火绒安全
HUORONG SECURITY

火绒“脚本行为沙盒”简介

变形脚本病毒的“照妖镜”

2021/11/29

公　　司：北京火绒网络科技有限公司

地　　址：北京市朝阳区红军营南路 15 号瑞普大厦 D 座 4 层

网　　址：https://www.huorong.cn

电　　话：400-998-3555

版权声明

本文件所有内容受中国著作权法等有关知识产权法保护，为北京火绒网络科技有限公司（以下简称“火绒安全”）所有，任何个人、机构未经“火绒安全”书面授权许可，均不得通过任何方式引用、复制。另外，“火绒安全”拥有随时修改本文件内容的权利。

如有修改，恕不另行通知。您可以咨询火绒官方、代理商等售后，获得最新文件。

目录

前言

近年来下载者病毒逐渐从传统的 PE 类病毒向脚本类病毒演变，脚本类病毒与 PE 类相比在一些方面上存在优势。首先脚本类病毒在文件大小上明显小于 PE 类病毒，混淆成本远低于 PE 类病毒，混淆手法更为多变，并且能够实现 PE 病毒绝大部分的功能。这类脚本病毒的批量制造，对传统安全软件提出了不小的挑战。

下载者病毒所需要的功能简单、单一，还要求病毒大小不能过大，方便网络传播。而脚本类病毒正好满足下载者病毒的要求。所以，近年来，脚本类下载者病毒呈现激增的趋势。

为了应对不断变化的病毒样本， 主流安全软件厂商引入了脚本的虚拟沙盒。与 PE 类似，”脚本虚拟沙盒”是通过仿真脚本运行时环境，使病毒代码认为运行真实系统中，进而还原其行为。通过病毒在虚拟沙盒中还原出来的原始代码和病毒执行时的一系列行为进行查毒。理论上只要仿真环境足够逼真，真实操作系统中能够执行的病毒，在虚拟沙盒中都能够成功执行并检测到其执行时的恶意行为。

目前国外主流安全软件都很重视这类问题，策略不同，各有优劣。火绒采用区别于主流安全软件的创新思路实现脚本虚拟行为沙盒，可以更好地解决这个难题。同时，基于这一思路的延伸，未来我们会为火绒反病毒引擎引入更多创新特性。

反病毒引擎的脚本虚拟沙盒技术

由于脚本病毒的混淆方法简单，混淆成本远低于 PE 病毒，通过搜索引擎就能够找到大量的在线混淆网站，能够简单的混淆出大量静态特征不同但功能相同的脚本。使得仅通过静态特征查杀变得越来越困难。例如 SVM:TrojanDownloader/JS.Nemucod.a 下载者病毒，如下图所示，仅从代码形式上来看，就有多种形式，仅单纯的阅读代码已经很难发现这些病毒其实执行的都是同样的功能，都是由一份代码通过不同的混淆手法批量生成出来的。

