Threat Thursday: BoratRAT

blogs.blackberry.com/en/2022/04/threat-thursday-boratrat

The BlackBerry Research & Intelligence Team

While the name “BoratRAT” might bring a certain comedian to mind, this threat is nothing to laugh at. BoratRAT is an all-in-one malware toolkit that is capable of a variety of destructive activities, including acting as a ransomware, and performing credential theft.

BoratRAT was first analyzed by researchers at Cyble, which gave us our first deep dive into this malware’s full range of features. In researching this threat further, we discovered some intriguing connections between BoratRAT, SantaRAT, and AsyncRAT.

Operating System

Windows	MacOS	Linux	Android
Yes	No	No	No

Risk & Impact

Impact	High
Risk	Low

Technical Analysis

BoratRAT's name is taken from a satirical and fictional character ("Borat Sagdiyev") played by comedian Sacha Baron Cohen. The malware author promotes this Trojan on the web as the "#1 Remote Administration Tool" and uses Cohen's likeness in its icon.

BoratRAT was first seen by researchers on March 31, 2022. This new remote access trojan (RAT) proudly promotes itself as having a large feature set including the following:

- Direct Denial of Service (DDoS) attacks
- Ransomware capabilities
- Keylogging and data exfiltration
- Peripheral device manipulation (cameras, monitors, etc.)
- User Account Control (UAC) Bypass and Windows Defender deactivation
- Audio and Webcam Recording
- Browser credential and password theft
- Discord Token theft
- File system manipulation
- Remote Desktop capabilities

In the spirit of brevity, this post will analyze just two modules packaged with BoratRAT: keylogger.exe and information.dll. We will examine the modules' functionality and compare them with a similar older threat, SantaRAT. There, we will aim to find a major set of commonalities between the two RATs.

BoratRAT Breakdown

BoratRAT is initially packaged as a zip file named Borat.zip (SHA256 D2ce3aa530ba6b6680759b79aa691260244ca91f5031aa9670248924cc983fb0). Inside the zip file is the BoratRAT.exe executable, a config file for BoratRAT, a server certificate, and a bin directory. All the feature-supporting modules are stored in the bin directory of the zip. For our purposes here, we will be focusing on the following three executables:

BoratRAT.exe
(b47c77d237243747a51dd02d836444ba067cf6cc4b8b3344e5cf791f5f41d20e)

Keylogger.exe
(16beb1ae2de2974ccc2371d9f619f492295e590abb65d3102e362c8ec27f2bbb)

Information.dll
(a15d72d990686d06d89d7e11df2b16bcd5719a40298c19d046fa22c40d56af44)

Running the BoratRAT.exe starts the user interface for the malware operator. From here, a threat actor can conduct all the malware's capabilities on their target machine, as shown in Figure 1. The malware author packed BoratRAT.exe with Enigma Protector for obfuscation, to make it more difficult to analyze.

