Deep Analysis of New Emotet Variant – Part 2

By Xiaopeng Zhang

Published: 2017-05-09 · Archived: 2026-04-05 15:54:26 UTC

Background

This is the second part of FortiGuard Labs' deep analysis of the new Emotet variant. In the first part of the analysis we demonstrated that by bypassing the server-side **Anti-Debug** or **Anti-Analysis** technique we could download three or four modules (.dll files) from the C&C server. In that first blog we only analyzed one module (I named it 'module2'). In this blog, we'll review how the other modules work. Here we go.

Stealing email addresses from MS Outlook PST files

As I detailed in Part 1 of this blog, the first module we're looking at here (I've named it 'module1') is loaded in a ThreadFunction, whose main function is to go through all Outlook accounts by reading the PST files. A PST file is a personal folder file in Microsoft Outlook that stores your email messages, calendar, tasks, and other items. PST files are usually located in the "Documents\Outlook Files" folder on your computer. See Figure 1.

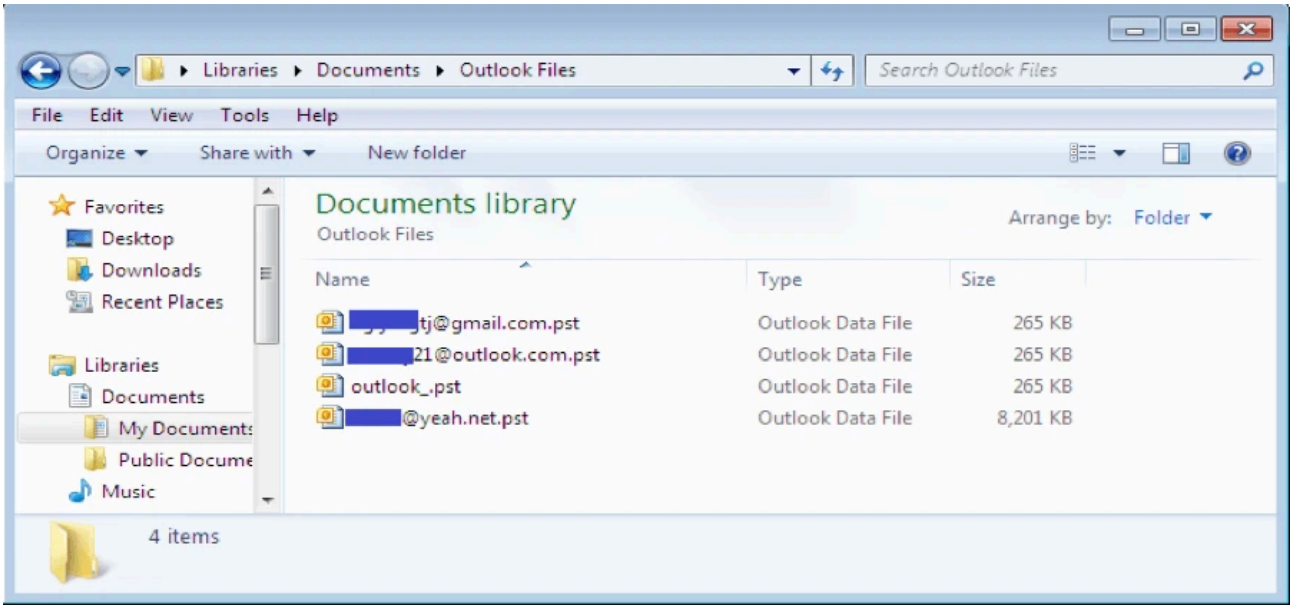

Figure 1. PST files

Microsoft has provided a group of APIs called MAPI (Microsoft Outlook Messaging API), which is the messaging architecture for Microsoft Outlook. Using the MAPIs you can operate PST files. The MAPIs are used in the module1 file.

Once module1 file is executed it creates a temporary file that is used to store the stolen Outlook version information and email addresses that have been collected. Loading MAPI functions is the next step. Figure 2

