folder doesn’t exist):

The folder will be created and RCSU.exe and iu.exe will be read from the resources and written to the folder. Then, the same procedure as above will get executed.

After the procedure, a message is getting shown in figure 5. This is a hint, that the Snow malware is made by Russians.

```
catch (Exception)
{
}
MessageBox.Show("Невозможно инициализировать DirectX", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Hand);
base.Close();
```

Figure 5. Snow.exe error message

12. Conclusion:

The Rurktar malware is yet not been found that often, but has the potential to be more popular in the coming months because of the amount of options an attacker has with this malware. Right now, most of the attacker IP's (80.78.251.138, 80.78.251.148) come from Russia since the malware seem to be developed by a Russian because the exception messages within the malware are mostly in Russian. But as always, people in other countries can get the malware in their hands, so more diversity in attacker countries IP's will be seen.

13. Additional YARA rule for Rurktar

```
rule Rurktar
{
        meta:
                author = "Nathan Stern"
                description = "Rurktar detection rule"

        strings:
    $a = "FriendlyID" wide ascii nocase
    $a2 = "CaptureMode" wide ascii nocase
    $a3 = "CaptureStart" wide ascii nocase
    $a4 = "CaptureMonikerString" wide ascii nocase
    $a5 = "ACaptureMonikerString" wide ascii nocase
    $a6 = "VideoCap" wide ascii nocase
    $a7 = "SkipFrames" wide ascii nocase
    $a8 = "SkipDetectionFrames" wide ascii nocase
    $a9 = "SkipTakeFrames" wide ascii nocase
    $a10 = "DetectionPreBuffer" wide ascii nocase
    $a11 = "MaxCaptureFrames" wide ascii nocase
    $a12 = "MaxCaptureFolderSize" wide ascii nocase
    $a13 = "NetworkImageQ" wide ascii nocase
    $a14 = "CaptureDirectory" wide ascii nocase
    $a15 = "DefPass" wide ascii nocase
    $a16 = "CaptureStopProcess1" wide ascii nocase
    $a17 = "CaptureStopProcess2" wide ascii nocase
```

```
        $a18 = "DetectPorog" wide ascii nocase
        $a19 = "WatchFiles" wide ascii nocase
        $a20 = "AutoSendPreviews" wide ascii nocase
        $a21 = "ControlExt" wide ascii nocase
        $a22 = "SendOriginPreviews" wide ascii nocase
        $a23 = "CopyOriginsToCaptureDir" wide ascii nocase
        $a24 = "WatchProc" wide ascii nocase
        $a25 = "ScreenshotExt" wide ascii nocase
        $a26 = "ScreenshotAutoCapture" wide ascii nocase
        $a27 = "ScreenshotAutoStartProcess" wide ascii nocase
        $a28 = "ScreenshotPause" wide ascii nocase
        $a29 = "ProxyEnabled" wide ascii nocase
        $b = "\\R_C_S.ini" wide ascii nocase
        $b2 = "\\RCS.ini" wide ascii nocase
        $b3 = "RCS.log" wide ascii nocase
        $b4 = "RCSU.exe" wide ascii nocase
       $b5 = "RCS.log" wide ascii nocase
        $c = "?share*" wide ascii nocase
        $c2 = "?find*" wide ascii nocase
        $c3 = "user?" wide ascii nocase
        $c4 = "?prefs*" wide ascii nocase
        $c5 = "?type*abstract" wide ascii nocase
        $c6 = "?FriendlyID*" wide ascii nocase
        $c7 = "?disc*" wide ascii nocase
            condition:
                    5 of ($a*) or
                    3 of ($b*) or
                    4 of ($c*)
}
```

File hashes:

[1] MSIL.Backdoor.Rurktar.A
b4b75bda475ea58f2a5cf3329e311a70fa56745ba6cb2785523fa53139d4e37f

[2] MSIL.Backdoor.Rurktar.A
54f25a6820b8a0e3fc26bdf4599e7db695ef7aefb7dcefaa2c2581bb58426a40

[3] MSIL.Backdoor.Rurktar.A
89110710eddd0da23ea206a6047c252bf1e16a2d1957729973d77a58219e614b

[4] MSIL.Backdoor.Rurktar.A
618908e3d368301a323ee8ae7df867db8d7f5a98b513cfb8c961fb945e62a9b6