HawkEye Credential Theft Malware Distributed in Recent Phishing Campaign

fireeye.com/blog/threat-research/2017/07/hawkeye-malware-distributed-in-phishing-campaign.html

Threat Research Blog

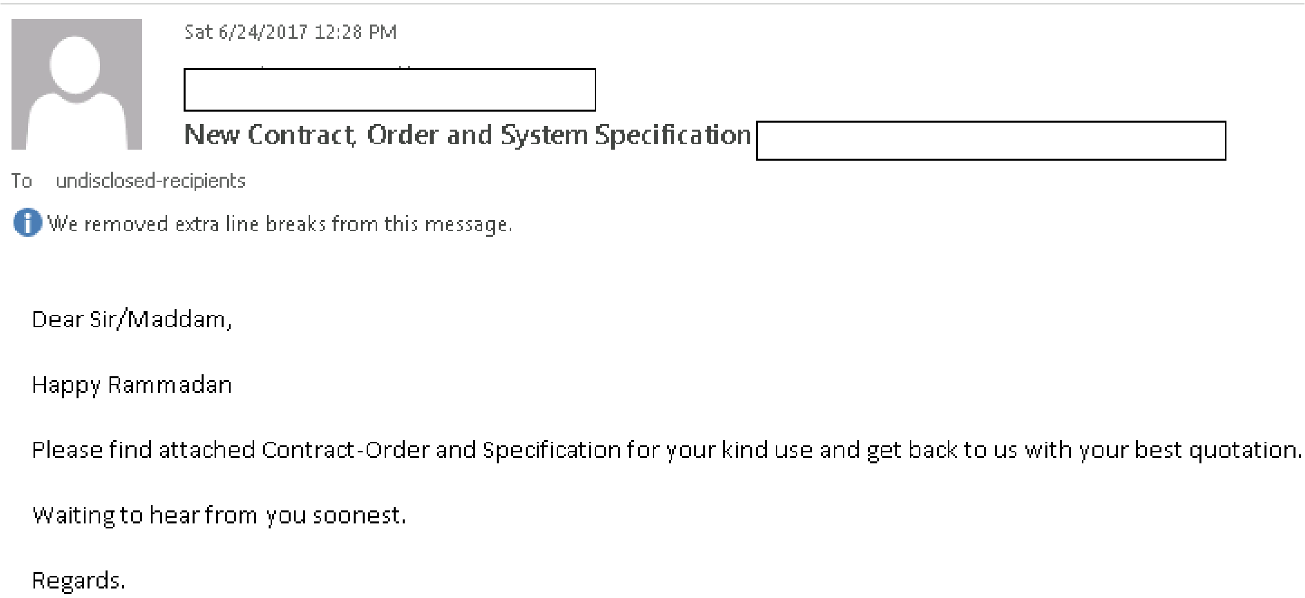
July 25, 2017 | by Swapnil Patil, Yogesh Londhe
Malware
Phishing

A wide variety of threat actors began distributing HawkEye malware through high-volume email campaigns after it became available for purchase via a public-facing website. The actors behind the phishing campaigns typically used email themes based on current events and media reports that would pique user interests, with the “Subject” line typically containing something about recent news. Although HawkEye malware has several different capabilities, it is most often associated with credential theft.

In the middle of June, we observed a phishing campaign involving the distribution HawkEye malware. The threat actors behind this campaign are not targeting any specific group of industries or any specific region.

Infection Vector & Execution

Figure 1 shows a sample phishing email used by HawkEye operators in this latest campaign. The message is designed to entice recipients to open the attachment. In this most recent campaign, the phishing email contained a DOCX attachment, and the attackers named the document appropriately so the recipient believed it involved a recent transaction or invoice.



Sat 6/24/2017 12:28 PM

[redacted]

New Contract, Order and System Specification [redacted]

To undisclosed-recipients

ⓘ We removed extra line breaks from this message.

Dear Sir/Maddam,

Happy Rammadan

Please find attached Contract-Order and Specification for your kind use and get back to us with your best quotation.

Waiting to hear from you soonest.

Regards.