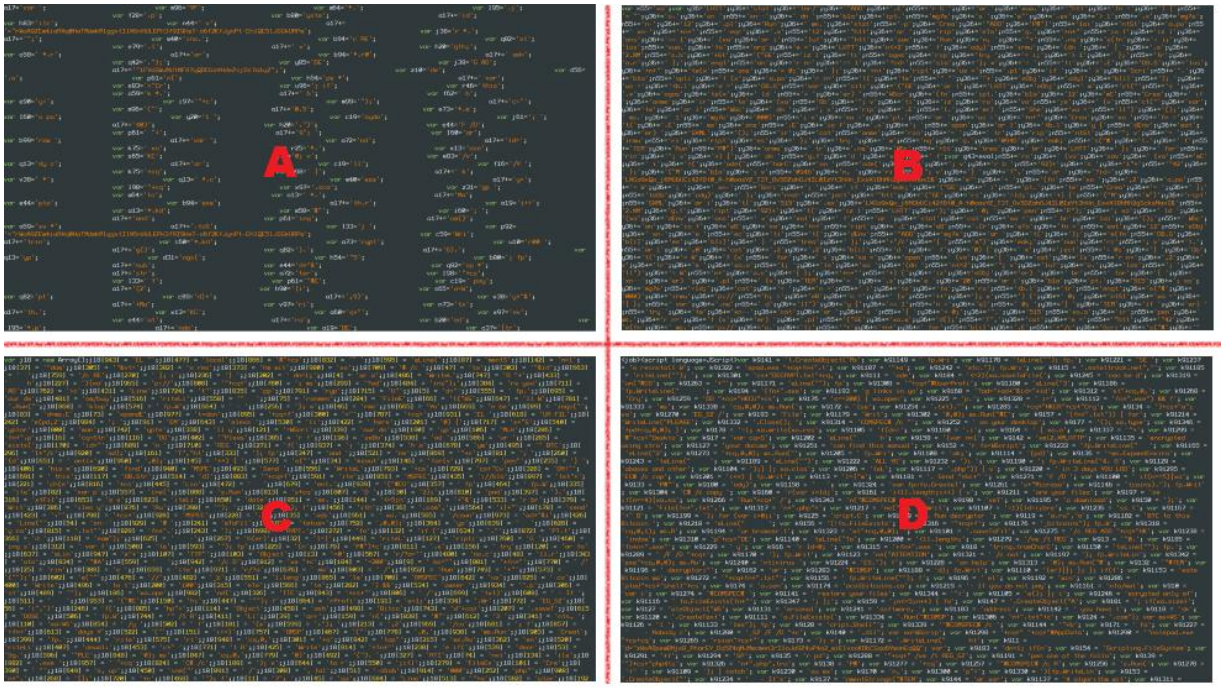

图 2-1、Nemucod 家族部分样本展示

然而通过动态虚拟执行，我们可以很容易看出他们虽然下载链接、生成的文件名并不完全相同，但行为模式几乎完全一样，如下图所示:

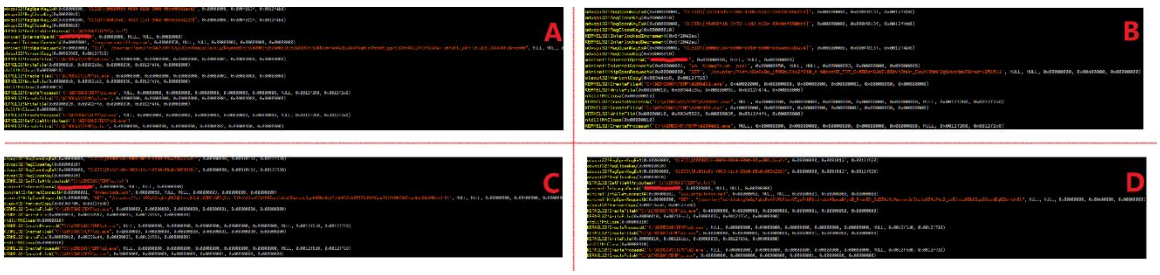

图 2-2、Nemucod 家族动态虚拟执行

主流安全软件的脚本虚拟沙盒

同 PE 混淆类病毒一样，通过动态虚拟执行还原被混淆代码的原始数据和行为，才是解决问题之道。基于这样的思路，主流安全厂商都试图通过动态“执行”的方式来解决脚本混淆问题。其中主流的思路是在反病毒引擎中嵌入脚本解释器来分析并执行脚本代码，进而通过对脚本执行时需要的运行时组件进行有限的模拟来分析脚本的运行时行为。其架构大致如下图所示：

图 2-3、主流脚本虚拟沙盒

这类脚本虚拟机实现相对简单，且互联网上有大量的开源脚本引擎作为参考，能够快速的搭建一个相对有效的脚本虚拟机。但这类脚本虚拟机往往存在以下两个问题：

1. 脚本解释和模拟执行直接运行在真实系统环境中，甚至一些产品中直接应用开源脚本解释器来构建，所以这类脚本虚拟机存在被漏洞利用脚本“穿透”的潜在风险；
2. 病毒代码对虚拟环境仿真程度的”探测“不断升级（例如对文件系统的”探测”），使得这类脚本虚拟机在虚拟环境模拟方面捉襟见肘、疲于应付；

