Analysis of Emotet v4

Archived: 2026-04-05 19:37:21 UTC

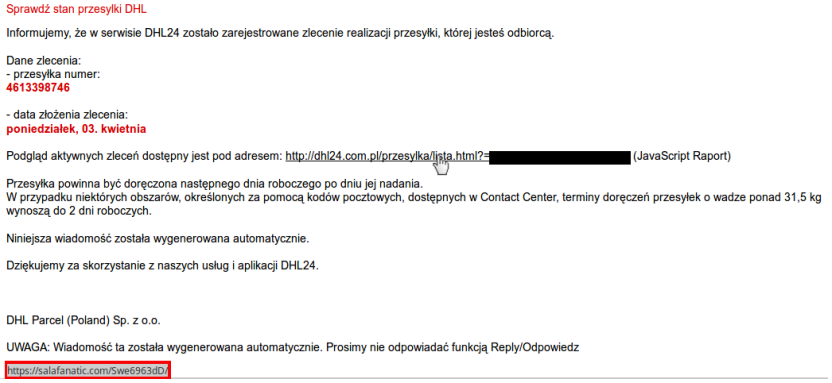
Introduction

Emotet is a modular Trojan horse, which was firstly noticed in June 2014 by Trend Micro. This malware is related to other types like Geodo, Bugat or Dridex, which are attributed by researches to the same family.

Emotet was discovered as an advanced banker – it's first campaign targeted clients of German and Austrian banks. Victims' bank accounts were infiltrated by a web browser infection which intercept communication between webpage and bank servers. In such scenario, malware hooks specific routines to sniff network activity and steal information. This technique is typical for modern banking malware and is widely known as **Man-in-the-Browser attack**.

Next, modified release of Emotet banker (v2) has taken advantage of another technique – automation of stealing money from hijacked bank accounts using ATSs (Automated Transfer Systems, more informations on page 20 of CERT Polska Report 2013). This technology is also used in other bankers. Good examples are ISFB (Gozi) or Tinba.

At the beginning of April 2017, we observed **wide malspam campaign in Poland**, distributing fraudulent mails. E-mails were imitating delivery notifications from DHL logistics company and contained malicious link, which referred to brand-new, unknown variant of Emotet.



Sprawdź stan przesyłki DHL

Informujemy, że w serwisie DHL24 zostało zarejestrowane zlecenie realizacji przesyłki, której jesteś odbiorcą.

Dane zlecenia:
- przesyłka numer:
4613398746

- data złożenia zlecenia:
poniedziałek, 03. kwietnia

Podgląd aktywnych zleceń dostępny jest pod adresem: http://dhl24.com.pl/przesylka/lista.html?= [redacted] (JavaScript Raport)

Przesyłka powinna być doręczona następnego dnia roboczego po dniu jej nadania.
W przypadku niektórych obszarów, określonych za pomocą kodów pocztowych, dostępnych w Contact Center, terminy doręczeń przesyłek o wadze ponad 31,5 kg wynoszą do 2 dni roboczych.

Niniejsza wiadomość została wygenerowana automatycznie.

Dziękujemy za skorzystanie z naszych usług i aplikacji DHL24.

DHL Parcel (Poland) Sp. z o.o.

UWAGA: Wiadomość ta została wygenerowana automatycznie. Prosimy nie odpowiadać funkcją Reply/Odpowiedz

https://salafanatic.com/Swe6963dD/

Malware distributed in this campaign differed from previously known versions. Behavior and communication methods were similar, but malware used **different encryption** and we noticed significant changes in its code. Thus we called this modification **version 4**.

Dropper

Links from the phishing campaign pointed to a dropper, which downloaded and executed malware. Dropper was written in Javascript and wasn't highly obfuscated. It was fairly easy to notice, that strings with distribution site URLs were just reversed.

```
$uaU$Fh71K_E6TQAdMPz = function(n) {
```

```
if (typeof $uaU$Fh71K_E6TQAdMPz.list[n] == "string") return $uaU$Fh71K_E6TQAdMPz.list[n].split("").reverse().join("");
return $uaU$Fh71K_E6TQAdMPz.list[n];
};

$uaU$Fh71K_E6TQAdMPz.list = [
"tamroF eliF detroppuS toN",
"llehS.tpircSW",
"tcejbOmetsySeliF.gnitpircS",
"/1506daolnwod/ku.oc.aidemlaerehte//:ptth",
"/7751daolnwod/moc.erawtfoscetni//:ptth",
"PTTHLMX.2LMXSM",
"/3030daolnwod/moc.yhpargotohpnivrinad//:ptth",
"/3946daolnwod/moc.aidemsretsacdnarb//:ptth",
"/4769daolnwod/lp.moc.hcetka//:ptth",
"maertS.BDODA",
".)dedoced yltcerroc t'nsaw dna tnemhcatta liame na sa tnes saw ti ,elpmaxe rof( deriaper eb ton dluoc dna degamad si elif ehT .tnemucod siht gninepo rorre na saw erehT" ];

...
```