Figure 1: Sample phishing email

As seen in Figure 2, the deployment of the malware has several stages of execution, including the following:

1. Phishing email containing a malicious DOCX file received by victim.
2. DOCX file uses an OLE object, which contains an embedded Microsoft Intermediate Language (MSIL) executable. The MSIL file, or HawkEye malware, is dropped into the %temp% folder. The malware has an encrypted resource section, which contains additional payloads such as a password extraction tool and a decoy PDF document.
3. On execution, HawkEye drops copies itself to the %AppData% folder with a random file name.
4. The decoy PDF file is launched from the %temp% location.
5. An XML file is created in the %temp% folder with a random file name. This XML file contains configuration details for scheduling a Windows task to execute during the user login process.
6. For the sample analyzed, the malware is injected into VBC.exe (a Visual Basic Command Line Compiler). The injected code has data stealing capabilities and is designed to extract passwords from email clients and web browsers.

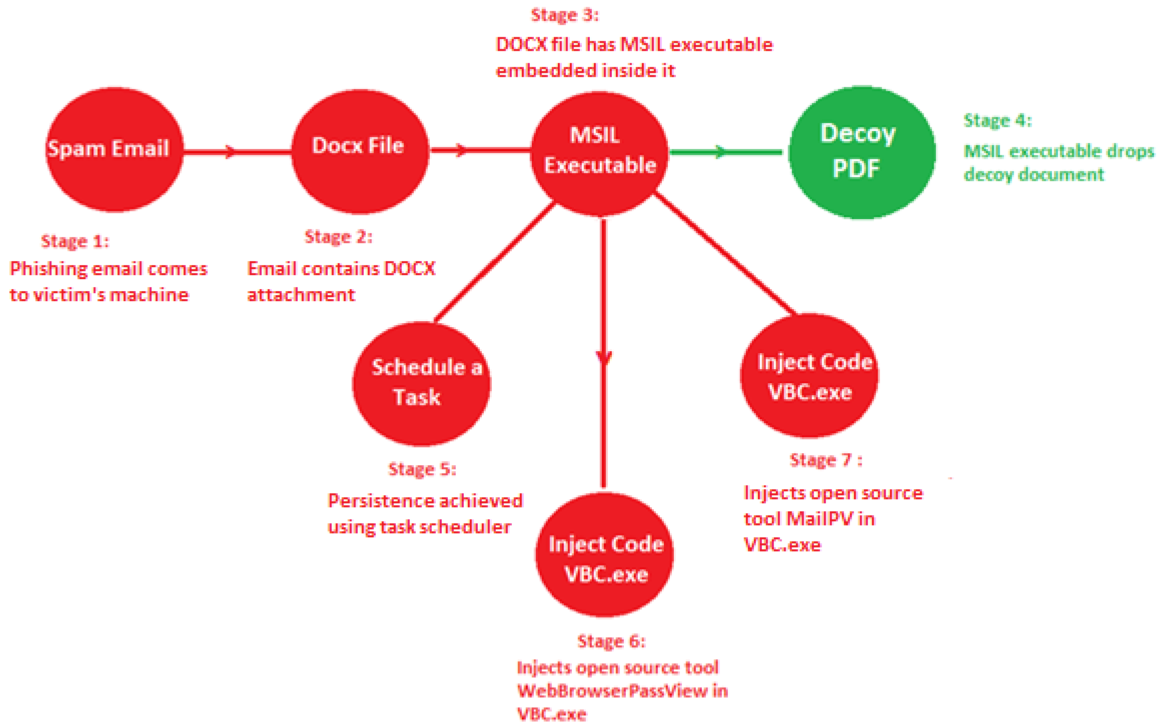
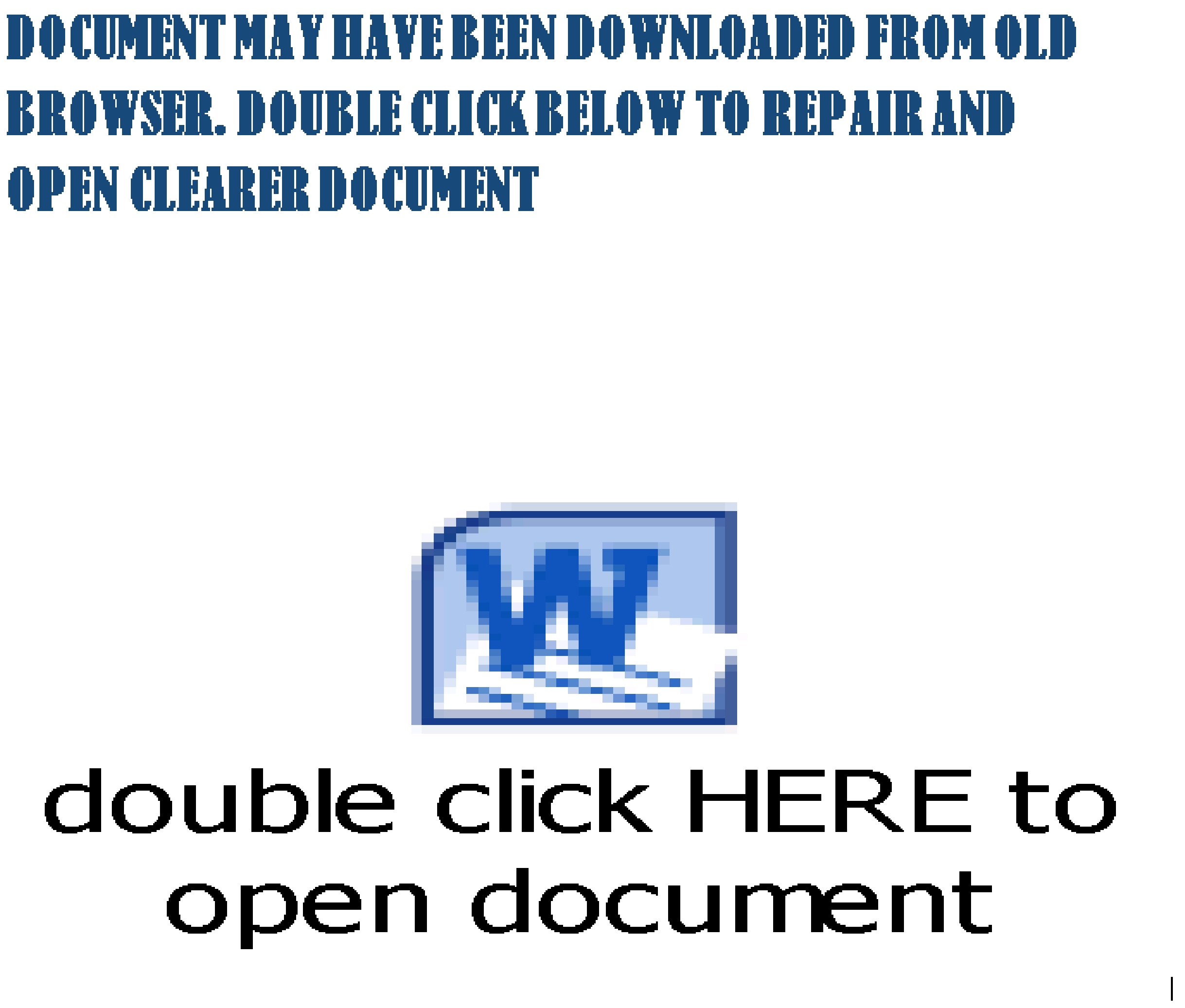