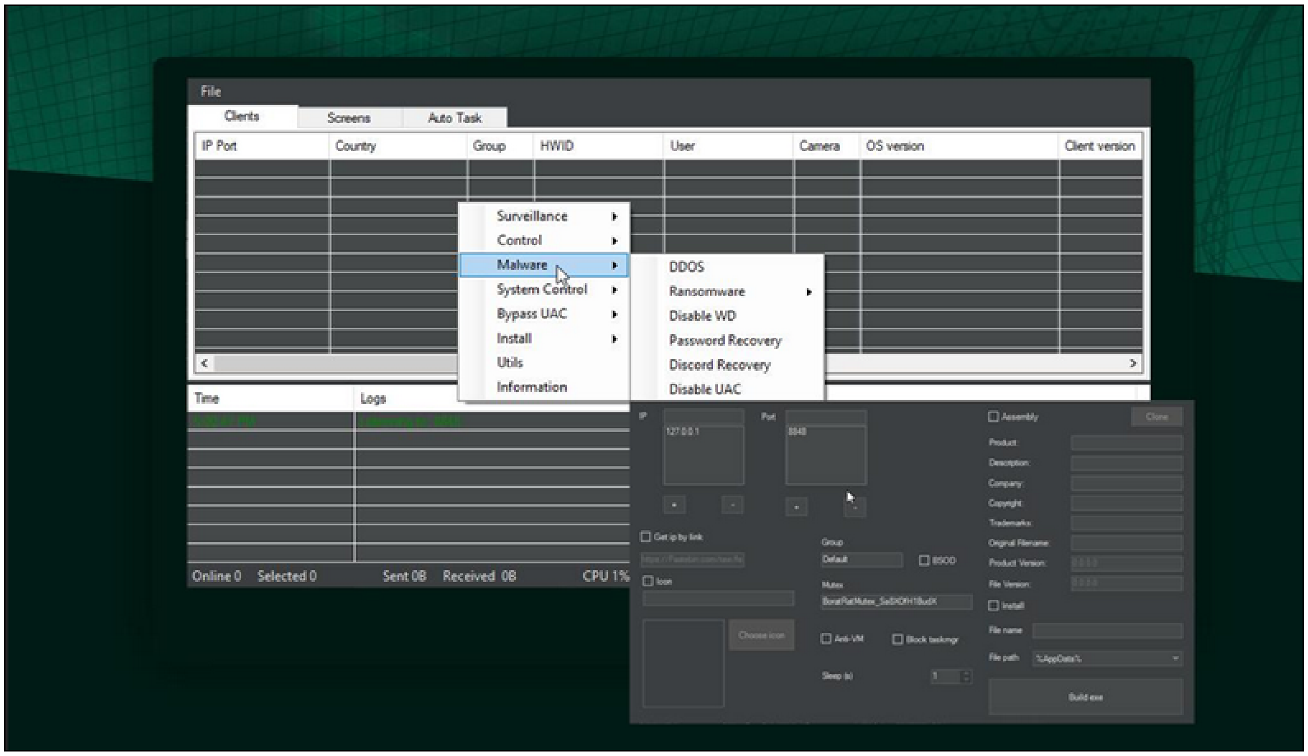
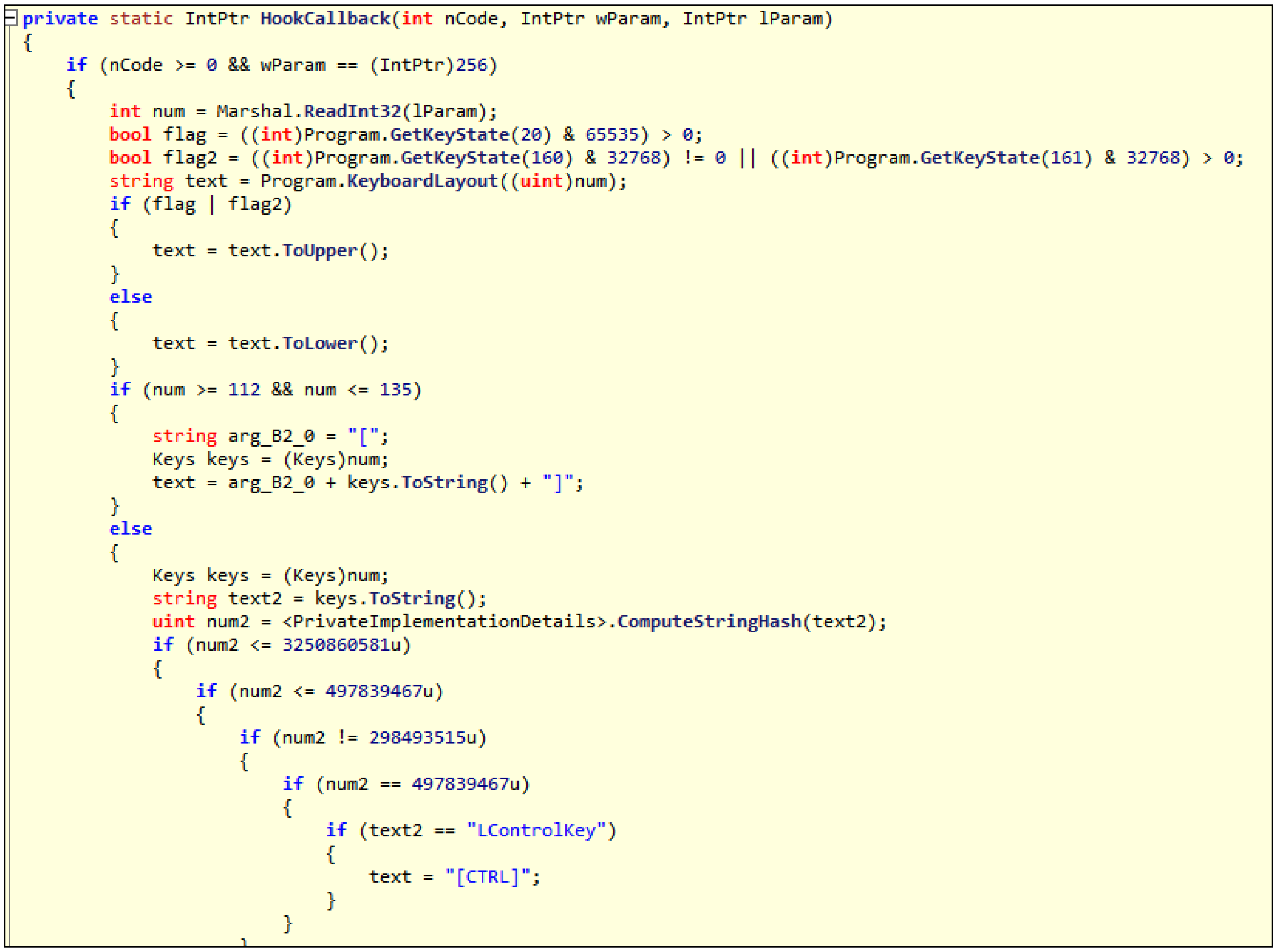