shows how, along with what it loads.

```
.text:10002A15 loc_10002A15:   ; CODE XREF: sub_10002830+1B7↑j
.text:10002A15
.text:10002A1C     lea     eax, [esp+740h+LibFileName] ; value of DLLPathEx
.text:10002A1D     push    eax             ; lpLibFileName
.text:10002A23     call    ds:LoadLibraryW
.text:10002A25     mov     edi, eax
.text:10002A29     mov     [esp+740h+nSize], edi
.text:10002A2B     test    edi, edi
.text:10002A31     jz      loc_10002F96
.text:10002A37     mov     esi, ds:GetProcAddress
.text:10002A3C     push    offset ProcName ; "MAPIInitialize"
.text:10002A3D     push    edi             ; hModule
.text:10002A3F     call    esi ; GetProcAddress
.text:10002A44     push    offset aMapiadminprofi ; "MAPIAdminProfiles"
.text:10002A45     push    edi             ; hModule
.text:10002A4A     mov     MAPIInitialize, eax
.text:10002A4C     call    esi ; GetProcAddress
.text:10002A51     push    offset aMapilogonex ; "MAPILogonEx"
.text:10002A52     push    edi             ; hModule
.text:10002A57     mov     MAPIAdminProfiles, eax
.text:10002A59     call    esi ; GetProcAddress
.text:10002A5E     push    offset aMapifreebuffer ; "MAPIFreeBuffer"
.text:10002A5F     push    edi             ; hModule
.text:10002A64     mov     MAPILogonEx, eax
.text:10002A66     call    esi ; GetProcAddress
.text:10002A6B     push    offset aMapiuninitiali ; "MAPIUninitialize"
.text:10002A6C     push    edi             ; hModule
.text:10002A71     mov     MAPIFreeBuffer, eax
.text:10002A73     call    esi ; GetProcAddress
.text:10002A79     mov     ecx, MAPIInitialize
.text:10002A7E     mov     MAPIUninitialize, eax
.text:10002A80     test    ecx, ecx
.text:10002A86     jz      loc_10002F8F
.text:10002A8D     cmp     MAPIAdminProfiles, 0
.text:10002A93     jz      loc_10002F8F
.text:10002A9A     cmp     MAPILogonEx, 0
.text:10002AA0     jz      loc_10002F8F
.text:10002AA7     cmp     MAPIFreeBuffer, 0
```

Figure 2. Loading MAPI functions

It then starts reading all PST files according to the Outlook accounts on the computer, going through all email messages with an unread status in every folder (Inbox, Deleted Items, Junk E-mail, Sent Items, etc.) under one email account. It steals the sender name and the email address from each unread email. Figure 3 shows a sample unread email about a Facebook notification that was sent to me.

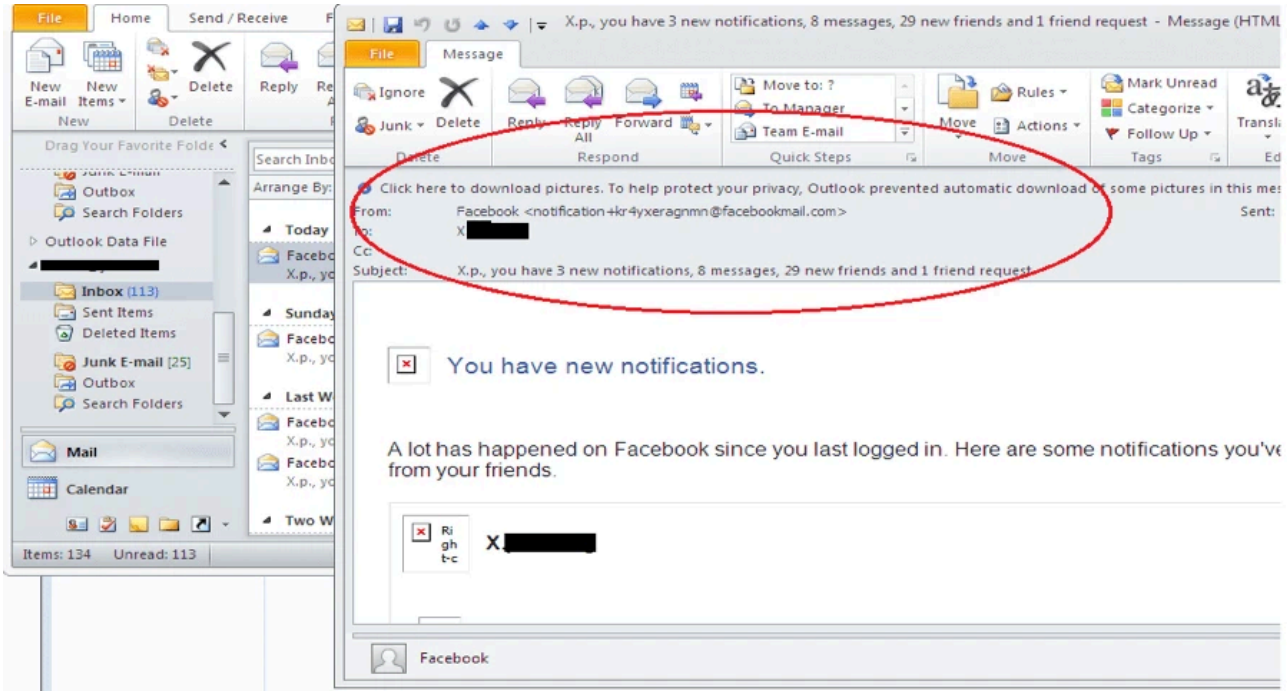

Figure 3. Sample unread email message

Figure 4 shows what module1 has stolen from the unread email message shown in Figure 3. “Facebook” is the sender name, and “notification+kr4yxeragnmn@facebookmail.com” is the sender’s email address.