Stage 3:
DOCX file has MSIL executable embedded inside it
Spam Email
Docx File
MSIL Executable
Decoy PDF
Stage 4:
MSIL executable drops decoy document
Stage 1:
Phishing email comes to victim's machine
Stage 2:
Email contains DOCX attachment
Schedule a Task
Inject Code VBC.exe
Stage 5:
Persistence achieved using task scheduler
Stage 7 :
Injects open source tool MailPV in VBC.exe
Inject Code VBC.exe
Stage 6:
Injects open source tool WebBrowserPassView in VBC.exe

Figure 2: Infection Vector and Execution

Initial Payload: DOCX File

In the observed campaign, the actors used an embedded OLE object to deliver the payload to the victim’s machine. The malicious payload, HawkEye, is embedded in the DOCX file and dropped in the %temp% folder after the victim double-clicks on the object (Figure 3).



DOCUMENT MAY HAVE BEEN DOWNLOADED FROM OLD BROWSER. DOUBLE CLICK BELOW TO REPAIR AND OPEN CLEARER DOCUMENT

double click HERE to open document

Figure 3: Embedded OLE Object

HawkEye Analysis

The HawkEye malware is primarily used for credential theft and is often combined with additional tools to extract passwords from email and web browser applications. These additional tools are contained in an encrypted resource section of the binary.