Figure 1 – BoratRAT UI snippet

The keylogger.exe application module is called through BoratRAT's user interface (UI). Keylogger.exe does exactly as the filename describes – it logs the victim's keystrokes and saves the input to a file. The keylogging application starts with a main function, which passes a low-level keyboard procedure to a function called SetHook. SetHook receives the return value from HookCallback as shown in Figure 2 below, which uses a combination of GetKeyState and a decision structure to get key status.



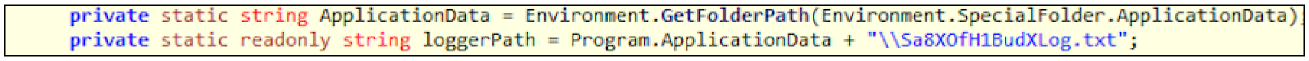
```
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
{
    if (nCode >= 0 && wParam == (IntPtr)256)
    {
        int num = Marshal.ReadInt32(lParam);
        bool flag = ((int)Program.GetKeyState(20) & 65535) > 0;
        bool flag2 = ((int)Program.GetKeyState(160) & 32768) != 0 || ((int)Program.GetKeyState(161) & 32768) > 0;
        string text = Program.KeyboardLayout((uint)num);
        if (flag | flag2)
        {
            text = text.ToUpper();
        }
        else
        {
            text = text.ToLower();
        }
        if (num >= 112 && num <= 135)
        {
            string arg_B2_0 = "[";
            Keys keys = (Keys)num;
            text = arg_B2_0 + keys.ToString() + "]";
        }
        else
        {
            Keys keys = (Keys)num;
            string text2 = keys.ToString();
            uint num2 = <PrivateImplementationDetails>.ComputeStringHash(text2);
            if (num2 <= 3250860581u)
            {
                if (num2 <= 497839467u)
                {
                    if (num2 != 298493515u)
                    {
                        if (num2 == 497839467u)
                        {
                            if (text2 == "LControlKey")
                            {
                                text = "[CTRL]";
                            }
                        }
```

Figure 2 - HookCallBack function with keystroke decision tree

Once the key state is recognized, a keystroke is then written to a string stream with StreamWriter. StreamWriter is a library native to the C# programming language, which is used to write characters to a stream in UTF-8 encoding.

At this point in the malware's execution, a check is made with an IF ELSE statement against the GetActiveWindowTitle() function. Then the function makes a call to GetForegroundWindow() from winuser.h, to return a handle to the window that the victim is currently working in. This is done to properly label which active window the victim is in, as it writes keystrokes to the file.

The output from this activity is then written to a file indicated by the program.loggerPath variable. LoggerPath is a static string used by the program that concatenates the working environment of the program to the output file, to give it a full environment path. In this instance, the file is named "\\Sa8X0fH1BudXLog.txt" and is written out to the environment folder path specified by the malware, as shown in Figure 3.