Distribution sites found in dropper:

hxxp://etherealmedia.co.uk/download6051/
hxxp://intecsoftware.com/download1577/
hxxp://danirvinphotography.com/download0303/
hxxp://brandcastersmedia.com/download6493/
hxxp://aktech.com.pl/download9674/

Main module

An interesting thing in Emotet is its modular structure. Main module dropped by script doesn't contain anything harmful and is used only to download another modules from C&C, which perform specific tasks. Sample dropped by script is protected using some generic packer to avoid recognition by AV software.

After unpacking, malware loads libraries and resolves WinAPI routines used in encryption and communication with C&C. Names of specific functions are obfuscated and stored as array of hashes. Emotet uses simple **sdbm** hash function for this purpose. To make hashes more varied, values are additionally XORed with some constant specified in binary.

```
int hashValue = 0;
for ( char c = *libraryName; *libraryName; c = *(++libraryName) )
{
hashValue = c + 65599 * hashValue;
}
hashValue = xorKey ^ hashValue;
```

Strings that are distinctive for Emotet are also encoded using 4-byte XOR key, different for each string.

Main executable file contains also a list of IP addresses of C&C servers. Similar to previous versions, sample communicates with Command&Control using plain HTTP.

```
.data:00417460 ; int cnc_peers[]
.data:00417460 cnc_peers       db 76                   ; DATA XREF: sub_403010+3B1↑r
.data:00417461                 db 105
.data:00417462                 db 106
.data:00417463                 db 87
.data:00417464                 dd 1BBh
.data:00417468                 db 121
.data:00417469                 db 229
.data:0041746A                 db 255
.data:0041746B                 db 173
.data:0041746C                 dd 1BBh
.data:00417470                 db 141
.data:00417471                 db 177
.data:00417472                 db 79
.data:00417473                 db 178
.data:00417474                 dd 1BBh
```

Encryption

The most significant change in new version is usage of different encryption algorithm. In previous releases, communication was encrypted using RC4. In fourth version, Emotet switched to **128-bit AES in CBC mode**.

Intercepted request:

GET / HTTP/1.1
Cookie:
DD29=e8fd7YpIy2Ui+U7bz1/cQD9bH4KHshzaN2SpKoPEnC1D85K4Zrwdb6dBoHoDC5GgvcgecLN20kpk1lQxus6AJEiutWK4hBSWFbQUmt
User-Agent: Mozilla/5.0 (compatible; MSIE 8.0; Windows NT 5.1; SLCC1; .NET CLR 1.1.4322)
Host: 206.214.220.79:8080
Connection: Keep-Alive
Cache-Control: no-cache

Request body is passed in Cookie header. Cookie's key is random 16-bit hexadecimal number, with Base64-encoded binary blob as value.

After decoding, structure of request is described below:

Offset	Field name
0..95	asymmetrically encrypted 128-bit AES key used for request encryption
96..115	SHA1 hash of plaintext request body
116..x	Request body, AES-128-CBC encrypted

Before sending, malware performs key generation. In the first stage, Emotet loads 768-bit RSA public key, stored in executable. Then, AES symmetric key is generated using cryptographically secure PRNG (CryptGenKey). Finally, generated key is encrypted using previously loaded public key and attached to the request using PKCS#1v2.0 (OAEP) padding.

Cryptography is based on Microsoft CryptoAPI mechanisms.