The HawkEye malware is capable of the following:

1. Email password stealing
2. Web browser password stealing
3. Keylogging and taking screenshots
4. Bitcoin wallet theft
5. USB propagation
6. Internet download manager stealing
7. JDownloader password stealing

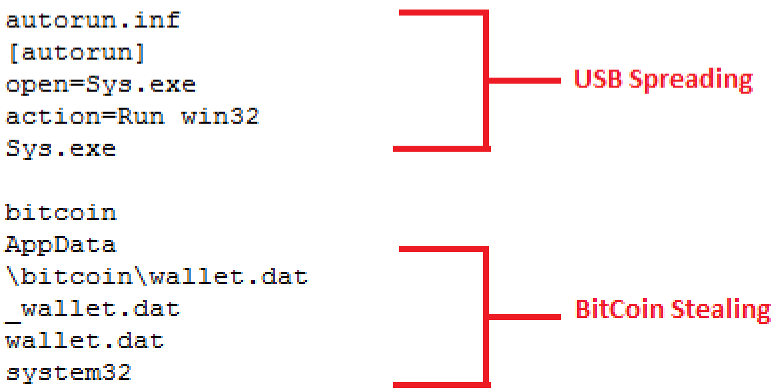
8. Anti-virus checking
9. Firewall checking

After initial checks and system enumeration, HawkEye sends the following data to the command and control (C2) server:

- Server Name
- Keylogger Enabled
- Clipboard-Logger Enabled
- Stealers Enabled
- Local Date and Time
- Installed Language
- Operating System
- Internal IP Address
- External IP Address
- Installed Anti-Virus
- Installed Firewall

USB Propagation and Bitcoin Wallet Theft

Along with its ability to steal sensitive information, HawkEye is capable of spreading through USB or removeable drives and can also steal Bitcoin wallets, as seen in Figure 4.

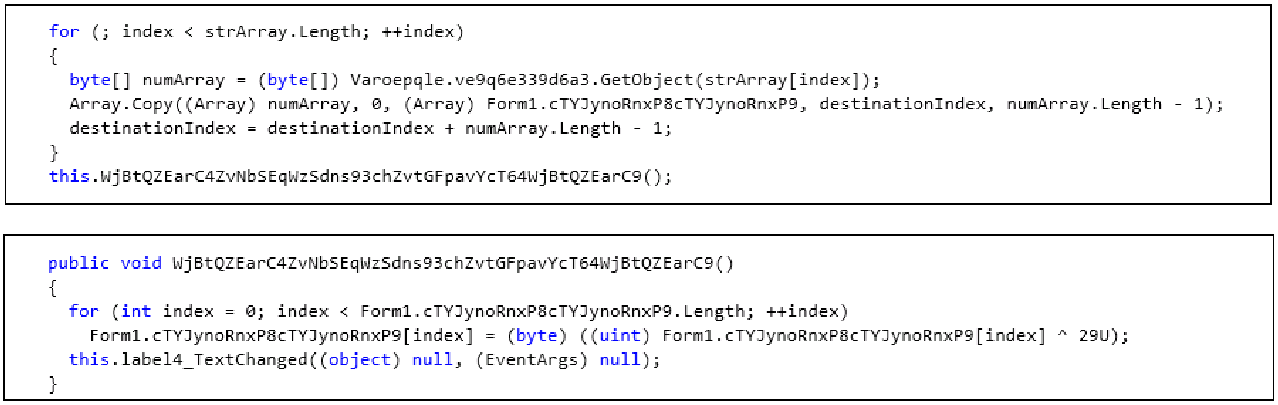


```
autorun.inf
[autorun]
open=Sys.exe
action=Run win32
Sys.exe

bitcoin
AppData
\bitcoin\wallet.dat
_wallet.dat
wallet.dat
system32
```

Figure 4 : USB spreading and Bitcoin Stealing

Encrypted Resource Section

The HawkEye malware in this campaign contained encrypted resources sections, which add functionality that enables the attackers to exfiltrate more data. FireEye observed the same pattern in previous HawkEye campaigns. The encrypted data is decrypted at run time and then injected in to the target process, vbc.exe. The encryption logic used is a custom algorithm and varies with the campaign. Figure 5 shows an example of the custom encryption algorithm.



```
for (; index < strArray.Length; ++index)
{
  byte[] numArray = (byte[]) Varoepqle.ve9q6e339d6a3.GetObject(strArray[index]);
  Array.Copy((Array) numArray, 0, (Array) Form1.cTYJynoRnxP8cTYJynoRnxP9, destinationIndex, numArray.Length - 1);
  destinationIndex = destinationIndex + numArray.Length - 1;
}
this.WjBtQZEarC4ZvNbSEqWzSdns93chZvtGFpavYcT64WjBtQZEarC9();
```