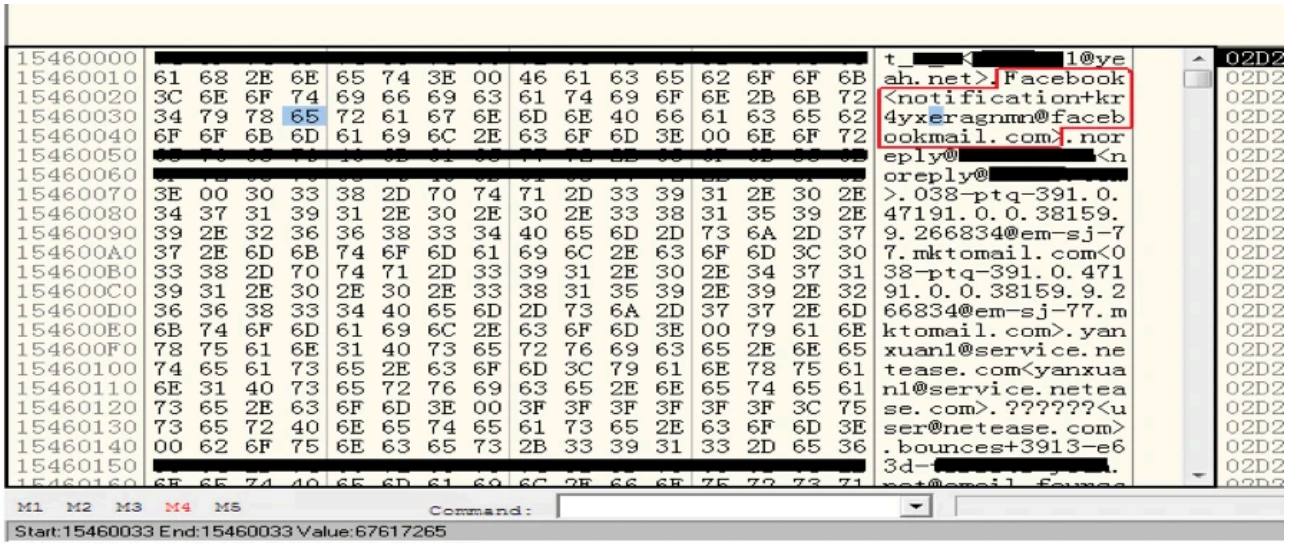

Figure 4. The stolen email information in the memory buffer

As I mentioned before, the stolen data is saved in a temporary file. In this case, it’s “AE74.tmp.” It will be read when module1 prepares to encrypt and send the stolen information to its C&C server. Figure 5 shows the data before encryption, which is read from “AE74.tmp.”