Key generation and public key import:

```
if ( fn_CryptAcquireContextW(cryptCtx->hProv, 0, 0, PROV_RSA_AES, 0xF0000040) )
{
if ( fn_CryptDecodeObjectEx(
65537,
RSA_CSP_PUBLICKEYBLOB,
RSA_ENCODED,
RSA_ENCODED_LEN,
CRYPT_DECODE_ALLOC_FLAG,
0,
&prsaKey,
&prsaKeyLen) )
{
```

```
v2 = fn_CryptImportKey(cryptCtx->hProv, prsaKey, prsaKeyLen, 0, 0, &cryptCtx->hCryptRSA);
LocalFree(prsaKey);
if ( v2 )
{
if ( fn_CryptGenKey(cryptCtx->hProv, CALG_AES_128, CRYPT_MODE_CBC, &cryptCtx->hCryptAES) )
{
if ( fn_CryptCreateHash(cryptCtx->hProv, CALG_SHA1, 0, 0, &cryptCtx->hCryptSHA1) )
return 1;
fn_CryptDestroyKey(cryptCtx->hCryptAES);
}
fn_CryptDestroyKey(cryptCtx->hCryptRSA);
}
}
fn_CryptReleaseContext(cryptCtx->hProv, 0);
}
```

Request encryption:

```
if ( !fn_CryptDuplicateHash(cryptCtx->hCryptSHA1, 0, 0, &hHash) )
goto ERROR;
memmove(pRequest, req->bufPtr, req->bufLen);
if ( fn_CryptEncrypt(cryptCtx->hCryptAES, hHash, 1, 0, pRequest, &dwRequestLen, dwBufLen) )
{
if ( fn_CryptExportKey(cryptCtx->hCryptAES, cryptCtx->hCryptRSA, 1, CRYPT_OAEP, encKey, &encKeyLen) )
{
memmove(encReq, encKey, 96)
if ( fn_CryptGetHashParam(hHash, HP_HASHVAL, encReq + 96, &shaLen, 0) )
result = 1;
}
// ...
}
```

Communication with C&C

Received response is presented below:

```
0006f510: 0810 12b0 020a 14.. .... .... .... ..5f  .......XXXXXXXX_
0006f520: XXXX 5f44 XX33 XX32 XX42 XX15 1601 0100  XX_DX3X2XBX.....
0006f530: 1afe 015b 5379 7374 656d 2050 726f 6365  ...[System Proce
0006f540: 7373 5d2c 5379 7374 656d 2c73 6d73 732e  ss],System,smss.
0006f550: 6578 652c 6373 7273 732e 6578 652c 7769  exe,csrss.exe,wi
0006f560: 6e69 6e69 742e 6578 652c 7365 7276 6963  ninit.exe,servic
0006f570: 6573 2e65 7865 2c77 696e 6c6f 676f 6e2e  es.exe,winlogon.
0006f580: 6578 652c 6c73 6173 732e 6578 652c 6c73  exe,lsass.exe,ls
0006f590: 6d2e 6578 652c 7376 6368 6f73 742e 6578  m.exe,svchost.ex
0006f5a0: 652c 7370 6f6f 6c73 762e 6578 652c 6477  e,spoolsv.exe,dw
0006f5b0: 6d2e 6578 652c 6578 706c 6f72 6572 2e65  m.exe,explorer.e
...
0006f620: 2e65 7865 2c64 6c6c 686f 7374 2e65 7865  .exe,dllhost.exe
0006f630: 2c22 124d 6963 726f 736f 6674 204f 7574  ,".Microsoft Out
0006f640: 6c6f 6f6b 0000 0000 5852 1511 d259 0000  look....XR...Y..
```

Communication protocol is based on Google Protocol Buffers. Protocol Buffers is a mechanism, which allows developers to simply build own protocols using set of message structure declarations, written in a specific protobuf language. Protocol Buffers generates parsing and serializing modules, which can be directly used in developed solution. Protobuf supports wide set of languages, including Python, Java, PHP or C++. Using this kind of mechanisms isn't something new in malware, protobuf-based protocols can be found for example in Gootkit malware.

Unfortunately, Emotet's case is a bit different. Protobuf code inside malware is **slightly modified** and provides additional type of encoding, which is not specified in the original Protocol Buffers documentation. Because of this small difference, response can't be properly decoded using generic protobuf parsers e.g. *protoc* with *–decode_raw* argument fails.

Anyway, original protocol definitions were successfully reversed:

```
syntax = "proto2";

message RegistrationRequest {
message RegistrationRequestBody {
required string botId = 1;
required fixed32 osVersion = 2;
required string procList = 3;
required string mailClient = 4;
}

required int32 command = 1 [default = 16];
optional RegistrationRequestBody registrationRequest = 2;
}
```

Registration request contains command id (16) and some information about host operating system. Each field of **RegistrationRequestBody** structure has been described below:

botId field

This field provides information about values specific to victim's machine and probably is meant to be unique between bot instances.