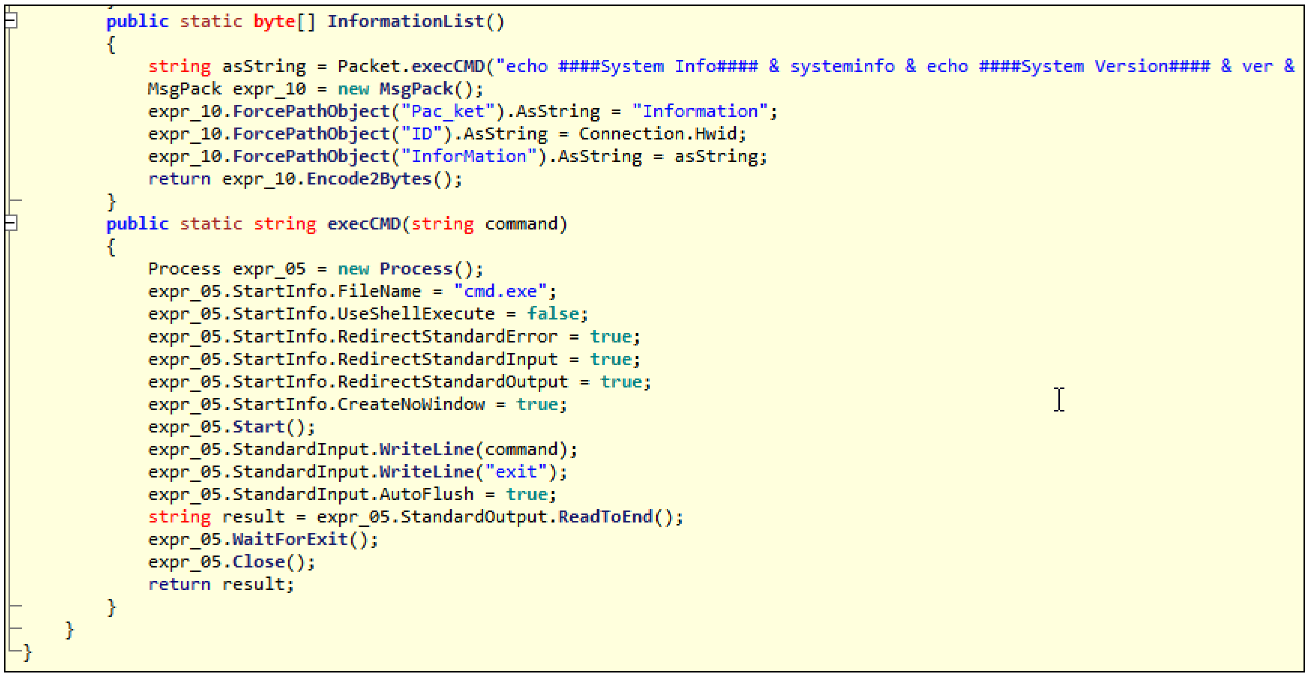


```
    private static string ApplicationData = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)
    private static readonly string loggerPath = Program.ApplicationData + "\\Sa8XOfH1BudXLog.txt";
```

Figure 3 – Text file "Sa8XofH1BudXLog.txt" location in Keylogger.exe

The Information.dll module contains commands to exfiltrate system information to the attacker. The function execCMD starts a command prompt with the "cmd.exe" application, which is launched in CreateNoWindow mode with standard output, input and error redirected. This redirection is done to hide the execution of the command prompt from the desktop.

At that point, the InformationList() function calls execCMD(). A command string is passed that writes a long stream of system information out to the command line, as seen in Figure 4. The information is then packeted and sent back to the threat actor. The string of echo commands is a full dump of environment, OS version, hardware, hosts, user/administrator information, and network information.



```
        public static byte[] InformationList()
        {
            string asString = Packet.execCMD("echo ####System Info#### & systeminfo & echo ####System Version#### & ver &
            MsgPack expr_10 = new MsgPack();
            expr_10.ForcePathObject("Pac_ket").AsString = "Information";
            expr_10.ForcePathObject("ID").AsString = Connection.Hwid;
            expr_10.ForcePathObject("InforMation").AsString = asString;
            return expr_10.Encode2Bytes();
        }
        public static string execCMD(string command)
        {
            Process expr_05 = new Process();
            expr_05.StartInfo.FileName = "cmd.exe";
            expr_05.StartInfo.UseShellExecute = false;
            expr_05.StartInfo.RedirectStandardError = true;
            expr_05.StartInfo.RedirectStandardInput = true;
            expr_05.StartInfo.RedirectStandardOutput = true;
            expr_05.StartInfo.CreateNoWindow = true;
            expr_05.Start();
            expr_05.StandardInput.WriteLine(command);
            expr_05.StandardInput.WriteLine("exit");
            expr_05.StandardInput.AutoFlush = true;
            string result = expr_05.StandardOutput.ReadToEnd();
            expr_05.WaitForExit();
            expr_05.Close();
            return result;
        }
    }
}
```

Figure 4 – Information.dll "InformationList" and "execCMD" functions

It is likely that the initial inspiration for BoratRAT was an earlier threat called AsyncRAT, which first appeared two years ago. Evidence supporting this origin was found by BleepingComputer, which pointed out similarities between these two threats. This assertion has also been backed up through string analysis using the Intezer Analyze tool.