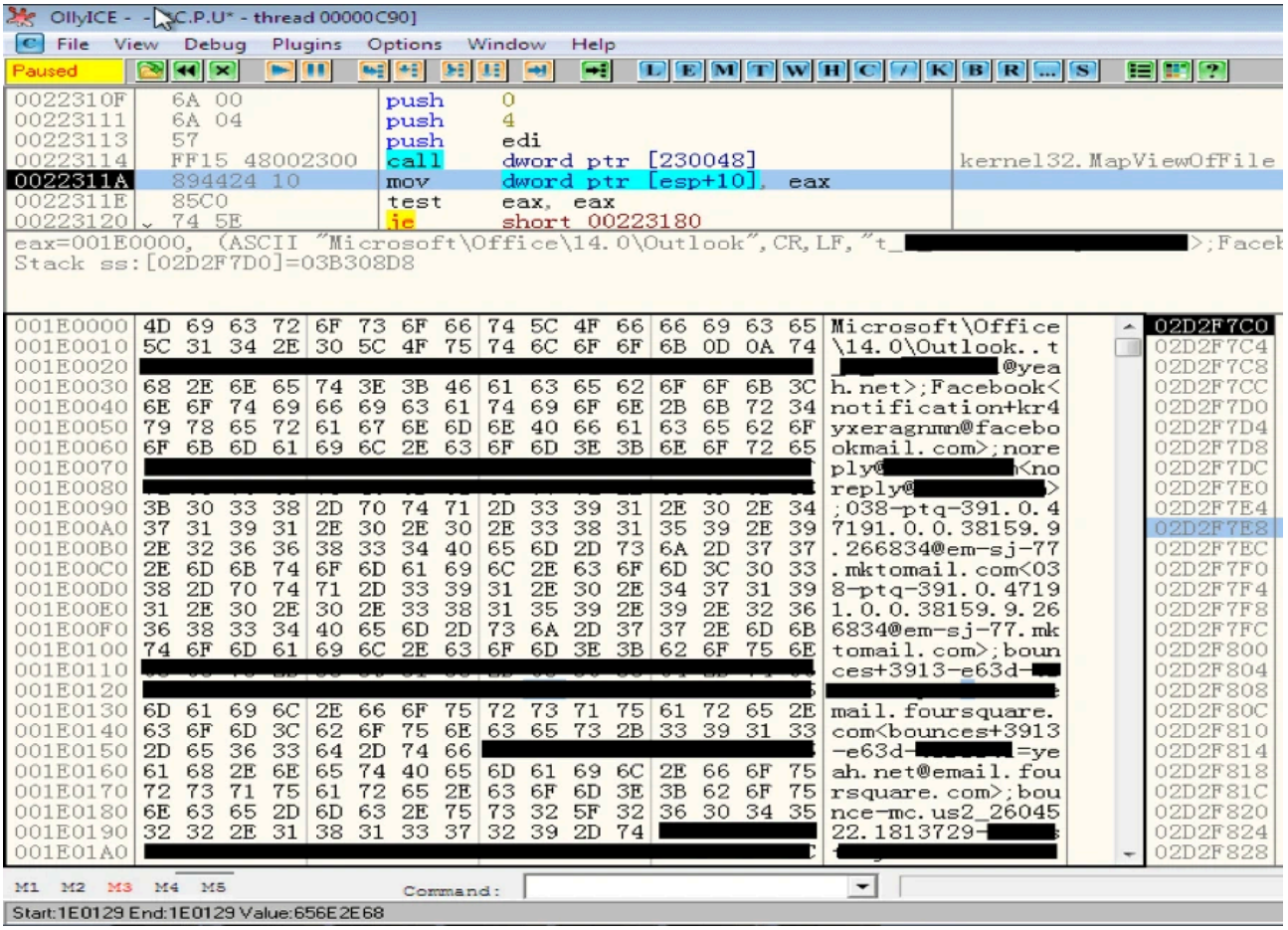

Figure 5. Data before encryption

As you can see, it contains the Outlook version and stolen email information. Once encrypted, the data will be sent to the C&C server through a “POST” request. Figure 6 is the packet screenshot from WireShark.