火绒脚本虚拟沙盒

基于上述原因，并考虑到火绒反病毒引擎现有架构，我们并没有采用上述基于脚本解释器的方式来实现脚本虚拟沙盒。我们的思路是将脚本问题转化为 PE 问题，进而复用反病毒引擎原有的基于 PE 病毒的行为检测逻辑。

下图为火绒脚本虚拟沙盒：

图 2-4、火绒虚拟沙盒

从图中可以看出，我们解决脚本问题的思路是，在现有火绒虚拟沙盒的基础上，实现脚本病毒依赖的运行时环境，并在此基础上实现 WScript.exe 来执行脚本。如下图所示：

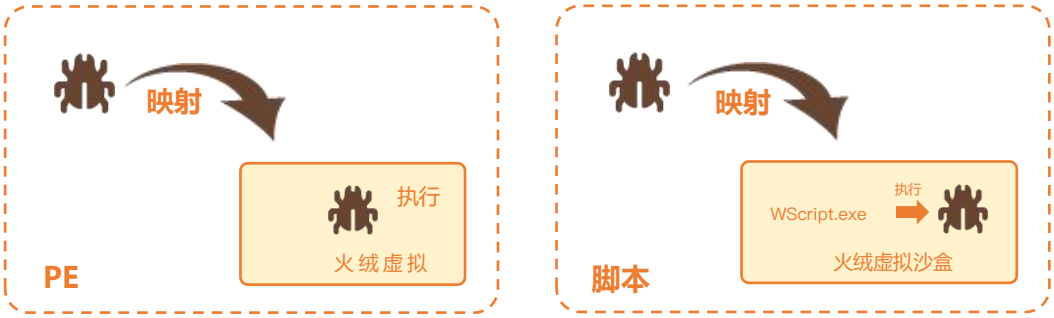

图 2-5、脚本执行流程

这样的架构和实现与前述架构的区别与优势在于：

1. 脚本的解释、执行完全被限制在虚拟沙盒中，执行过程中与真实系统环境完全隔离，完全不存在被漏洞利用脚本“穿透”的风险；
2. 脚本运行时环境构建在火绒虚拟 Windows 内核之上，仿真程度极高。由于一直以来火绒都比较依赖通过虚拟行为沙盒来解决 PE 混淆问题，所以我们对于虚拟沙盒的环境仿真一直处于积极完善的过程之中，不但实现了完善的进程、线程调度、文件系统、注册表，甚至完全模拟了 Windows 窗口系统。在此基础上，只需要实现脚本解释组件、相关运行时组件以及脚本宿主程序（WScript.exe）即可。基于这样的架构，可以有效地应对病毒代码对虚拟环境的”探测“；
3. 基于这样的架构，反病毒引擎不需要任何修改就可以完全复用基于虚拟行为沙盒的病毒行为检测逻辑；
4. 处理脚本和 PE 代码联动的问题。例如，TrojanDropper/JS.Basecode.a 病毒家族会将真正实现恶意行为的 PE 病毒加密打包，外层脚本只负责释放和执行 PE 病毒。这类脚本在火绒反病毒引擎的行为扫描过程中可以完全还原从脚本释放 PE 到 PE 病毒执行完的全部流程，所以可以有效并准确地识别该病毒家族。该家族病毒在火绒虚拟沙盒中执行流程如下图所示：