This is not the only RAT that was inspired by this threat – SantaRAT was also built on AsyncRAT, and it could have been another starting point for BoratRAT's development. In fact, on the main page of the GitHub repository for SantaRAT, the owner gives a personal "thank you" to the author of the AsyncRAT library.

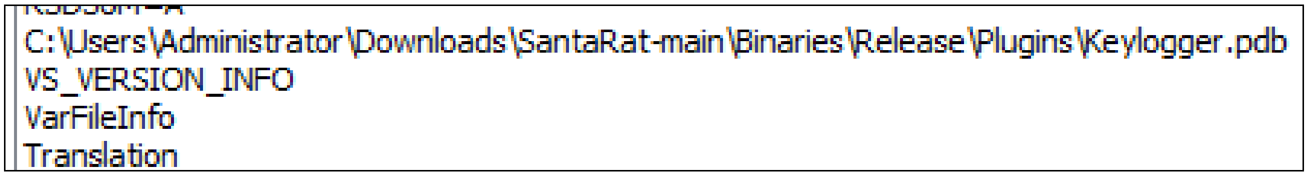
From the starting point of AsyncRAT's functionality, SantaRAT added modules like audio recording and file system modifications. But SantaRAT's last change was made five months ago. BoratRAT began with a feature set that is closer to that of SantaRAT.

The two modules we will analyze in this section are just a fraction of this RAT's capabilities, to show a representative example of the links between the SantaRAT source code and BoratRAT's module code.

On a high level, the similarities are quickly apparent. For instance, both threats are written in C#. They have a very similar feature set; they are both capable of ransoming a victim, stealing credentials, and obtaining remote access, to name but a few examples.

When we get down to the modules, the similarities continue. Whenever source code files (legitimate or otherwise) are compiled, each module contains a directory path that references a program database file. Program database files are used to store debugging information about a compiled source code file.

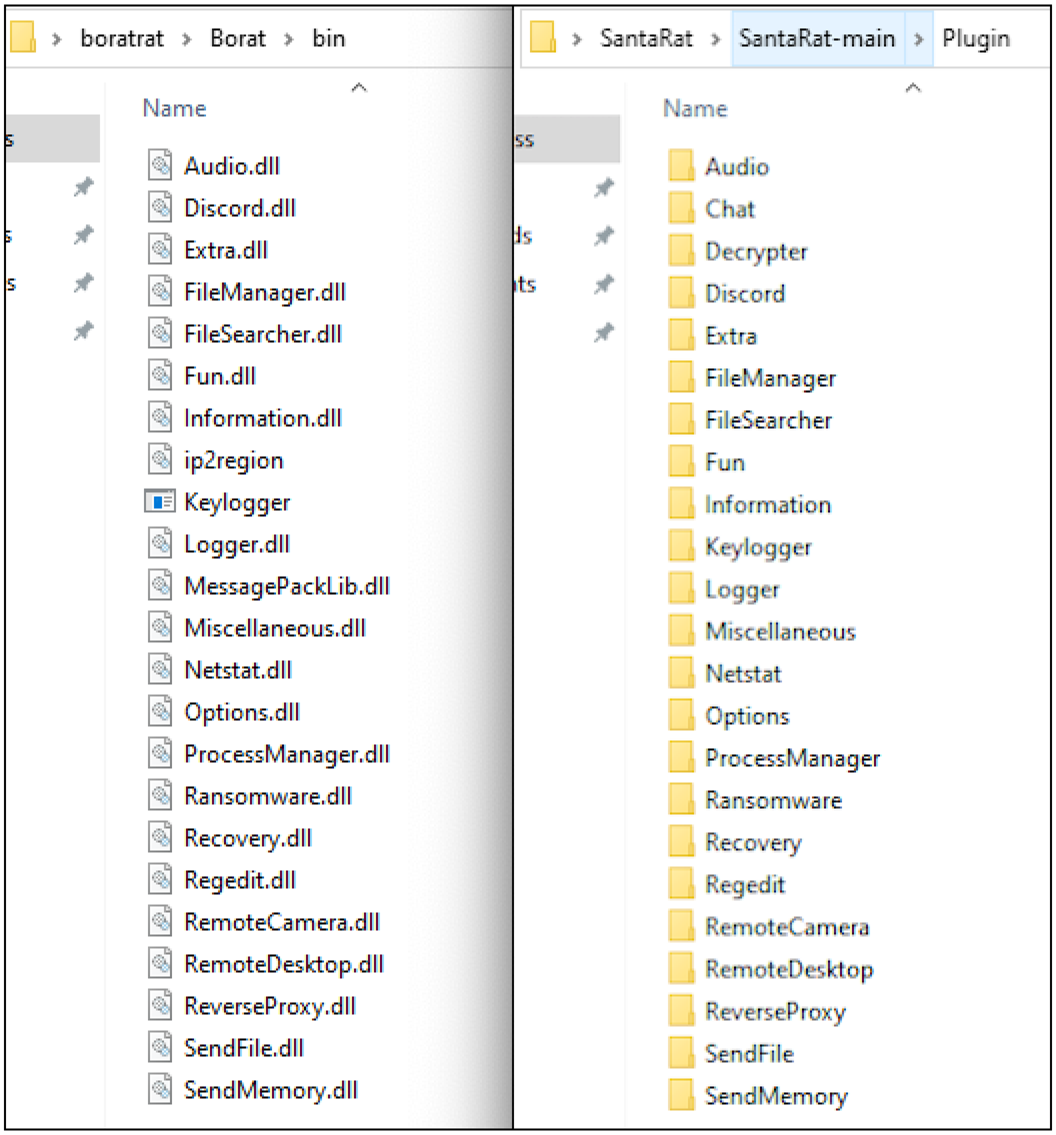
In BoratRAT's path, we see a reference to SantaRAT, which is shown in Figure 5. This shows that SantaRAT modules were downloaded, compiled, and were in use in the same environment. This pathway with "SantaRat-main" is seen in BoratRAT's information.dll library as well.



```
C:\Users\Administrator\Downloads\SantaRat-main\Binaries\Release\Plugins\Keylogger.pdb
VS_VERSION_INFO
VarFileInfo
Translation
```