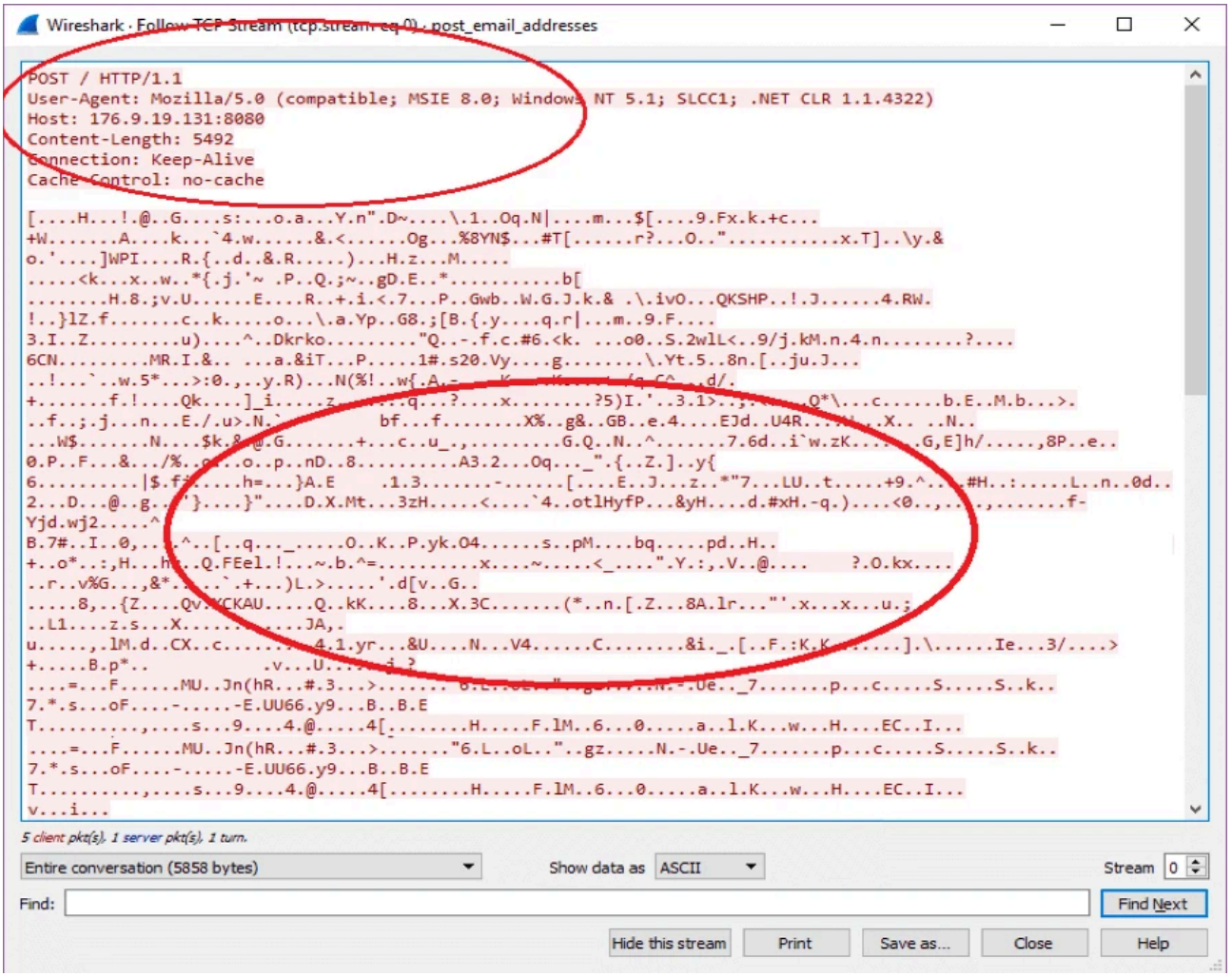

Figure 6. Sending the encrypted data to the C&C server

Sending spam using the C&C server template

This is the largest Emotet module (I have named it ‘module4’) of the malware’s four modules. Its main function is to send spam to the email addresses which were stolen and sent to the C&C server. When it is executed in a thread it generates a GUID by calling the CoCreateGuid function. It then base64-encodes the GUID and sends it as a cookie to the C&C server. The response provides the encrypted spam message, as well as the email addresses that the spam will be sent to. The two figures below show the packet from the C&C server, as well as the content after decryption.