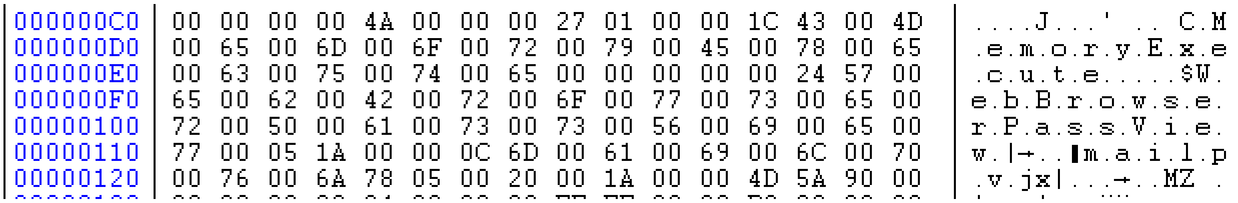
```
public void WjBtQZEarC4ZvNbSEqWzSdns93chZvtGFpavYcT64WjBtQZEarC9()
{
  for (int index = 0; index < Form1.cTYJynoRnxP8cTYJynoRnxP9.Length; ++index)
    Form1.cTYJynoRnxP8cTYJynoRnxP9[index] = (byte) ((uint) Form1.cTYJynoRnxP8cTYJynoRnxP9[index] ^ 29U);
  this.label4_TextChanged((object) null, (EventArgs) null);
}
```

Figure 5: Custom decryption routine

After decrypting the resource section, the following files can be extracted:

1. Decoy pdf file.
2. <Random_Name>.XML

   Contains configuration data for a Windows task creation
3. CMemoryExecute.dll
4. WebBrowserPassView.exe
5. MailPV.exe

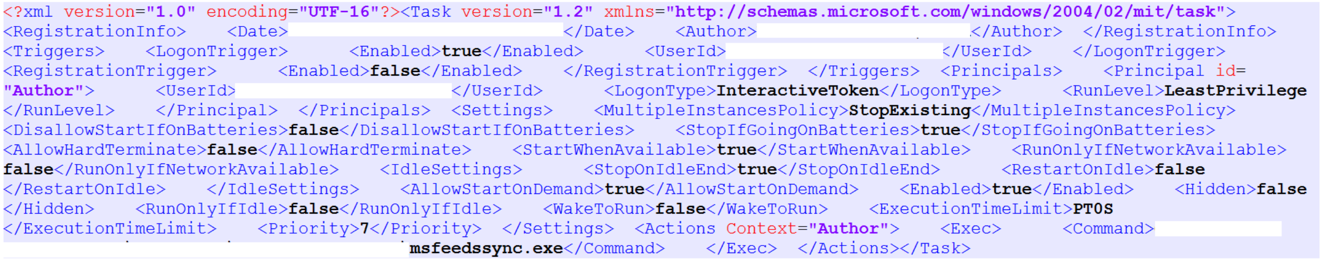


```
000000C0  00 00 00 00 4A 00 00 00 27 01 00 00 1C 43 00 4D  ....J...' .. C.M
000000D0  00 65 00 6D 00 6F 00 72 00 79 00 45 00 78 00 65  .e.m.o.r.y.E.x.e
000000E0  00 63 00 75 00 74 00 65 00 00 00 00 00 24 57 00  .c.u.t.e.....$W.
000000F0  65 00 62 00 42 00 72 00 6F 00 77 00 73 00 65 00  e.b.B.r.o.w.s.e.
00000100  72 00 50 00 61 00 73 00 73 00 56 00 69 00 65 00  r.P.a.s.s.V.i.e.
00000110  77 00 05 1A 00 00 0C 6D 00 61 00 69 00 6C 00 70  w.|→..▌m.a.i.l.p
00000120  00 76 00 6A 78 05 00 20 00 1A 00 00 4D 5A 90 00  .v.jx|...→..MZ .
```