Figure 5 – SantaRAT-main library path string in BoratRAT's packaged Keylogger.exe

Beyond that, the structured code and module names also nearly mirror each other. With the exclusion of a few modules in SantaRAT, the module packages are nearly identical when compared side by side, as seen below in Figure 6. The missing directories are Chat, Decryptor, and messagepacklib. This could be due to the author refactoring some code functionality into similar libraries.


boratrat › Borat › bin

Name

Audio.dll
Discord.dll
Extra.dll
FileManager.dll
FileSearcher.dll
Fun.dll
Information.dll
ip2region
Keylogger
Logger.dll
MessagePackLib.dll
Miscellaneous.dll
Netstat.dll
Options.dll
ProcessManager.dll
Ransomware.dll
Recovery.dll
Regedit.dll
RemoteCamera.dll
RemoteDesktop.dll
ReverseProxy.dll
SendFile.dll
SendMemory.dll

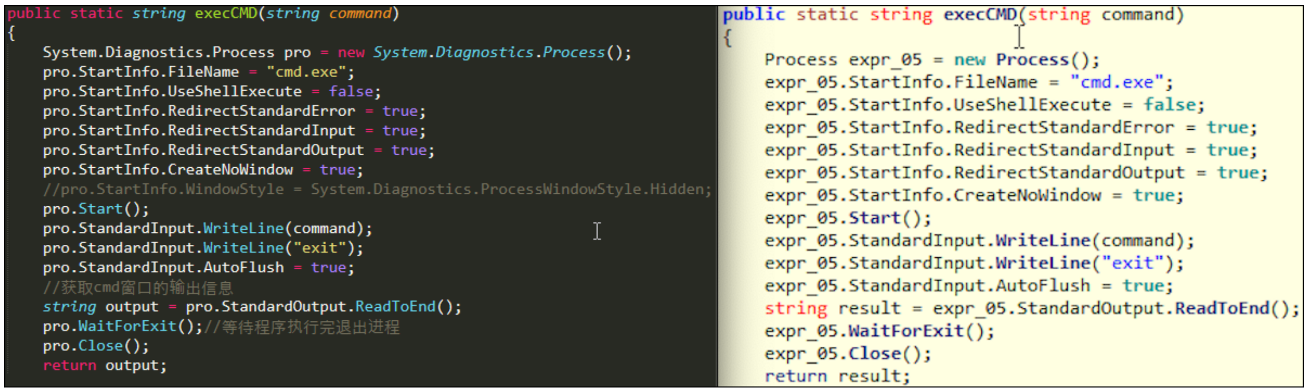
SantaRat › SantaRat-main › Plugin

Name

Audio
Chat
Decrypter
Discord
Extra
FileManager
FileSearcher
Fun
Information
Keylogger
Logger
Miscellaneous
Netstat
Options
ProcessManager
Ransomware
Recovery
Regedit
RemoteCamera
RemoteDesktop
ReverseProxy
SendFile
SendMemory