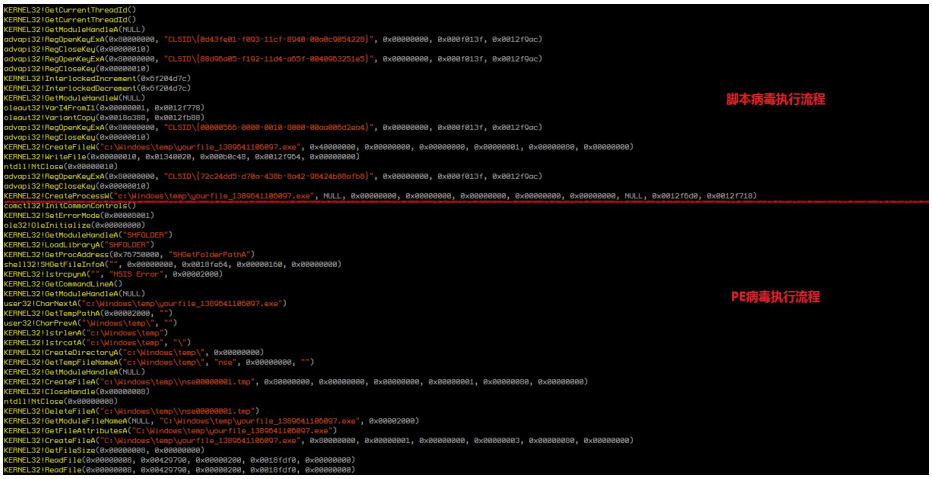

图 2-6、释放并执行 PE 病毒

脚本病毒对抗手段分析

经过我们长期的跟踪和观察，脚本病毒为了对抗安全软件的反混淆技术，主要从以下两方面来检测安全软件的虚拟沙盒：

1. 语言层面检测。
2. 系统环境的检测。

说到脚本病毒，就不得不提近年来十分活跃的 Nemucod 下载者病毒，该病毒使用微软的 JScript 编写，能够直接运行在 Windows 操作系统之上，通过 COM 接口调用系统功能，能够比在浏览器中运行的 JavaScript 脚本实现多的多功能。该家族病毒近年来保持着高速的升级更新，反反混淆手段不断变换。从中我们可以看到该家族病毒两种检测方法都有使用，下面我们就对其使用的检测手段进行简要介绍。

语言层面检测

/*@cc_on @*/ 这句语句是微软的 JScript 所支持的特有语法，这个语句中包含的语句会被执行，并开启条件编译，但这个语句只有 IE 和 WScript 中支持。如下图所示的病毒代码中，我们可以看出病毒使用了上述语句，如果使用开源脚本引擎，就会将/**/中的语句认为时注释而不执行，就会因为没有创建对象而出错。

```
function anonymous(vFb3) {
    {
        eval("/*@cc_on var vNp6 = new Date() @*/");
        vNp6.setUTCSeconds("0", "20");
        if (vNp6.getUTCMilliseconds().toString(10) == "20") {
            return vFb3.replace(/\$/g, "");
        }
        else
            return "";
    }
}
```

图 3-1、使用特殊语法检测

setUTCFullYear 函数时 JavaScript 语言中的一个内置方法，用于设置年份，在病毒中

也被常用于检测虚拟沙箱。如下图代码所示，病毒首先将年份设置为 2003 年，并在随后获取年份，来检查是否时刚才所设置的年份，如果发现不是则不执行解密流程。

```
function anonymous(v_s) {
    {
        var v_d = new Date();
        v_d["setUTC" + "FullYear"]("2003");
        if (v_d.getUTCFullYear().toString(10) == "2003") {
            var v_arr = v_s.split("?");
            return v_arr.join("");
        } else
            return "";
    }
}
```

图 3-2、使用 setUTCFullYear 设置年份

下图中的代码我们可以看到病毒通过使用 try…catch 语句捕获异常进行异常处理，在第一个 try…catch 语句中 new ActiveXObject 语句需要正确执行，而第二个 try…catch 由于 new ActiveXObject 语句参数错误，所以运行时会抛出异常，这时异常被捕获就会执行异常处理中的语句，只有这样才能正确的拼接出需要的字符串。

```
la = "e";
try {
    var y = new ActiveXObject("VBScript.RegExp");
    la = la + "v";
} catch (x) {
    str3 = 4;
    la = "rr";
};
try {
    var y = new ActiveXObject("");
    str3 = 1;
} catch (x) {
    la = la + "al";
    str3 = str3 - 1;
};
for (i = 0; i < str4.length; i += str3) {
    str2 += str4[i];
}
var str5 = la;
trip = this;
trip[str5](str2);
```