Figure 6: Components of malware

Task Scheduler – Persistence Mechanism

The payload uses the Windows task scheduling feature for its persistance mechanism on the victim's computer. It schedules a task to execute on user login. The configuration data shown in Figure 7 is used to schedule the task.



```
<?xml version="1.0" encoding="UTF-16"?><Task version="1.2" xmlns="http://schemas.microsoft.com/windows/2004/02/mit/task">
<RegistrationInfo>    <Date>[redacted]</Date>    <Author>[redacted]</Author>  </RegistrationInfo>
<Triggers>    <LogonTrigger>      <Enabled>true</Enabled>      <UserId>[redacted]</UserId>    </LogonTrigger>
<RegistrationTrigger>      <Enabled>false</Enabled>    </RegistrationTrigger>  </Triggers>  <Principals>    <Principal id=
"Author">      <UserId>[redacted]</UserId>      <LogonType>InteractiveToken</LogonType>      <RunLevel>LeastPrivilege
</RunLevel>    </Principal>  </Principals>  <Settings>    <MultipleInstancesPolicy>StopExisting</MultipleInstancesPolicy>
<DisallowStartIfOnBatteries>false</DisallowStartIfOnBatteries>    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
<AllowHardTerminate>false</AllowHardTerminate>    <StartWhenAvailable>true</StartWhenAvailable>    <RunOnlyIfNetworkAvailable>
false</RunOnlyIfNetworkAvailable>    <IdleSettings>      <StopOnIdleEnd>true</StopOnIdleEnd>      <RestartOnIdle>false
</RestartOnIdle>    </IdleSettings>    <AllowStartOnDemand>true</AllowStartOnDemand>    <Enabled>true</Enabled>    <Hidden>false
</Hidden>    <RunOnlyIfIdle>false</RunOnlyIfIdle>    <WakeToRun>false</WakeToRun>    <ExecutionTimeLimit>PT0S
</ExecutionTimeLimit>    <Priority>7</Priority>  </Settings>  <Actions Context="Author">    <Exec>      <Command>[redacted]
msfeedssync.exe</Command>    </Exec>  </Actions></Task>
```

Figure 7: Task Scheduler.xml

CMemoryExecute.dll

CMemoryExecute.dll is responsible for running a .NET executable capable of using the Windows Native API to inject MailPV.exe and WebBrowserPassView.dll into VBC.exe, which the Visual Basic Command Line Compiler. MailPV and WebBrowserPassView are used in order to extract credentials from the list of email and web browser clients noted in the following section.

WebBrowserPassView

WebBrowserPassView.dll, extracted from the resource section, is a password recovery tool that extracts passwords stored in the following web browsers:

- Internet Explorer (Version 4.0 – 11.0)
- Mozilla Firefox (All Versions)
- Google Chrome
- Safari
- Opera

The extracted passwords are stored in a created text file: “%temp%\holderwb.txt”

MailPV

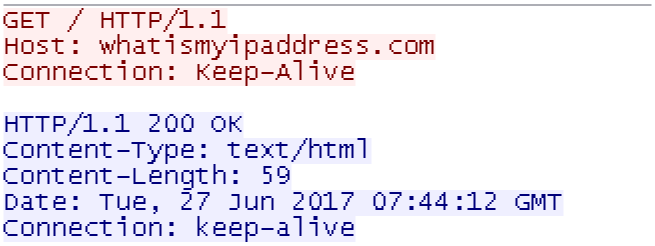
The MailPV.exe file is password recovery tool that extracts password for following email clients:

- Outlook Express
- IncrediMail
- Eudora
- Group Mail Free
- MS Outlook
- MS Outlook 2002/2003/2007/2010
- Gmail
- Hotmail/MSN
- Yahoo! Mail
- Netscape Mail
- Thunderbird
- Google Desktop
- Windows Mail
- Windows Live Mail
- Outlook 2013

The extracted passwords are stored in a created text file: “%temp%\holdermail.txt”

Command and Control Communications

The first C2 traffic observed is the malware’s check to get the external IP address of the infected machine. Figure 8 shows an example of the external IP address query.