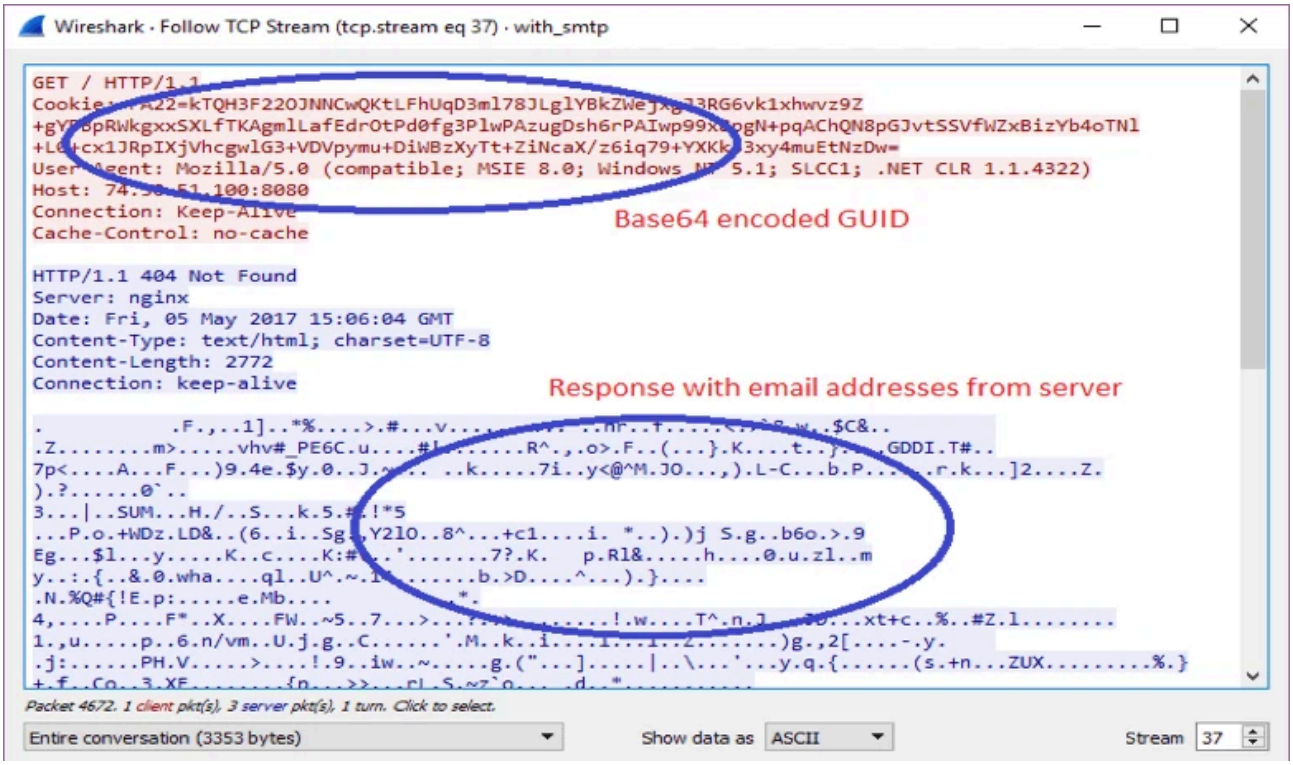

Figure 7. Sent GUID and response from the C&C server

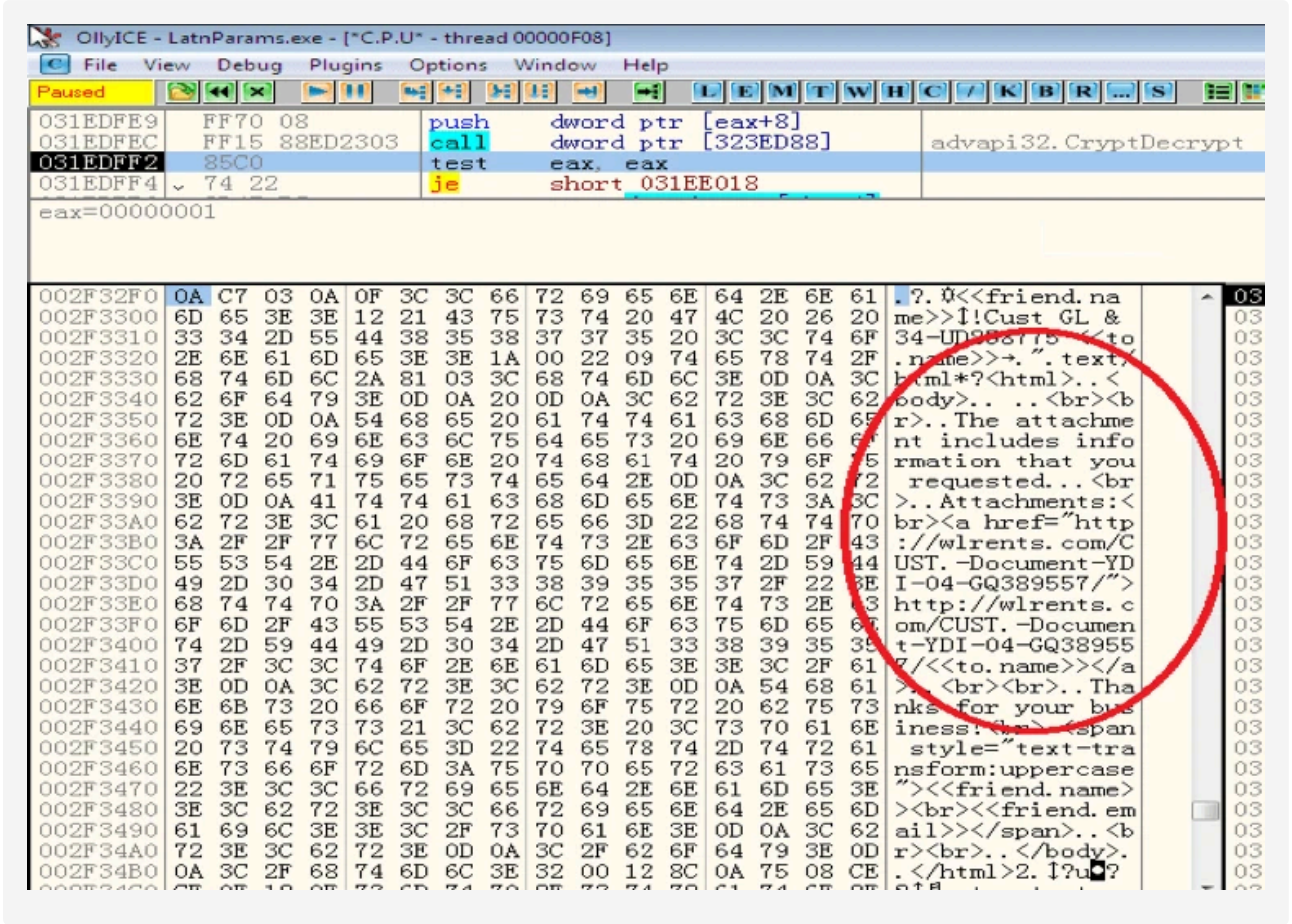

Figure 8. Decrypted spam template and email addresses

Once module4 receives the decrypted data, it reads out the spam template and the email addresses the spam message is being sent to. In module4, it supports SMTP protocol over both port 25 (regular) and port 587 (SSL). The figures below show how it uses the SMTP protocol to spread this spam, and what the spam looks like in an email client.