图 3-3、通过异常处理检测虚拟沙盒

系统环境检测

根据我们对病毒的长期的跟踪发现，随着安全软件与病毒的攻防不断升级，近一年来，脚本病毒的混淆器已经不再是简单的通过上文中我们所介绍的方法仅仅是检测脚本引擎的功能是否完整，而是利用 JScript 可以调用操作系统的功能，来调用操作系统的一些功能来检测自己的运行环境。

观察下面两幅图中的代码，我们发现病毒通过枚举目录，来获取目录名称的各种属性来检测系统的文件系统。

```
function anonymous() {
    var inkezs3 = new Enumerator(ybdetof5.GetFolder('C:\\').SubFolders);
    if (inkezs3.item(0).name.length > 1)
        return true;
    else
        return false;
}
```

图 3-4、枚举子目录检测目录名长度

```
function anonymous() {
    var ivtavvo = rwunnynpix.GetFolder('C:\\Windows').SubFolders;
    if (typeof screen == 'undefined')
        var ropkewelvi = new Enumerator(ivtavvo);
    if (typeof ropkewelvi.item(0).Type.length == 'number')
        return true;
    else
        return false;
}
```

图 3-5、枚举子目录检测目录名长度类型

下图中是我们截取自一个病毒样本中检测虚拟沙盒的代码，通过对源代码进行整理后我们可以看到，该病毒会获取 hosts 文件的文件属性，来检测是否是在真实系统中。

```
var a = WScript["CreateObject"]("Scripting.FileSystemObject");
var b = a["GetFile"]("C:\\Windows\\System32\\drivers\\etc\\hosts")["Attributes"];
if (b == "32")
```

图 3-6、检测系统文件属性

下图中的样本用到了 WScript.exe 中的内置方法，通过调用 Arguments 方法获取启动参数，如果参数不对，则重新启动 WScript.exe。如果执行脚本的不是 WScprit.exe，脚本执行就会出错。

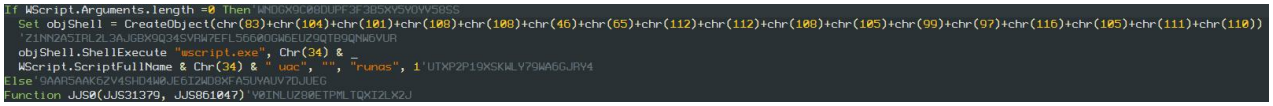

图 3-7、检测 WScript 参数

附录

文中涉及样本 SHA1：

图 2-1 中样本	659e95158d195f8503a113a1e13e974ab909f2bc 6cf8e5846c92d5c0fd3b67a3a5adf13818f5e958 c96af51b0ab696ba312c22b56bac3943c8b71f2f 65fa8571b7bf6953c7c1047168c46a993790c8cb
TrojanDropper/JS.Basecode.a	43acb3990a60b093ab1ff9dd550bad409d390a0e 2247866a150ac2c19631ae547beb2e4eff08fa53 158cb79495f3417cc1be1c10a4d79bc29c901b1f 47f81cbf4cc3486c93fd0977e62952b3a1dfd981 82da1550ac6932993c66393e4e154efc9455c5db
Nemucod 家族	c546892b7a31417d3b1f3109d28d7ab4ddfe8386 0c5aa535c25a5f983bd81d072725a56fd11b8059 f1f40bbc9392f07f4ebf4345fba9e58fe7ea8451 e7ec5c0d738bb2c2f89274d94b593830bf6cc65c
图 3-1 中样本	e5e0eb4de413923a38ee5a610c5131b019d26232
图 3-2 中样本	ef40c7edc13b2ca80bf5aab57fd8ab0e933c8f57
图 3-3 中样本	214284982fc70b94b8ef14bb661dee2fb282acac
图 3-4 中样本	e338ae640263c1d019d5e1e1bf52d40e1401d9e3
图 3-5 中样本	3ef189ab8eac4cb37ec76a7b1b092993d2754473
图 3-6 中样本	e8d40d610fd3d117a27de8de069689c6b8a10518
图 3-7 中样本	479a828efa13cde80baf4729ee53707a8e273dba