Figure 6 – BoratRAT bin and SantaRAT plugin

Almost all the module code from both RATs is identical. Information.dll shares the exact same code as packet.cs in the SantaRAT repository on GitHub. A snippet of the execCMD taken from SantaRAT and BoratRAT, seen in Figure 7, shows almost a line-for-line copy between the two. This type of code reuse is found in the keylogger.exe application and packaged in BoratRAT and program.cs from SantaRAT's repository as well.



```
public static string execCMD(string command)
{
    System.Diagnostics.Process pro = new System.Diagnostics.Process();
    pro.StartInfo.FileName = "cmd.exe";
    pro.StartInfo.UseShellExecute = false;
    pro.StartInfo.RedirectStandardError = true;
    pro.StartInfo.RedirectStandardInput = true;
    pro.StartInfo.RedirectStandardOutput = true;
    pro.StartInfo.CreateNoWindow = true;
    //pro.StartInfo.WindowStyle = System.Diagnostics.ProcessWindowStyle.Hidden;
    pro.Start();
    pro.StandardInput.WriteLine(command);
    pro.StandardInput.WriteLine("exit");
    pro.StandardInput.AutoFlush = true;
    //获取cmd窗口的输出信息
    string output = pro.StandardOutput.ReadToEnd();
    pro.WaitForExit();//等待程序执行完退出进程
    pro.Close();
    return output;
```

```
public static string execCMD(string command)
{
    Process expr_05 = new Process();
    expr_05.StartInfo.FileName = "cmd.exe";
    expr_05.StartInfo.UseShellExecute = false;
    expr_05.StartInfo.RedirectStandardError = true;
    expr_05.StartInfo.RedirectStandardInput = true;
    expr_05.StartInfo.RedirectStandardOutput = true;
    expr_05.StartInfo.CreateNoWindow = true;
    expr_05.Start();
    expr_05.StandardInput.WriteLine(command);
    expr_05.StandardInput.WriteLine("exit");
    expr_05.StandardInput.AutoFlush = true;
    string result = expr_05.StandardOutput.ReadToEnd();
    expr_05.WaitForExit();
    expr_05.Close();
    return result;
```

Figure 7 – SantaRAT execCMD function (left) BoratRAT execCMD function (right)

Conclusion

It's clear from the supporting analysis that the malware author of BoratRAT used SantaRAT as a development reference. In every module we analyzed there is obvious code reuse. Some of the modules have been exact line-for-line copies.

So, what does this mean? We think that BoratRAT's code reuse could indicate a continuation of development and rebranding of SantaRAT. It is entirely possible that a new author could take over the evolution of this malware, or it could be that BoratRAT is meant to be yet another short-lived development. Time will tell where this threat is headed.

Despite BoratRAT's comical name, the versatility of this RAT makes it a dangerous threat that both private individuals and companies should be wary of. The impact of BoratRAT obtaining entry to a system (either an individual or an organization) could be catastrophic, and proper security practices should be observed to help mitigate threats such as this.