Figure 9. Related code and data generating SMTP packets

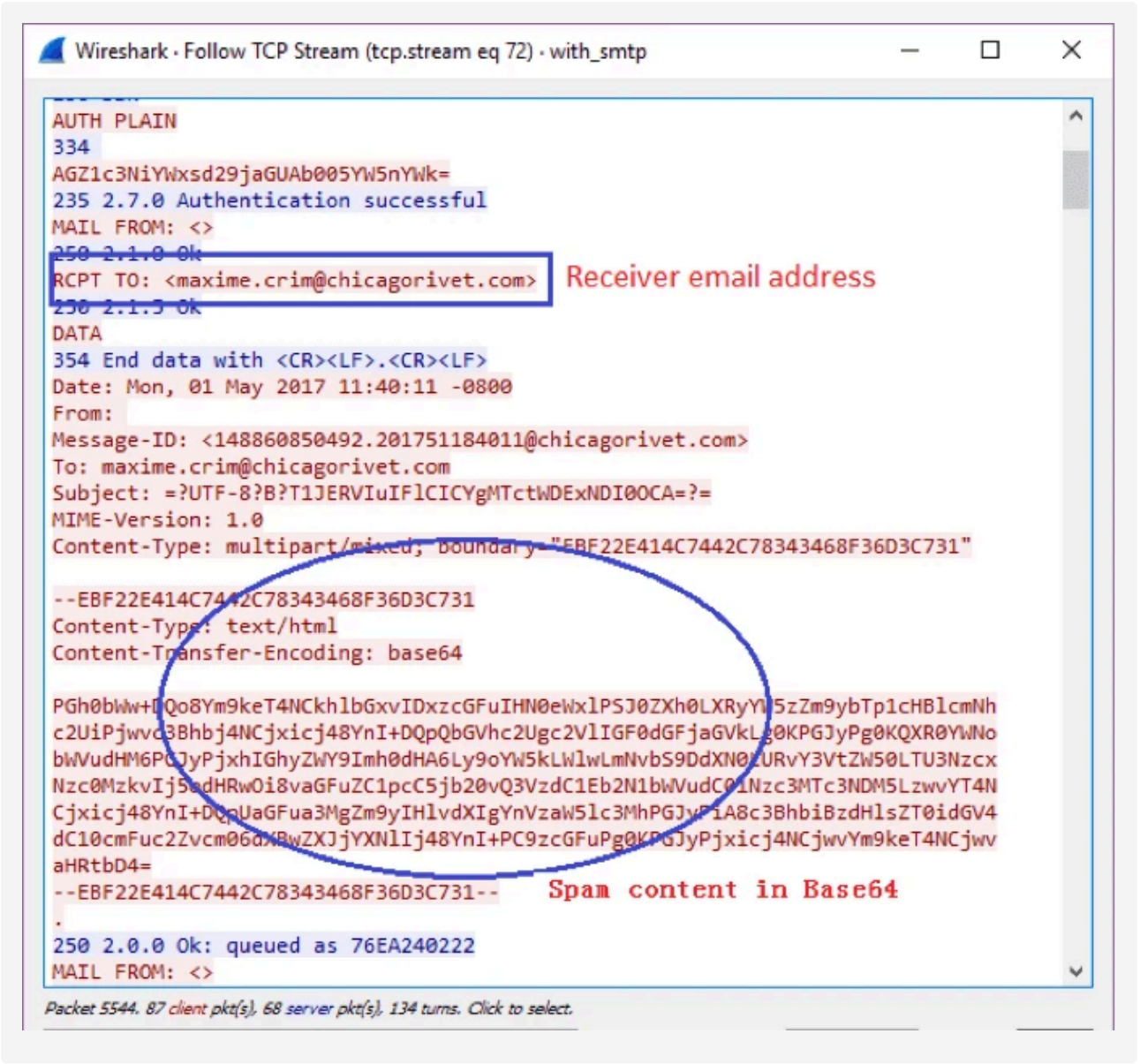

Figure 10. Spam shown in Wireshark

Figure 11. Spam shown in email client

As you can see in Figure 11, the spam attempts to trick the email recipients into opening a URL, that points to a malicious Word file. Figure 12 shows its Antivirus detection rating on VirusTotal.