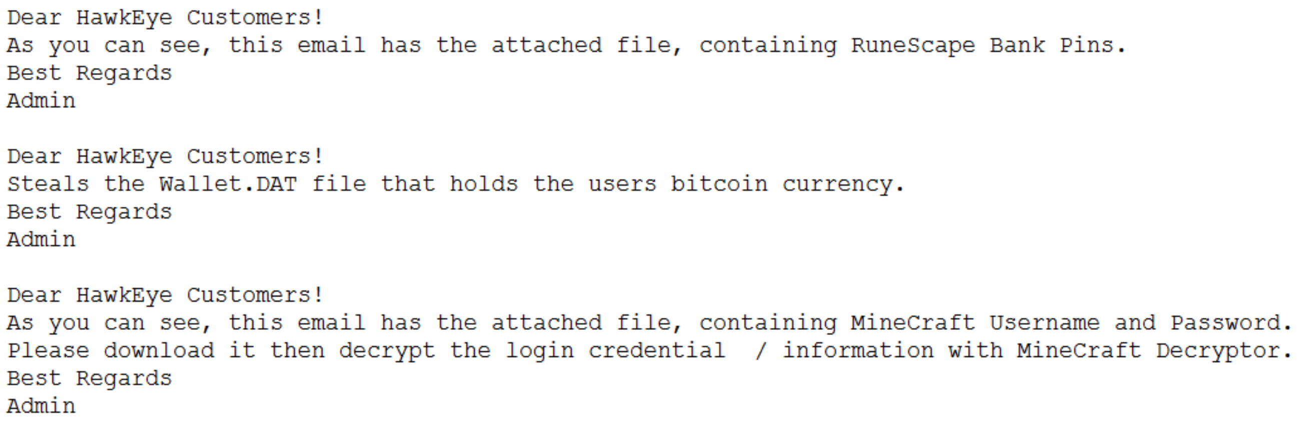
```
GET / HTTP/1.1
Host: whatismyipaddress.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 59
Date: Tue, 27 Jun 2017 07:44:12 GMT
Connection: keep-alive
```

Figure 8 : External IP Address Query

As noted, the malware sends gathered system information and security program data to the C2 server after the external IP address is known. HawkEye can be configured to send this information through multiple methods, including via email or FTP.

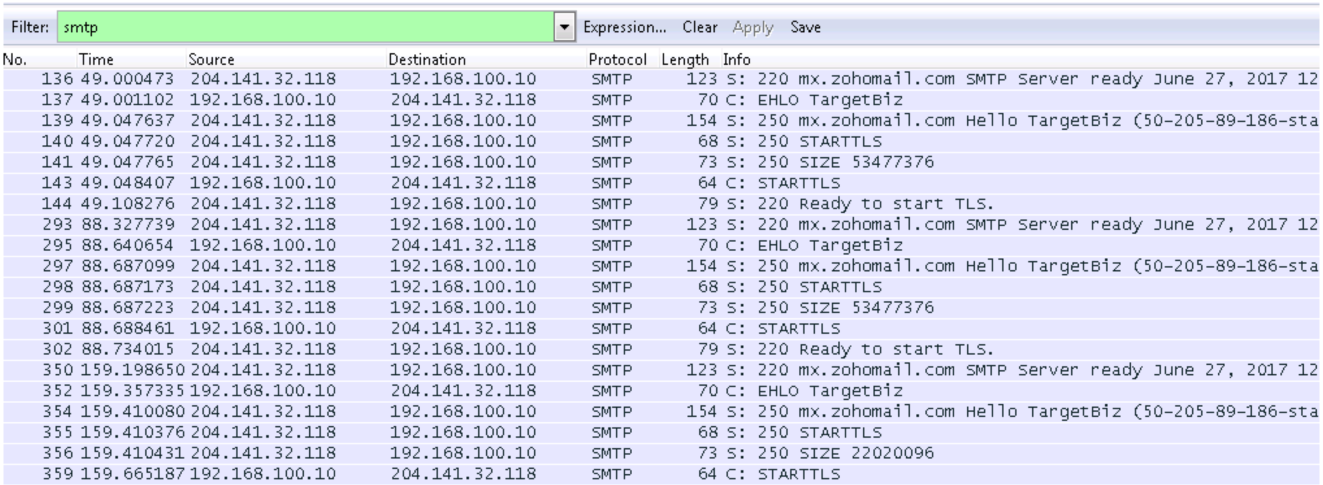
In addition to the system data, the malware will upload any collected credentials from email and web browser applications. To do this, the malware will validate that holdermail.txt and holderweb.txt exist and send the data to the C2 server. After the data is exfiltrated, the TXT files are deleted from the victim’s machine.