YARA Rules

The following YARA rule was authored by the BlackBerry Research & Intelligence team to catch the threat described in this document:

```
rule BoratRAT{

    meta:

        description = "Detects BoratRAT.exe"

        author = "BlackBerry Threat Research Team"

        date = "2022-04-13"

        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"
```

```
    strings:

        $s1 = "enigma1"

        $s2 = "enigma2"

        $s3 = "Server.Forms.FormFileManager.resources"

        $s4 = "Server.Forms.FormFileSearcher.resources"

        $s5 = "Server.Forms.FormKeylogger.resources"

        $s6 = "Server.Forms.FormNetstat.resources"

        $s7 = "Server.Forms.FormFun.resources"

        $s8 = "Server.Forms.FormWebcam.resources"

        $s9 = "BoratRat"

        $s10 = "Keylogger.exe"

    condition:

        uint16(0) == 0x5a4d and all of them

}

rule BoratRATKeylogger{

    meta:

        description = "Detects BoratRAT Keylogger"

        author = "BlackBerry Threat Research Team"

        date = "2022-04-13"

        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:

        $s1 = "Sa8XOfH1BudXLog.txt" wide

        $s2 = "[CAPSLOCK: ON]" wide

        $s3 = "[CAPSLOCK: OFF]" wide

        $s4 = "[SPACE]" wide

        $s5 = "[ENTER]" wide
        $sp = { 43 3A 5C 55 73 65 72 73 5C 41 64 6D 69 6E 69 73 74 72 61 74 6F 72 5C

            44 6F 77 6E 6C 6F 61 64 73 5C 53 61 6E 74 61 52 61 74 2D 6D 61 69 6E

            5C 42 69 6E 61 72 69 65 73 5C 52 65 6C 65 61 73 65 5C 50 6C 75 67 69

            6E 73 5C 4B 65 79 6C 6F 67 67 65 72 2E 70 64 62 }

    condition:

        uint16(0) == 0x5a4d and all of them

}

rule BoratRATInformation{

    meta:

        description = "Detects BoratRAT Information Module"

        author = "BlackBerry Threat Research Team"

        date = "2022-04-13"

        license = "This Yara rule is provided under the Apache License 2.0
(https://www.apache.org/licenses/LICENSE-2.0) and open to any user or organization, as
long as you use it under this license and ensure originator credit in any derivative to The
BlackBerry Research & Intelligence Team"

    strings:

        $s1 = "set_UseShellExecute"

        $s2 = "execCMD"

        $s3 = "hostname" wide

        $s4 = "ipconfig" wide

        $s5 = "tasklist" wide

        $s6 = "arp -a" wide

        $sp = {43 3A 5C 55 73 65 72 73 5C 41 64 6D 69 6E 69

            73 74 72 61 74 6F 72 5C 44 6F 77 6E 6C 6F 61

            64 73 5C 53 61 6E 74 61 52 61 74 2D 6D 61 69

            6E 5C 42 69 6E 61 72 69 65 73 5C 52 65 6C 65

            61 73 65 5C 50 6C 75 67 69 6E 73 5C 49 6E 66

            6F 72 6D 61 74 69 6F 6E 2E 70 64 62}
    condition:

        uint16(0) == 0x5a4d and all of them

}
```

Indicators of Compromise (IoCs)

b47c77d237243747a51dd02d836444ba067cf6cc4b8b3344e5cf791f5f41d20e – BoratRAT.exe

16beb1ae2de2974ccc2371d9f619f492295e590abb65d3102e362c8ec27f2bbb – Keylogger.exe

a15d72d990686d06d89d7e11df2b16bcd5719a40298c19d046fa22c40d56af44 – Information.dll

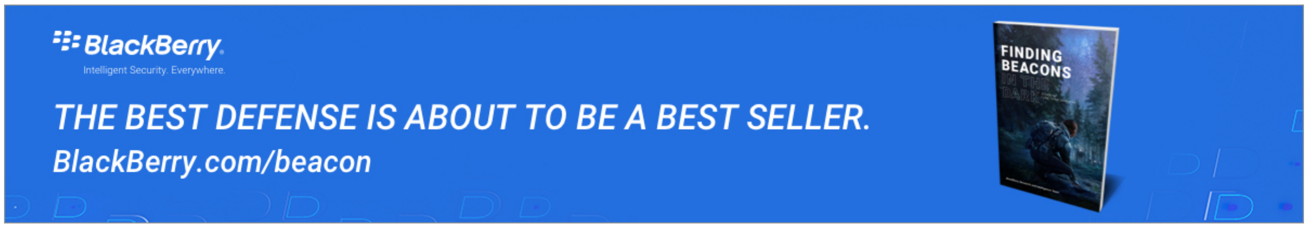
D2ce3aa530ba6b6680759b79aa691260244ca91f5031aa9670248924cc983fb0 – Borat.zip

BlackBerry Assistance

If you're battling this malware or a similar threat, you've come to the right place, regardless of your existing BlackBerry relationship.

The BlackBerry Incident Response team is made up of world-class consultants dedicated to handling response and containment services for a wide range of incidents, including ransomware and Advanced Persistent Threat (APT) cases.

We have a global consulting team standing by to assist you, providing around-the-clock support where required, as well as local assistance. Please contact us **here**.


BlackBerry
Intelligent Security. Everywhere.
THE BEST DEFENSE IS ABOUT TO BE A BEST SELLER.
BlackBerry.com/beacon
FINDING BEACONS

About The BlackBerry Research & Intelligence Team

The BlackBerry Research & Intelligence team examines emerging and persistent threats, providing intelligence analysis for the benefit of defenders and the organizations they serve.

Back