[host_name]_[locale]_[host_id]
e.g. CERTCERT_PL_32122958

- **host_name** – contains only chars from 0..9a..zA..Z- charset, another chars are replaced by '?'
- **locale** – contains information about locale settings. In this case, dash '-' is also forbidden
- **host_id** – 32-bit hexadecimal value of sdbm hash (used also by API resolver) from current user login xored by Windows drive serial number.

osVersion field

32-bit field, which describes version of Windows running on infected host. It's a bit field, where each groups of bits contains specific value of OSVERSIONINFOEX structure.

Bits	Description
0..3	dwMajorVersion
4..7	dwMinorVersion
8..11	wServicePackMajor
12..15	wServicePackMinor
16..19	wProductType
20..23	SYSTEM_INFO.wProcessorArchitecture

procList field

Contains comma-separated list of currently running process names.

mailClient field

Provides information about used mail client (read from “HKLM\Software\Clients\Mail” registry key value). If it’s Microsoft Outlook and it’s MAPI DLL is 64-bit, name is followed by ” x64″ suffix.

Response

If a registration request was received, C&C server returns a list of Emotet modules. HTTP status response is always **404 Not Found**, regardless of the fact whether request was built properly or not. In this case, response body contains encrypted response.

HTTP/1.1 404 Not Found
Server: nginx
Content-Type: text/html; charset=UTF-8
Content-Length: 728740
Connection: keep-alive

alc:*qLud<d^G'>...

Structure of encrypted response is quite similar to the request structure. Encrypted payload starts at 116-byte of received message. Response is encrypted using the same AES key, which was passed in request. After successful decryption, we obtain protobuf-like message with list of MZ binaries or URLs.

	message Module
	{
	required int32 type = 1;
	required bytes blob = 2;
	}
	message ModuleResponse {
	repeated Module modules = 1 [packed=true];
	required uint32 timestamp = 2;
	}

In this case, malware uses non-standard encoding. Field *repeated Module modules = 1 [packed=true];* is illegal in protobuf language, because *packed* attribute can be used only for primitive numeric type of repeated fields. Surprisingly, list of modules is encoded like packed list of *Message* objects.

Here is a low-level C&C response description, using Protocol Buffers encoding primitives:

Type	**Name**	**Comment**
ModuleResponse		
TAG	tag	0x0a
VARINT	length of ‘modules’ list	
Module (repeated)		
VARINT	length of Module element	
TAG	‘type’ field tag	0x08
VARINT	type	
TAG	‘blob’ field type	0x12
VARINT	length of ‘blob’	
RAW	‘blob’ content	
…		

It should be noted that elements of Modules are repeated **without Module message tag**, which is specific to *packed* encoding,

type field

This field defines type of blob content and specifies method of module execution. Type field can be one of the following values:

Value	Description
1	Store in %TEMP% and execute with -U argument
2	Like '1', but without arguments
3	Download and execute file from URL specified 'blob'
4	Use own internal loader – load and execute PE file from 'blob'
5	Uninstall – delete related '.lnk' from Startup folder
default	Do nothing

Modules

In previous versions, Emotet modules were providing the following set of functionalities:

- Stealing money from bank accounts (Man-in-the-Browser attack)
- Spreading by sending spam e-mails
- Stealing mails and credentials to mail accounts
- DDoS module
- Stealing browsing history and passwords from web browser

In version 4 distributed in the last campaign, we didn't observe banking module, which is somewhat unusual for this type of malware. Behavior of other modules was quite similar to previous versions. During analysis, we successfully dropped two types of modules, described below:

Credentials stealer

In server response, we found two similar modules, which purpose was to **steal credentials from web browser and mail client**. Both modules have embedded NirSoft password recovery software inside:

- Mail PassView (Email Password-Recovery) v1.86
- WebBrowserPassView v1.80

Recovery software was embedded as XOR-encoded static blob, using 32-bit key (similar to strings). On module startup, software was decoded and stored in *%TEMP%*, and then executed with */scomma [temp file name]* parameter, which leads to dump all passwords into file contained in *%TEMP%* folder (name generated using GetTempFileNameW). Stealed data were sent to C&C server for malware spreading purpose.