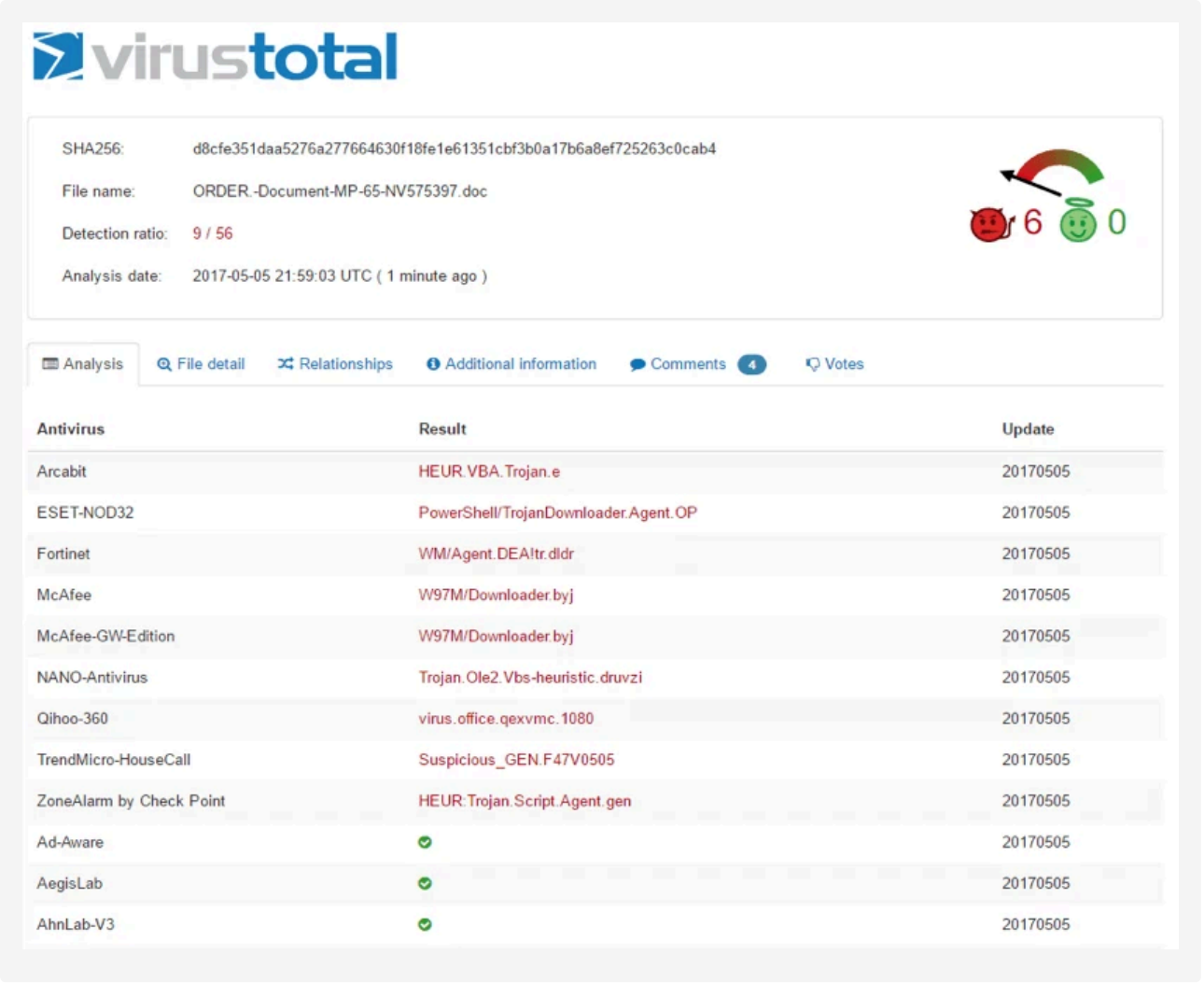

Figure 12. Antivirus detection rate on VirusTotal

Conclusion

From this deep analysis of the new Emotet variant we can see that it focuses on stealing email-related data from a victim's device, and then uses that device and the email addresses it has collected from it to send spam that can spread other malware.

NOTE: at the end of my analysis, I noticed that the Anti-Debug technique on the server side sometimes worked, and sometimes didn't.

The URL attached to the spam generated by this malware has been detected as **Malicious Websites** by the FortiGuard Webfilter service, and the downloaded Word file has been detected as **WM/Agent.DEA!tr.dldr** by the FortiGuard Antivirus service.

Summary of the four Received Modules

Module1 (size 1c000H): steals email addresses and the recipients' names from Outlook PST files.

Module2 (size 32000h): steals credentials from installed Office Outlook, IncrediMail, Group Mail, MSN Messenger, Mozilla ThunderBird, etc. The analysis of this module was provided in the first blog.

Module3 (size 70000h): steals saved information in browsers. Since it’s simple, I chose to not provide any analysis.

Module4 (size 0F0000h): sends spams to spread other malware.

IoC

URL:

"hxxp:// hand-ip.com/Cust-Document-5777177439/"

Sample SHA256:

ORDER.-Document-7023299286.doc

D8CFE351DAA5276A277664630F18FE1E61351CBF3B0A17B6A8EF725263C0CAB4

Reference

https://support.office.com/en-us/article/Introduction-to-Outlook-Data-Files-pst-and-ost-6d4197ec-1304-4b81-a17d-66d4eef30b78

https://support.microsoft.com/en-us/help/287070/how-to-manage-.pst-files-in-microsoft-outlook

https://msdn.microsoft.com/en-us/library/office/cc765775(v=office.14).aspx

---