In this campaign, the HawkEye payload was configured to upload the data via email. Once the extracted data is received by the C2 server, the server sends emails to the threat actors behind the campaign to notify them that new stolen information is available. Figure 9 shows some of the email templates used in this campaign and Figure 10 shows the SMTP traffic on the network.



```
Dear HawkEye Customers!
As you can see, this email has the attached file, containing RuneScape Bank Pins.
Best Regards
Admin

Dear HawkEye Customers!
Steals the Wallet.DAT file that holds the users bitcoin currency.
Best Regards
Admin

Dear HawkEye Customers!
As you can see, this email has the attached file, containing MineCraft Username and Password.
Please download it then decrypt the login credential  / information with MineCraft Decryptor.
Best Regards
Admin
```

Figure 9: Email notification to HawkEye Customers



Filter: smtp

```
No.  Time       Source          Destination     Protocol Length Info
 136 49.000473  204.141.32.118  192.168.100.10  SMTP     123 S: 220 mx.zohomail.com SMTP Server ready June 27, 2017 12
 137 49.001102  192.168.100.10  204.141.32.118  SMTP      70 C: EHLO TargetBiz
 139 49.047637  204.141.32.118  192.168.100.10  SMTP     154 S: 250 mx.zohomail.com Hello TargetBiz (50-205-89-186-sta
 140 49.047720  204.141.32.118  192.168.100.10  SMTP      68 S: 250 STARTTLS
 141 49.047765  204.141.32.118  192.168.100.10  SMTP      73 S: 250 SIZE 53477376
 143 49.048407  192.168.100.10  204.141.32.118  SMTP      64 C: STARTTLS
 144 49.108276  204.141.32.118  192.168.100.10  SMTP      79 S: 220 Ready to start TLS.
 293 88.327739  204.141.32.118  192.168.100.10  SMTP     123 S: 220 mx.zohomail.com SMTP Server ready June 27, 2017 12
 295 88.640654  192.168.100.10  204.141.32.118  SMTP      70 C: EHLO TargetBiz
 297 88.687099  204.141.32.118  192.168.100.10  SMTP     154 S: 250 mx.zohomail.com Hello TargetBiz (50-205-89-186-sta
 298 88.687173  204.141.32.118  192.168.100.10  SMTP      68 S: 250 STARTTLS
 299 88.687223  204.141.32.118  192.168.100.10  SMTP      73 S: 250 SIZE 53477376
 301 88.688461  192.168.100.10  204.141.32.118  SMTP      64 C: STARTTLS
 302 88.734015  204.141.32.118  192.168.100.10  SMTP      79 S: 220 Ready to start TLS.
 350 159.198650 204.141.32.118  192.168.100.10  SMTP     123 S: 220 mx.zohomail.com SMTP Server ready June 27, 2017 12
 352 159.357335 192.168.100.10  204.141.32.118  SMTP      70 C: EHLO TargetBiz
 354 159.410080 204.141.32.118  192.168.100.10  SMTP     154 S: 250 mx.zohomail.com Hello TargetBiz (50-205-89-186-sta
 355 159.410376 204.141.32.118  192.168.100.10  SMTP      68 S: 250 STARTTLS
 356 159.410431 204.141.32.118  192.168.100.10  SMTP      73 S: 250 SIZE 22020096
 359 159.665187 192.168.100.10  204.141.32.118  SMTP      64 C: STARTTLS
```

Figure 10: SMTP Handshake

HawkEye User Base

HawkEye is a versatile Trojan used by diverse actors for multiple purposes. The malware has been sold through a public-facing website, which has allowed many different operators to use it. As is often the case with commercial Trojans, HawkEye offers a variety of functions for stealing stored data, grabbing form data, self-spreading, and performing other functions. Consequently, HawkEye may facilitate a number of different exploitative operations in compromised environments, and can be used by actors with a wide range of motivations. We have seen different HawkEye campaigns infecting organizations across many sectors globally, and stealing user credentials for diverse online services. This particular campaign represents one segment of the numerous HawkEye activity sets.

Some notable threat operations where we have previously reported HawkEye use include business email compromise campaigns, phishing against Middle Eastern organizations, and prolific spam operations (get an iSIGHT intelligence subscription to learn more about these campaigns).

Conclusion

Based on previous observations, the phishing and lure techniques used in these recent HawkEye campaigns have remained consistent, as have the HawkEye binaries and associated payloads. However, the attackers have altered the initial delivery method to use an embedded OLE object, as opposed to past methods such as a macro embedded in a Word document. The threat landscape is continiously evolving, and we expect to see more new tricks and tactics being used by the actors using this malware family.

FireEye Multi Vector Execution (MVX) engine is able to recognize and block this threat.

Acknowledgement

Special thanks to John Miller and Nart Villeneuve for their contributions to this blog.

Previous Post
Next Post