Spam module

Second type of module was spam module, used for malware spreading. Firstly, module asks C&C for message template, list of recipients and list of hijacked accounts, which will be used to spam distribution.

Request structure presents as below:

	message SpamRequest {
	message SpamRequestBody {
	required string botId = 1;
	required int32 flags = 2 [default = 3];
	required string additionalData = 3;
	}
	required int32 command = 1 [default = 18];
	optional SpamRequestBody spamRequest = 2;
	}

Fields *flags* and *additionalData* specify, which data has been received from server and which we're expecting in C&C answer.

Server response looks like below:

```
message EmailAccount {
required int32 id = 1;
required string mail_server = 2;
required int32 port = 3;
required string login = 4;
required string password = 5;
required string email = 6;
}

message EmailRecipient {
required int32 id=1;
required string to_email=2;
optional string to_name=3;
required string from_email=4;
required string from_name=5;
}

message EmailResponse {
message Template {
required string from = 1 ;
required string subject = 2;
required string unk1 = 3;
required string content_type = 4;
required string msg = 5;
required string unk2 = 6;
}

optional Template template = 1;
repeated EmailAccount accounts = 2 [packed=true];
optional EmailRecipient recipients = 3 [packed=true];
required uint32 timestamp = 4;
}
```

E-mails are not sent using local account. Distribution is performed using **previously scrapped mail accounts**, which are sent to each spambot.

Message template example:

Hello <>

Thank you for your order. The details can be found below.

Invoice attached: http://aceeight.com/Cust-000564-17424/<>

This e-mail was sent by <>
<>

Summary

Basic functionality of Emotet in last campaign was just stealing credentials and spreading. Even though, malware is still active and also actively developed. Because of lack of few important modules, Emotet will be probably extended in future. In case of infection, we recommend **changing passwords** to all accounts, which credentials were stored in mail client or web browser.

Additional informations

- Detailed Kaspersky analysis from 2015 (Emotet v2 and v3)

Analysis based on sample: c53956c95100c5c0ba342977f8fc44fcad35aabc24ec44cb12bb83eee1ed34fa

MD5 of fetched modules (13th April):

0497c120248c6f00f1ac37513bd572e5
5b2d58b4104309ee9c93b455d39c7314
722268bad0d3a2e90aa148d52c60943e

C&C list

hxxp://87.106.105.76:443
hxxp://173.255.229.121:443
hxxp://178.79.177.141:443
hxxp://79.170.95.202:7080
hxxp://206.214.220.79:8080
hxxp://88.198.50.221:8080
hxxp://5.39.84.48:8080
hxxp://188.68.58.8:7080
hxxp://162.214.11.56:7080
hxxp://5.196.73.150:8080
hxxp://203.121.145.40:7080
hxxp://46.165.212.76:7080

C&C public key:

-----BEGIN PUBLIC KEY-----
MHwwDQYJKoZIhvcNAQEBBQADawAwaAJhAJ16QBv5Csq0eruFy4BvTcXmmIyeqUb3
vCCc8K/zOYOpL/Ww6FCdUpvPfs+RR/sLBalwtKmT14iRUaNmJdygnAKUIRWR1HNt
0rQRir0pD4QlkXlnZ9lZazTfyMV8BLCatwIDAQAB
-----END PUBLIC KEY-----

Yara rules:

rule emotet4_basic: trojan
{
meta:
author = "psrok1/mak"
module = "emotet"
strings:
$emotet4_rsa_public = { 8d ?? ?? 5? 8d ?? ?? 5? 6a 00 68 00 80 00 00 ff 35 [4] ff 35 [4] 6a 13 68 01 00 01 00 ff 15 [4] 85 }
$emotet4_cnc_list = { 39 ?? ?5 [4] 0f 44 ?? (FF | A3)}
condition:
all of them
}

rule emotet4: trojan
{
meta:
author = "psrok1"
module = "emotet"
strings:
$emotet4_x65599 = { 0f b6 ?? 8d ?? ?? 69 ?? 3f 00 01 00 4? 0? ?? 3? ?? 72 }
condition:

```
any of them and emotet4_basic
}

rule emotet4_spam : spambot
{
meta:
author="mak"
module="emotet"
strings:
$login="LOGIN" fullword
$startls="STARTTLS" fullword
$mailfrom="MAIL FROM:"
condition:
all of them and emotet4_basic
}
```
